Unión de DataFrames con dplyr

Rubén Sierra Serrano

2024-03-25

Índice

La función inner_join()

A lo largo del documeto se va a trabajar con diversos DataFrames procedentes de la empresa LEGO.

El DataFrame "base" se llama `sets`, dicho DataFrame tiene una columna llamada `theme_id` que no tiene sentido en sí misma, se trata de un identificador, dicho identificador nos permite unir el DataFrame `sets` con el DataFrame `themes`.

```
head(sets)
```

```
## # A tibble: 6 x 4
##   set_num   name                                                year theme_id
##   <chr>     <chr>                                              <dbl>    <dbl>
## 1 700.3-1   Medium Gift Set (ABB)                               1949      365
## 2 700.1.1-1 Single 2 x 4 Brick (ABB)                            1950      371
## 3 700.B.2-1 Single 1 x 2 x 3 Window without Glass (ABB)         1950      371
## 4 700.1-2   Extra-Large Gift Set (Mursten)                      1953      366
## 5 700.F-1   Automatic Binding Bricks - Small Brick Set (Lego Mur~  1953      371
## 6 700.24-1  Individual 2 x 12 Bricks                            1954      371
```

```
head(themes)
```

```
## # A tibble: 6 x 3
##      id name           parent_id
##   <dbl> <chr>              <dbl>
## 1     1 Technic               NA
## 2     2 Arctic Technic         1
## 3     3 Competition            1
## 4     4 Expert Builder         1
## 5     5 Model                  1
## 6     6 Airport                5
```

Para unirlos hay que emplear la función **inner_join()**. Dicha función solo conserva las observaciones que se encuentran en el primer DataFrame y que tienen una coincidencia en el segundo, une ambos DataFrames por una columna común.

```
sets %>%
  inner_join(themes, by = c("theme_id" = "id"))
```

```
## # A tibble: 4,977 x 6
##    set_num   name.x                                year theme_id name.y parent_id
##    <chr>     <chr>                                <dbl>    <dbl> <chr>      <dbl>
##  1 700.3-1   Medium Gift Set (ABB)                 1949      365 System        NA
##  2 700.1.1-1 Single 2 x 4 Brick (ABB)              1950      371 Suppl~       365
##  3 700.B.2-1 Single 1 x 2 x 3 Window without Gl~   1950      371 Suppl~       365
##  4 700.1-2   Extra-Large Gift Set (Mursten)        1953      366 Basic~       365
##  5 700.F-1   Automatic Binding Bricks - Small B~   1953      371 Suppl~       365
##  6 700.24-1  Individual 2 x 12 Bricks              1954      371 Suppl~       365
##  7 700.C.1-1 Individual 1 x 6 x 4 Panorama Wind~   1954      371 Suppl~       365
##  8 700.C.4-1 Individual 1 x 4 x 3 Window (with ~   1954      371 Suppl~       365
##  9 700.H-1   Individual 4 x 4 Corner Bricks        1954      371 Suppl~       365
## 10 1200-1    LEGO Town Plan Board, Large Plastic   1955      372 Town ~       365
## # i 4,967 more rows
```

Esto une el primer DataFrame `sets` con el segundo DataFrame `themes`. La sintaxis **by = c("theme_id" = "id")** indica que se está uniendo las tablas donde el valor de `theme_id` en el conjunto de datos `sets` es igual al valor de `id` en el conjunto de datos `themes`. Por tanto, es una relación de uno a uno (1:1).

La función **inner_join()** tiene el argumento `suffix`, el cuál permite añadir sufijos a las columnas de los conjuntos de datos de entrada que tienen nombres de columna en común, pero que no son los campos de unión.

```
sets %>%
  inner_join(themes, by = c("theme_id" = "id"), suffix = c("_set", "_theme"))
```

```
## # A tibble: 4,977 x 6
##    set_num   name_set                              year theme_id name_theme parent_id
##    <chr>     <chr>                                <dbl>    <dbl> <chr>          <dbl>
##  1 700.3-1   Medium Gift Set (ABB)                 1949      365 System            NA
##  2 700.1.1-1 Single 2 x 4 Brick (ABB)              1950      371 Supplemen~       365
##  3 700.B.2-1 Single 1 x 2 x 3 Window withou~       1950      371 Supplemen~       365
##  4 700.1-2   Extra-Large Gift Set (Mursten)        1953      366 Basic Set        365
##  5 700.F-1   Automatic Binding Bricks - Sma~       1953      371 Supplemen~       365
##  6 700.24-1  Individual 2 x 12 Bricks              1954      371 Supplemen~       365
##  7 700.C.1-1 Individual 1 x 6 x 4 Panorama ~       1954      371 Supplemen~       365
##  8 700.C.4-1 Individual 1 x 4 x 3 Window (w~       1954      371 Supplemen~       365
##  9 700.H-1   Individual 4 x 4 Corner Bricks        1954      371 Supplemen~       365
## 10 1200-1    LEGO Town Plan Board, Large Pl~       1955      372 Town Plan        365
## # i 4,967 more rows
```

Nótese que el uso del operador pipe %>% permite emplear otras funciones dentro de la consulta con el fin de obtener más información:

```
sets %>%
  inner_join(themes, by = c("theme_id" = "id"), suffix = c("_set", "_theme")) %>%
  count(name_theme, sort = TRUE)
```

```
## # A tibble: 419 x 2
##    name_theme        n
##    <chr>         <int>
##  1 Supplemental    180
##  2 Basic Set       171
##  3 Technic         144
##  4 Friends         133
##  5 Gear            122
##  6 City            120
##  7 Town            117
##  8 Ninjago          95
##  9 Service Packs    94
## 10 Star Wars        94
## # i 409 more rows
```

Unión de una relación de uno a varios (1:N)

El DataFrame `inventories` tiene una identificador `set_num` que lo relaciona con el DataFrame `sets`.

```
head(inventories)
```

```
## # A tibble: 6 x 3
##      id version set_num
##   <dbl>   <dbl> <chr>
## 1     1       1 7922-1
## 2     3       1 3931-1
## 3     4       1 6942-1
## 4    15       1 5158-1
## 5    16       1 903-1
## 6    17       1 850950-1
```

Al unir ambos DataFrames de la misma forma que se menciono anteriormente (el identificador que permite unir ambos DataFrame tiene el mismo nombre, por tanto, no es necesario igualar el identificador dentro del parámetro `by` en contraposición al código anterior):

```
sets %>%
  inner_join(inventories, by = c("set_num"))
```

```
## # A tibble: 5,056 x 6
##    set_num   name                                      year theme_id    id version
##    <chr>     <chr>                                    <dbl>    <dbl> <dbl>   <dbl>
##  1 700.3-1   Medium Gift Set (ABB)                     1949      365 24197       1
##  2 700.3-1   Medium Gift Set (ABB)                     1949      365 24214       2
##  3 700.3-1   Medium Gift Set (ABB)                     1949      365 24215       3
##  4 700.1.1-1 Single 2 x 4 Brick (ABB)                  1950      371 11831       1
##  5 700.1.1-1 Single 2 x 4 Brick (ABB)                  1950      371 24230       2
##  6 700.1.1-1 Single 2 x 4 Brick (ABB)                  1950      371 24231       3
##  7 700.1.1-1 Single 2 x 4 Brick (ABB)                  1950      371 24232       4
##  8 700.1.1-1 Single 2 x 4 Brick (ABB)                  1950      371 24233       5
##  9 700.B.2-1 Single 1 x 2 x 3 Window without Glass~    1950      371   537       1
## 10 700.B.2-1 Single 1 x 2 x 3 Window without Glass~    1950      371 24240       2
## # i 5,046 more rows
```

Nótese que previamente el DataFrame tenía 4.977 observaciones y que tras la unión, tiene 5.056 observaciones. Esto se debe a que se trata de una relación uno a varios (1:N); es decir, un mismo set puede tener varias versiones. Para corroborarlo, se puede filtrar el DataFrame resultante de tal forma que sólo se muestre la primera versión:

```
sets %>%
  inner_join(inventories, by = c("set_num")) %>%
  filter(version == 1)
```

```
## # A tibble: 4,976 x 6
##    set_num   name                                      year theme_id    id version
##    <chr>     <chr>                                    <dbl>    <dbl> <dbl>   <dbl>
##  1 700.3-1   Medium Gift Set (ABB)                     1949      365 24197       1
##  2 700.1.1-1 Single 2 x 4 Brick (ABB)                  1950      371 11831       1
##  3 700.B.2-1 Single 1 x 2 x 3 Window without Glass~    1950      371   537       1
##  4 700.1-2   Extra-Large Gift Set (Mursten)            1953      366 12985       1
##  5 700.F-1   Automatic Binding Bricks - Small Bric~    1953      371 11265       1
##  6 700.24-1  Individual 2 x 12 Bricks                  1954      371  7645       1
##  7 700.C.1-1 Individual 1 x 6 x 4 Panorama Window ~    1954      371  3896       1
##  8 700.C.4-1 Individual 1 x 4 x 3 Window (with gla~    1954      371  3663       1
##  9 700.H-1   Individual 4 x 4 Corner Bricks           1954      371 15503       1
## 10 1200-1    LEGO Town Plan Board, Large Plastic      1955      372 10761       1
## # i 4,966 more rows
```

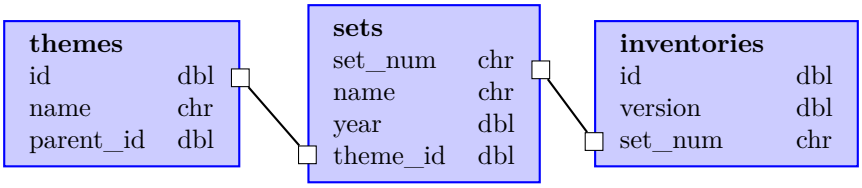
Este DataFrame tiene 4.976 observaciones, lo que significa que un set no tiene una primera versión.

Unión de más de dos DataFrames

Gracias al operador pipe %>% es posible unir de manera simultáneamente más de dos DataFrames con la función **inner_join()**, aquí toma especial relevancia el argumento `suffix` para no confundir atributos:

```
sets %>%
  inner_join(inventories, by = c("set_num")) %>%
  inner_join(themes, by = c("theme_id" = "id"), suffix = c("_set", "_theme"))
```

```
## # A tibble: 5,056 x 8
##    set_num   name_set           year theme_id    id version name_theme parent_id
##    <chr>     <chr>             <dbl>    <dbl> <dbl>   <dbl> <chr>          <dbl>
##  1 700.3-1   Medium Gift Set ~  1949      365 24197       1 System            NA
##  2 700.3-1   Medium Gift Set ~  1949      365 24214       2 System            NA
##  3 700.3-1   Medium Gift Set ~  1949      365 24215       3 System            NA
##  4 700.1.1-1 Single 2 x 4 Bri~  1950      371 11831       1 Supplemen~       365
##  5 700.1.1-1 Single 2 x 4 Bri~  1950      371 24230       2 Supplemen~       365
##  6 700.1.1-1 Single 2 x 4 Bri~  1950      371 24231       3 Supplemen~       365
##  7 700.1.1-1 Single 2 x 4 Bri~  1950      371 24232       4 Supplemen~       365
##  8 700.1.1-1 Single 2 x 4 Bri~  1950      371 24233       5 Supplemen~       365
##  9 700.B.2-1 Single 1 x 2 x 3~  1950      371   537       1 Supplemen~       365
## 10 700.B.2-1 Single 1 x 2 x 3~  1950      371 24240       2 Supplemen~       365
## # i 5,046 more rows
```



Uniones por la izquierda y por la derecha

La función left_join()

La función **left_join()** combina dos DataFrames basándose en una columna de identificación, manteniendo todas las filas del primer DataFrame (llamado "izquierdo") e incorporando las correspondientes filas del segundo DataFrame (llamado "derecho") que coincidan con la columna de identificación. Si no hay coincidencias en el segundo DataFrame, se asignan valores NA a las columnas correspondientes.

En el siguiente código se juntan los DataFrames `inventories` e `inventory_part`, de tal forma que el DataFrame resultante guarda las piezas de LEGO dentro de cada set.

```
inventory_parts_joined <- inventories %>%
  inner_join(inventory_parts, by = c("id" = "inventory_id")) %>%
  select(-id, -version) %>%
  arrange(desc(quantity))

head(inventory_parts_joined)
```

```
## # A tibble: 6 x 4
##   set_num  part_num color_id quantity
##   <chr>    <chr>       <dbl>    <dbl>
## 1 40179-1  3024           72      900
## 2 40179-1  3024           15      900
## 3 40179-1  3024            0      900
## 4 40179-1  3024           71      900
## 5 40179-1  3024           14      900
## 6 k34434-1 3024           15      810
```

A continuación, se obtiene los sets del batmobile y del batwing:

```
batmobile <- inventory_parts_joined %>%
  filter(set_num == "7784-1") %>%
  select(-set_num)

batmobile
```

```
## # A tibble: 173 x 3
##    part_num color_id quantity
##    <chr>       <dbl>    <dbl>
##  1 3023           72       62
##  2 2780            0       28
##  3 50950           0       28
##  4 3004           71       26
##  5 43093           1       25
##  6 3004            0       23
##  7 3010            0       21
##  8 30363           0       21
##  9 32123b         14       19
## 10 3622            0       18
## # i 163 more rows
```

```r
batwing <- inventory_parts_joined %>%
  filter(set_num == "70916-1") %>%
  select(-set_num)

batwing
```

```
## # A tibble: 309 x 3
##    part_num color_id quantity
##    <chr>       <dbl>    <dbl>
##  1 3023            0       22
##  2 3024            0       22
##  3 3623            0       20
##  4 11477           0       18
##  5 99207          71       18
##  6 2780            0       17
##  7 3666            0       16
##  8 22385           0       14
##  9 3710            0       14
## 10 99563           0       13
## # i 299 more rows
```

Por tanto, los DataFrames albergan las piezas de cada set, su color y su cantidad.

El siguiente DataFrame muestra las piezas que comparten ambos sets y el color y la cantidad de estas:

```r
batmobile %>%
  inner_join(batwing, by = c("part_num"), suffix = c("_batmobile", "_batwing"))
```

```
## # A tibble: 157 x 5
##    part_num color_id_batmobile quantity_batmobile color_id_batwing
##    <chr>                 <dbl>              <dbl>            <dbl>
##  1 3023                     72                 62                0
##  2 3023                     72                 62               70
##  3 3023                     72                 62               71
##  4 3023                     72                 62               36
##  5 2780                      0                 28                0
##  6 50950                     0                 28                0
##  7 3004                     71                 26                1
##  8 3004                     71                 26                0
##  9 3004                     71                 26                4
## 10 3004                     71                 26               71
## # i 147 more rows
## # i 1 more variable: quantity_batwing <dbl>
```

Al usar **left_join()**, el DataFrame resultante contiene todas las observaciones del DataFrame izquierdo y las que compartan ambos, en este caso el DataFrame izquierdo es `batmobile`:

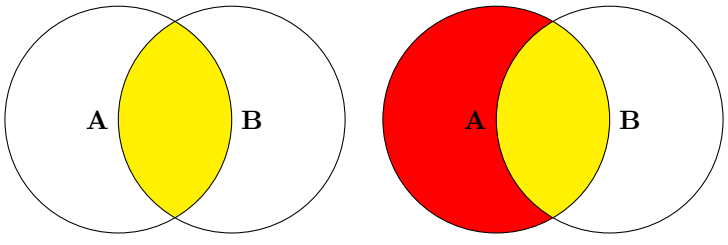
```
batmobile %>%
  left_join(batwing, by = c("part_num"), suffix = c("_batmobile", "_batwing"))
```

```
## # A tibble: 250 x 5
##    part_num color_id_batmobile quantity_batmobile color_id_batwing
##    <chr>                 <dbl>              <dbl>            <dbl>
##  1 3023                     72                 62                0
##  2 3023                     72                 62               70
##  3 3023                     72                 62               71
##  4 3023                     72                 62               36
##  5 2780                      0                 28                0
##  6 50950                     0                 28                0
##  7 3004                     71                 26                1
##  8 3004                     71                 26                0
##  9 3004                     71                 26                4
## 10 3004                     71                 26               71
## # i 240 more rows
## # i 1 more variable: quantity_batwing <dbl>
```

El argumento `by` permite unir empleando varias columnas de identificación (es válido para cualquier función de de la familia **join()**)

```
batmobile %>%
  left_join(batwing, by = c("part_num","color_id"), suffix = c("_batmobile", "_batwing"))
```

```
## # A tibble: 173 x 4
##    part_num color_id quantity_batmobile quantity_batwing
##    <chr>       <dbl>              <dbl>            <dbl>
##  1 3023           72                 62               NA
##  2 2780            0                 28               17
##  3 50950           0                 28                2
##  4 3004           71                 26                2
##  5 43093           1                 25                6
##  6 3004            0                 23                4
##  7 3010            0                 21               NA
##  8 30363           0                 21               NA
##  9 32123b         14                 19               NA
## 10 3622            0                 18                2
## # i 163 more rows
```



inner_join() **left_join()**

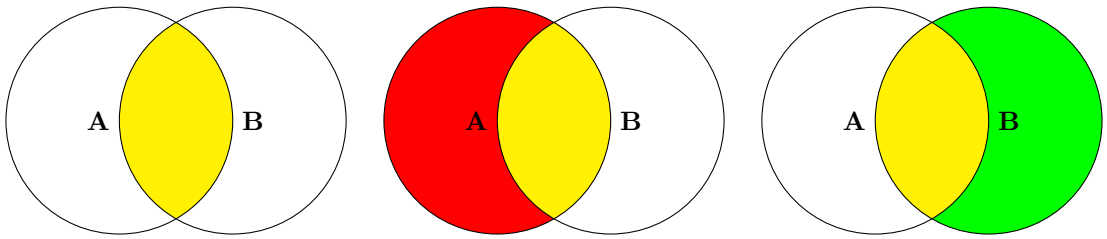
La función right_join()

La función **right_join()** combina dos DataFrames basándose en una columna de identificación, manteniendo todas las filas del segundo DataFrame (el derecho) e incorporando las correspondientes filas del primer DataFrame (el izquierdo) que coincidan con la columna de identificación. Si no hay coincidencias en el primer DataFrame, se asignan valores NA a las columnas correspondientes.

```
batmobile %>%
  right_join(batwing, by = c("part_num","color_id"), suffix = c("_batmobile", "_batwing"))
```

```
## # A tibble: 312 x 4
##    part_num color_id quantity_batmobile quantity_batwing
##    <chr>       <dbl>              <dbl>             <dbl>
##  1 2780            0                 28                17
##  2 50950           0                 28                 2
##  3 3004           71                 26                 2
##  4 43093           1                 25                 6
##  5 3004            0                 23                 4
##  6 3622            0                 18                 2
##  7 4286            0                 16                 1
##  8 3039            0                 12                 2
##  9 4274           71                 12                 7
## 10 3001            0                 11                 4
## # i 302 more rows
```

Nótese que **right_join()** y **left_join()** son funciones complementarias que realizan operaciones de unión en dos conjuntos de datos, pero en direcciones opuestas, por tanto, son "imágenes espejo".



inner_join() **left_join()** **right_join()**

La función replace_na()

Al trabajar con DataFrames obtenidos mediante alguna de las funciones mencionadas anteriormente, la presencia de NAs en los datos puede resultar molesta. Para abordar este problema, se puede emplear la función **replace_na()** de la librería **tidyr**, la cual asigna un valor específico a los NAs, permitiendo así manejarlos de manera más conveniente en el análisis de los datos.

```
sets %>%
  count(theme_id) %>%
  right_join(themes, by = c("theme_id" = "id")) %>%
  replace_na(list(n = 0))
```

```
## # A tibble: 665 x 4
##    theme_id     n name           parent_id
##       <dbl> <int> <chr>              <dbl>
##  1        1    58 Technic               NA
##  2        2     1 Arctic Technic         1
##  3        3     4 Competition            1
##  4        4    13 Expert Builder         1
##  5        5     6 Model                  1
##  6        6     7 Airport                5
##  7        7    20 Construction           5
##  8        9     2 Fire                   5
##  9       10     3 Harbor                 5
## 10       11    12 Off-Road               5
## # i 655 more rows
```

Al comprobarlo, efectivamente no hay NAs en el atributo `n`:

```
sets %>%
  count(theme_id) %>%
  right_join(themes, by = c("theme_id" = "id")) %>%
  replace_na(list(n = 0)) %>%
  filter(is.na(n))
```

```
## # A tibble: 0 x 4
## # i 4 variables: theme_id <dbl>, n <int>, name <chr>, parent_id <dbl>
```

Relaciones de DataFrames consigo mismos

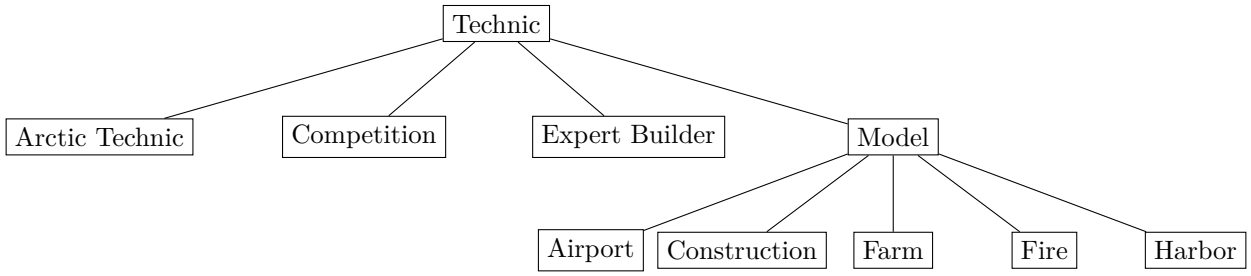
Una relación de DataFrame consigo mismo es una situación en la que un DataFrame se asocia con sí misma mediante una relación definida por uno o más campos. Esta práctica es común cuando se necesita modelar datos que tienen una estructura jerárquica o que están vinculados a sí mismos de alguna manera significativa.

El DataFrame `themes` contiene dos atributos: `id` y `parent_id`. La presencia de este último atributo indica que el DataFrame establece una relación consigo mismo, lo que implica la existencia de una jerarquía dentro de los datos que representa.

```
themes
```

```
## # A tibble: 665 x 3
##       id name            parent_id
##    <dbl> <chr>               <dbl>
##  1     1 Technic                NA
##  2     2 Arctic Technic          1
##  3     3 Competition             1
##  4     4 Expert Builder          1
##  5     5 Model                   1
##  6     6 Airport                 5
##  7     7 Construction            5
##  8     8 Farm                    5
##  9     9 Fire                    5
## 10    10 Harbor                  5
## # i 655 more rows
```

Por ejemplo, los temas `Arctic Technic`, `Competition`, `Expert Builder` y `Model` todos tienen el `parent_id` 1. Observando la primera columna, se puede ver que ese ID corresponde al primer tema, llamado `Technic`. También se puede observar que las siguientes cinco filas tienen el `parent_id` 5, el cual corresponde al tema `Model`.



Al aplicar **inner_join()** al DataFrame consigo mismo, se obtiene un DataFrame que muestra las relaciones padre-hijo:

```
themes %>%
  inner_join(themes, by = c("parent_id" = "id"), suffix = c("_hijo","_padre"))
```

```
## # A tibble: 544 x 5
##       id name_hijo      parent_id name_padre parent_id_padre
##    <dbl> <chr>              <dbl> <chr>                <dbl>
##  1     2 Arctic Technic         1 Technic                 NA
##  2     3 Competition            1 Technic                 NA
##  3     4 Expert Builder         1 Technic                 NA
##  4     5 Model                  1 Technic                 NA
##  5     6 Airport                5 Model                    1
##  6     7 Construction           5 Model                    1
##  7     8 Farm                   5 Model                    1
##  8     9 Fire                   5 Model                    1
##  9    10 Harbor                 5 Model                    1
## 10    11 Off-Road               5 Model                    1
## # i 534 more rows
```

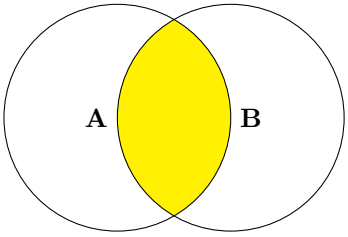
Al aplicar otra vez **inner_join()** se obtiene también las relaciones padre-nieto:

```
themes %>%
  inner_join(themes, by = c("id" = "parent_id"), suffix = c("_padre", "_hijo")) %>%
  inner_join(themes, by = c("id_hijo" = "parent_id"), suffix = c("_padre", "_nieto"))
```

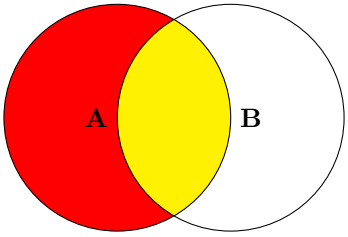
```
## # A tibble: 158 x 7
##    id_padre name_padre parent_id id_hijo name_hijo id_nieto name
##       <dbl> <chr>          <dbl>   <dbl> <chr>        <dbl> <chr>
##  1        1 Technic           NA       5 Model            6 Airport
##  2        1 Technic           NA       5 Model            7 Construction
##  3        1 Technic           NA       5 Model            8 Farm
##  4        1 Technic           NA       5 Model            9 Fire
##  5        1 Technic           NA       5 Model           10 Harbor
##  6        1 Technic           NA       5 Model           11 Off-Road
##  7        1 Technic           NA       5 Model           12 Race
##  8        1 Technic           NA       5 Model           13 Riding Cycle
##  9        1 Technic           NA       5 Model           14 Robot
## 10        1 Technic           NA       5 Model           15 Traffic
## # i 148 more rows
```

La función full_join()

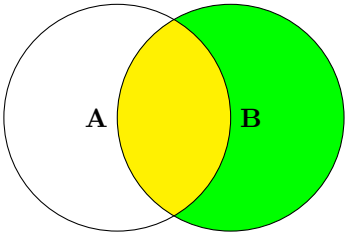
La función **full_join()** une dos DataFrames manteniendo todas las filas de ambos DataFrames, y rellenando con valores nulos en aquellos casos donde no hay coincidencias en la columna (o columnas) común. Esto significa que el resultado final contendrá todas las combinaciones posibles de filas de ambos DataFrames, incluso si no hay una correspondencia directa entre ellos en la columna utilizada para la unión.



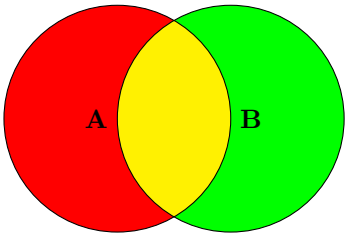
inner_join()



left_join()



right_join()



full_join()

```
batmobile %>%
  full_join(batwing, by = c("part_num", "color_id"), suffix = c("_batmobile","_batwing"))
```

```
## # A tibble: 440 x 4
##    part_num color_id quantity_batmobile quantity_batwing
##    <chr>       <dbl>              <dbl>            <dbl>
##  1 3023           72                 62               NA
##  2 2780            0                 28               17
##  3 50950           0                 28                2
##  4 3004           71                 26                2
##  5 43093           1                 25                6
##  6 3004            0                 23                4
##  7 3010            0                 21               NA
##  8 30363           0                 21               NA
##  9 32123b         14                 19               NA
## 10 3622            0                 18                2
## # i 430 more rows
```

De nuevo, no resulta cómodo trabajar con valores NAs dentro del DataFrame, por tanto, es recomendable sustituirlos con **repace_na()**:

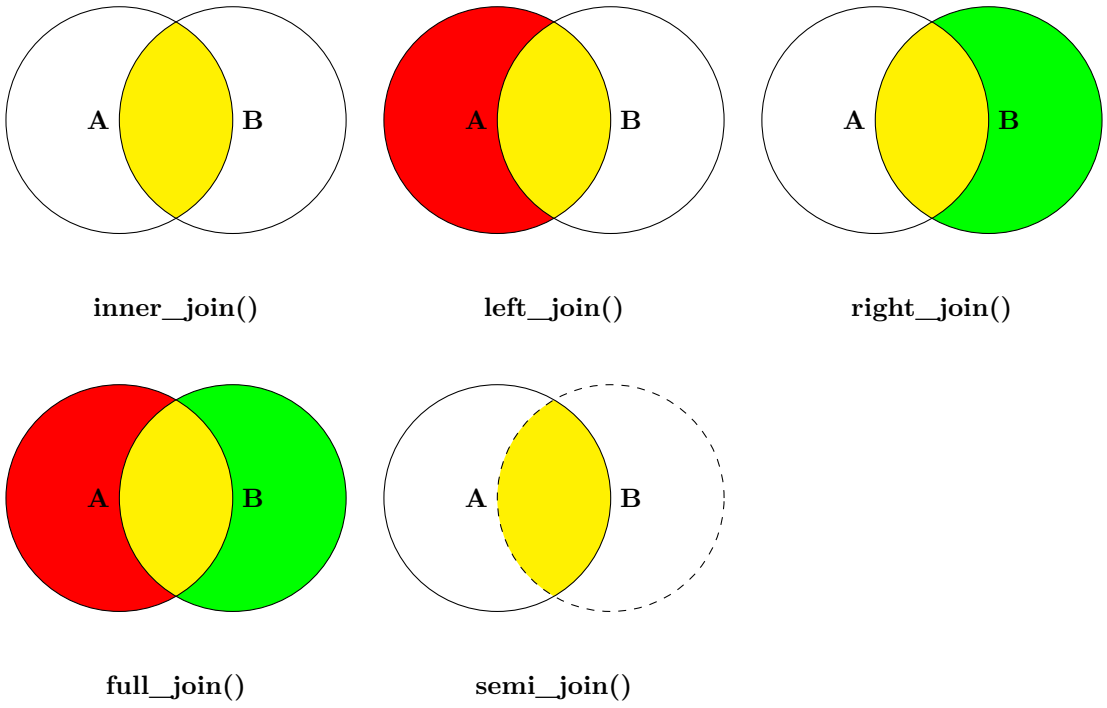
```
batmobile %>%
  full_join(batwing, by = c("part_num", "color_id"), suffix = c("_batmobile","_batwing")) %>%
  replace_na(list(quantity_batmobile = 0,
                  quantity_batwing = 0))
```

```
## # A tibble: 440 x 4
##    part_num color_id quantity_batmobile quantity_batwing
##    <chr>       <dbl>              <dbl>            <dbl>
##  1 3023           72                 62                0
##  2 2780            0                 28               17
##  3 50950           0                 28                2
##  4 3004           71                 26                2
##  5 43093           1                 25                6
##  6 3004            0                 23                4
##  7 3010            0                 21                0
##  8 30363           0                 21                0
##  9 32123b         14                 19                0
## 10 3622            0                 18                2
## # i 430 more rows
```

Las uniones filtro

La función semi_join()

La función **semi_join()** devuelve todas las filas del primer DataFrame que tienen una correspondencia en el segundo DataFrame, pero no agrega las columnas del segundo DataFrame al resultado final.

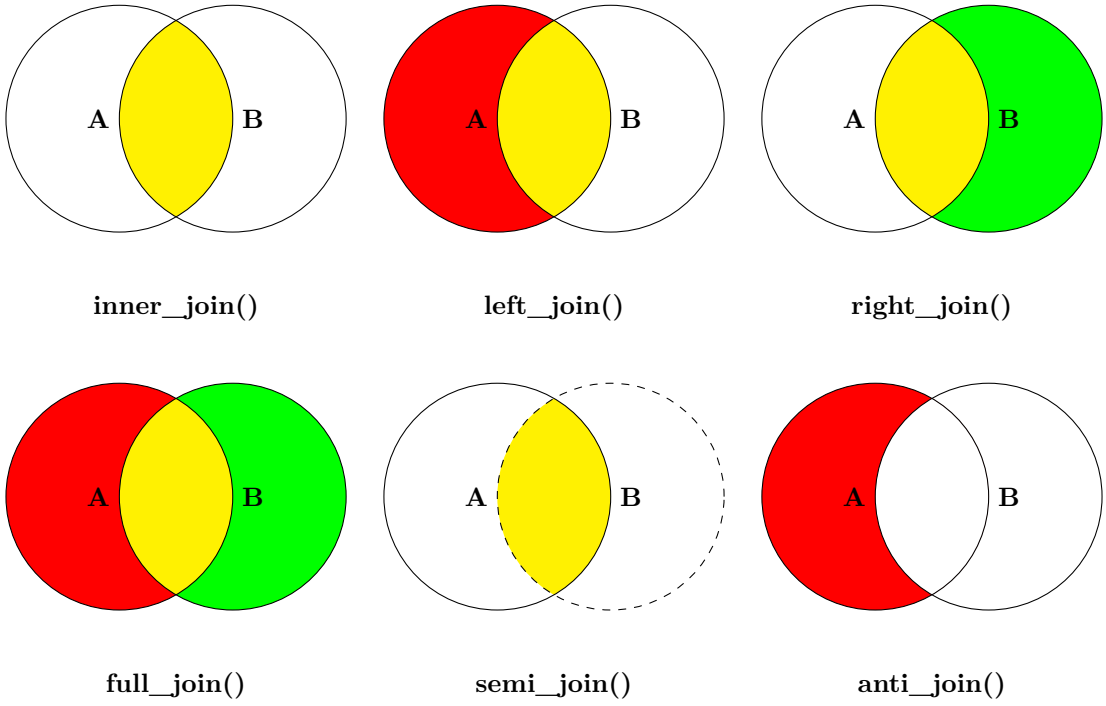



```
batmobile %>%
  semi_join(batwing, by = c("color_id","part_num"))
```

```
## # A tibble: 45 x 3
##    part_num color_id quantity
##    <chr>       <dbl>    <dbl>
##  1 2780            0       28
##  2 50950           0       28
##  3 3004           71       26
##  4 43093           1       25
##  5 3004            0       23
##  6 3622            0       18
##  7 4286            0       16
##  8 3039            0       12
##  9 4274           71       12
## 10 3001            0       11
## # i 35 more rows
```

La función anti_join()

La función **anti_join()** devuelve todas las filas del primer DataFrame que no tienen una correspondencia en el segundo DataFrame, ignorando las filas que tienen coincidencias.




```
batmobile %>%
  anti_join(batwing, by = c("color_id","part_num"))
```

```
## # A tibble: 128 x 3
##    part_num color_id quantity
##    <chr>       <dbl>    <dbl>
##  1 3023           72       62
##  2 3010            0       21
##  3 30363           0       21
##  4 32123b         14       19
##  5 50950         320       18
##  6 6541            0       18
##  7 3040b           0       14
##  8 3298            0       14
##  9 3660            0       14
## 10 42022           0       14
## # i 118 more rows
```