dplyr (continuación)

Entornos de Análisis de Datos: R

Alberto Torres

2021-01-07

Operaciones agrupadas

- La función `group_by()` convierte un data frame en otro agrupado por una o más variables
- En los data frames agrupados todas las operaciones anteriores se realizan "por grupo"
- `ungroup()` elimina la agrupación.

Slice con group_by

- Los indices son relativos al grupo.

```
mpg %>%
  group_by(cyl) %>%
  slice(1:2)
## # A tibble: 8 x 11
## # Groups:   cyl [4]
##   manufacturer model              displ  year   cyl trans      drv     cty   hw
##   <chr>        <chr>              <dbl> <int> <int> <chr>      <chr> <int> <int
## 1 audi         a4                   1.8  1999     4 auto(l5)   f        18    2
## 2 audi         a4                   1.8  1999     4 manual(m5) f        21    2
## 3 volkswagen   jetta                2.5  2008     5 auto(s6)   f        21    2
## 4 volkswagen   jetta                2.5  2008     5 manual(m5) f        21    2
## 5 audi         a4                   2.8  1999     6 auto(l5)   f        16    2
## 6 audi         a4                   2.8  1999     6 manual(m5) f        18    2
## 7 audi         a6 quattro           4.2  2008     8 auto(s6)   4        16    2
## 8 chevrolet    c1500 suburban 2wd   5.3  2008     8 auto(l4)   r        14    2
```

Select con group_by

- `select()` mantiene siempre las variables agrupadas, aunque no se indique explicitamente.

```
data <- mpg %>%
  group_by(cyl) %>%
  select(cty)
## Adding missing grouping variables: `cyl`

glimpse(data)
## Rows: 234
## Columns: 2
## Groups: cyl [4]
## $ cyl <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, 8, 8, 8, 8, 8
## $ cty <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17, 15, 15, 17,
```

arrange con group_by

- `arrange()` ignora la agrupación, a no ser que el parámetro `.by_group` sea `TRUE`

```
mpg %>%
  select(year, cty) %>%
  group_by(year) %>%
  arrange(cty)
## # A tibble: 234 x 2
## # Groups:   year [2]
##     year   cty
##    <int> <int>
##  1  2008     9
##  2  2008     9
##  3  2008     9
##  4  2008     9
##  5  2008     9
##  6  2008    11
##  7  2008    11
##  8  1999    11
##  9  2008    11
## 10  1999    11
## # ... with 224 more rows
```

```
mpg %>%
  select(year, cty) %>%
  group_by(year) %>%
  arrange(cty, .by_group = TRUE)
## # A tibble: 234 x 2
## # Groups:   year [2]
##     year   cty
##    <int> <int>
##  1  1999    11
##  2  1999    11
##  3  1999    11
##  4  1999    11
##  5  1999    11
##  6  1999    11
##  7  1999    11
##  8  1999    11
##  9  1999    11
## 10  1999    11
## # ... with 224 more rows
```

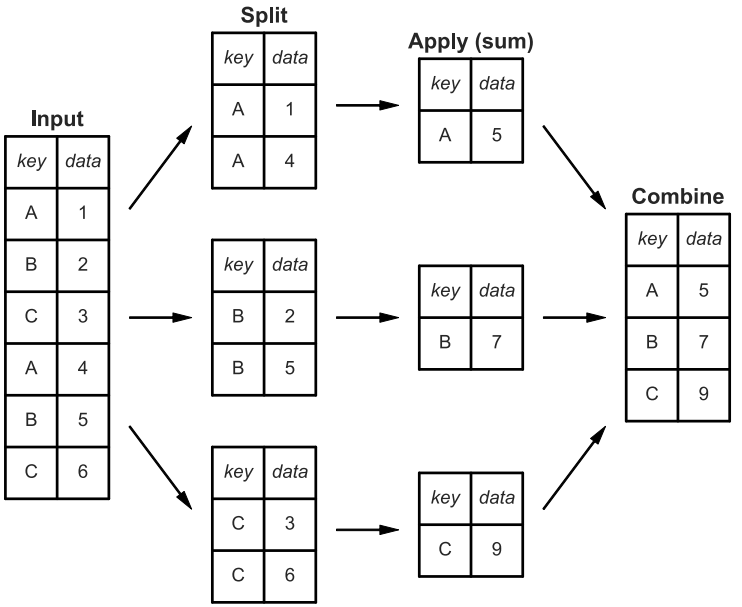
mutate con group_by

Un `mutate()` sobre un data frame agrupado devuelve siempre otro data frame con el mismo número de filas que el original.

```
mpg %>%
  # sin group_by
  mutate(avg_cty = mean(cty)) %>%
  # ungroup no necesario
  select(cyl, cty, avg_cty)
## # A tibble: 234 x 3
##      cyl   cty avg_cty
##    <int> <int>   <dbl>
##  1     4    18    16.9
##  2     4    21    16.9
##  3     4    20    16.9
##  4     4    21    16.9
##  5     6    16    16.9
##  6     6    18    16.9
##  7     6    18    16.9
##  8     4    18    16.9
##  9     4    16    16.9
## 10     4    20    16.9
## # ... with 224 more rows
```

```
mpg %>%
  group_by(cyl) %>%
  mutate(avg_cty = mean(cty)) %>%
  ungroup() %>%
  select(cyl, cty, avg_cty)
## # A tibble: 234 x 3
##      cyl   cty avg_cty
##    <int> <int>   <dbl>
##  1     4    18    21.0
##  2     4    21    21.0
##  3     4    20    21.0
##  4     4    21    21.0
##  5     6    16    16.2
##  6     6    18    16.2
##  7     6    18    16.2
##  8     4    18    21.0
##  9     4    16    21.0
## 10     4    20    21.0
## # ... with 224 more rows
```

Metodología split-apply-combine



Jake VanderPlas. Group-by From Scratch

group_by + summarize

- La metodología anterior se implementa con `group_by + summarize`

```
mpg %>%
   group_by(cyl) %>%
   summarize(avg_cty = mean(cty))
## # A tibble: 4 x 2
##     cyl avg_cty
##   <int>   <dbl>
## 1     4    21.0
## 2     5    20.5
## 3     6    16.2
## 4     8    12.6
```

- Devuelve un dataframe con tantas filas como grupos (valores distintos de la/s variable/s usadas para agrupar).

Agrupar por múltiples columnas

- Podemos agrupar por múltiples columnas

```
mpg %>%
  group_by(drv, year) %>%
  summarize(avg_hwy = mean(hwy))
## # A tibble: 6 x 3
## # Groups:   drv [3]
##   drv    year avg_hwy
##   <chr> <int>   <dbl>
## 1 4      1999    18.8
## 2 4      2008    19.5
## 3 f      1999    27.9
## 4 f      2008    28.4
## 5 r      1999    20.6
## 6 r      2008    21.3
```

- `summarize` elimina un nivel de la agrupación (empezando por la derecha), por lo que hay que tener cuidado si realizamos operaciones posteriores sobre el resultado

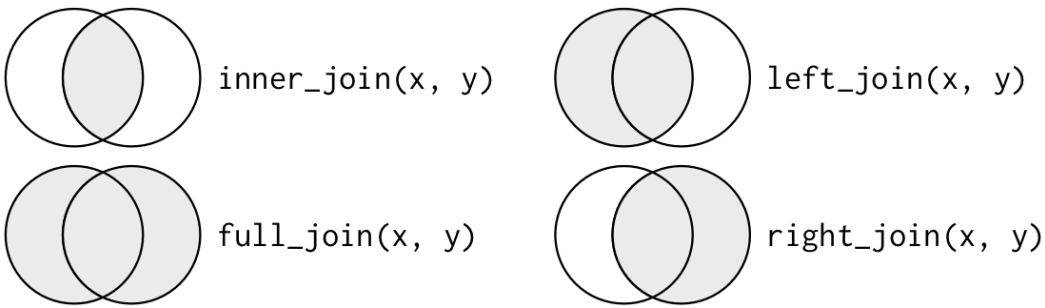
Número de grupos

- El número de grupos es el número de combinaciones posibles de los valores distintos

```
mpg %>%
  summarize(n_year = n_distinct(year),
            n_trans = n_distinct(drv),
            n_comb = n_distinct(year, drv))
## # A tibble: 1 x 3
##   n_year n_trans n_comb
##    <int>   <int>  <int>
## 1      2       3      6
```

joins

- La librería `dplyr` implementa funciones para unir data frames:
  - `inner_join(x,y)` : Devuelve las filas que crucen tant en x como en y.
  - `left_join(x,y)` : Devuelve todas, las filas en x y las que crucen en y (completa con NA)
  - `right_join(x,y)` : Devuelve todas las filas en y y las que crucen en x (completa con NA).
  - `full_join(x,y)` : Devuelve todas las filas de x e y (completa con NA).
  - `semi_join(x,y)` : Devuelve solo las filas de x que crucen con y (pero no y).
  - `anti_join(x,y)` : Devuelve solo las filas de x que NO crucen con y.
- Diagrama de Venn. R for Data Science. Relational data



Equivalencia con SQL

dplyr	SQL
`inner_join(x, y, by = "z")`	`SELECT * FROM x INNER JOIN y USING (z)`
`left_join(x, y, by = "z")`	`SELECT * FROM x LEFT OUTER JOIN y USING (z)`
`right_join(x, y, by = "z")`	`SELECT * FROM x RIGHT OUTER JOIN y USING (z)`
`full_join(x, y, by = "z")`	`SELECT * FROM x FULL OUTER JOIN y USING (z)`

Fuente: R for Data Science. Relational data

Ejemplo (I)

- **t4a**: Número de casos de tuberculosis en Afganistán, Brasil y China durante los años 1999 y 2000
- **t4b**: Población de Afganistán, Brasil y China durante los años 1999 y 2000

```
t4a
## # A tibble: 6 x 3
##   country     year   cases
##   <chr>       <chr>  <int>
## 1 Afghanistan 1999     745
## 2 Brazil      1999   37737
## 3 China       1999  212258
## 4 Afghanistan 2000    2666
## 5 Brazil      2000   80488
## 6 China       2000  213766

t4b
## # A tibble: 6 x 3
##   country     YEAR  populat
##   <chr>       <chr>     <
## 1 Afghanistan 1999    1998
## 2 Brazil      1999   17200
## 3 China       1999   12729
## 4 Afghanistan 2000    2059
## 5 Brazil      2000   17450
## 6 China       2000   12804
```

```
join <- inner_join(t4a, t4b,
                   by=c("year" = "YEAR", "count
join
## # A tibble: 6 x 4
##   country     year   cases population
##   <chr>       <chr>  <int>      <int>
## 1 Afghanistan 1999     745   19987071
## 2 Brazil      1999   37737  172006362
## 3 China       1999  212258 1272915272
## 4 Afghanistan 2000    2666   20595360
## 5 Brazil      2000   80488  174504898
## 6 China       2000  213766 1280428583
```

Ejemplo (II)

```
x <- tibble(x1=c("A","B","C"),x2=1:3)
x
## # A tibble: 3 x 2
##   x1       x2
##   <chr> <int>
## 1 A         1
## 2 B         2
## 3 C         3
y <- tibble(x1=c("B","C","D"),x3=2:4)
y
## # A tibble: 3 x 2
##   x1       x3
##   <chr> <int>
## 1 B         2
## 2 C         3
## 3 D         4
left_join(x,y,by=c("x1"))
## # A tibble: 3 x 3
##   x1       x2    x3
##   <chr> <int> <int>
## 1 A         1    NA
## 2 B         2     2
## 3 C         3     3
```

Ejemplo (III)

```
right_join(x,y,by=c("x1"))
## # A tibble: 3 x 3
##   x1       x2    x3
##   <chr> <int> <int>
## 1 B         2     2
## 2 C         3     3
## 3 D        NA     4
full_join(x,y,by=c("x1"))
## # A tibble: 4 x 3
##   x1       x2    x3
##   <chr> <int> <int>
## 1 A         1    NA
## 2 B         2     2
## 3 C         3     3
## 4 D        NA     4
semi_join(x,y,by=c("x1"))
## # A tibble: 2 x 2
##   x1       x2
##   <chr> <int>
## 1 B         2
## 2 C         3
anti_join(x,y,by=c("x1"))
## # A tibble: 1 x 2
##   x1       x2
##   <chr> <int>
## 1 A         1
```

Operaciones sobre múltiples columnas

- Una misma operación sobre múltiples columnas

```
summarize(mpg,
          avg_hwy = mean(hwy),
          avg_cty = mean(cty),
          avg_displ = mean(displ))
## # A tibble: 1 x 3
##   avg_hwy avg_cty avg_displ
##     <dbl>   <dbl>     <dbl>
## 1    23.4    16.9      3.47
```

- Se puede reescribir de forma más compacta usando la función `across` (dplyr >= 1.0)

```
summarize(mpg, across(c(hwy, cty, displ), mean))
## # A tibble: 1 x 3
##     hwy   cty displ
##   <dbl> <dbl> <dbl>
## 1  23.4  16.9  3.47
```

- Más información: Column-wise operations

Función across

- Recibe dos argumentos:
    1. Selección de columnas (nombre, posición o tipo, al igual que `select`)
    2. Lista de funciones a aplicar sobre las columnas

```
summarize(mpg, across(is.numeric, mean))
## # A tibble: 1 x 5
##   displ  year   cyl   cty   hwy
##   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  3.47 2004.  5.89  16.9  23.4
```

```
summarize(mpg, across(is.character, n_distinct))
## # A tibble: 1 x 6
##   manufacturer model trans   drv    fl class
##          <int> <int> <int> <int> <int> <int>
## 1           15    38    10     3     5     7
```

```
summarize(mpg, across(c(year, cyl), range))
## # A tibble: 2 x 2
##    year   cyl
##   <int> <int>
## 1  1999     4
## 2  2008     8
```

Múltiples funciones

- `across` acepta varias funciones mediante una lista

```
summarize(mpg, across(c(year, cyl), list(min, max)))
## # A tibble: 1 x 4
##   year_1 year_2 cyl_1 cyl_2
##    <int>  <int> <int> <int>
## 1   1999   2008     4     8
```

- Es recomendable nombrar los elementos de la lista, para que las columnas de salida tengan el nombre `"columna_función"`

```
summarize(mpg, across(is.numeric, list(media = mean, desv = sd)))
## # A tibble: 1 x 10
##   displ_media displ_desv year_media year_desv cyl_media cyl_desv cty_media
##         <dbl>      <dbl>      <dbl>     <dbl>     <dbl>    <dbl>     <dbl>
## 1        3.47       1.29      2004.      4.51      5.89     1.61      16.9
```

- El formato anterior se puede cambiar modificando el parámetro opcional `.names`

```
summarize(mpg, across(c(year, cyl), list(menor = min, mayor = max),
                      .names = "{fn}-{col}"))
## # A tibble: 1 x 4
##   `menor-year` `mayor-year` `menor-cyl` `mayor-cyl`
##          <int>        <int>       <int>       <int>
## 1         1999         2008           4           8
```

Group_by + across

- `across` no aplica sobre las variables que se usan para agrupar

```
iris %>%
  group_by(Species) %>%
  summarize(across(is.numeric, mean))
## # A tibble: 3 x 5
##   Species    Sepal.Length Sepal.Width Petal.Length Petal.Width
##   <fct>             <dbl>       <dbl>        <dbl>       <dbl>
## 1 setosa             5.01        3.43         1.46       0.246
## 2 versicolor         5.94        2.77         4.26       1.33
## 3 virginica          6.59        2.97         5.55       2.03
```

Errores comunes

- `summarize` realiza las operaciones de izquierda a derecha, hay que tener cuidado cuando se combina `across` con otras operaciones:

```
iris %>%
  group_by(Species) %>%
  summarize(n = n(), across(is.numeric, sd))
## # A tibble: 3 x 6
##   Species        n Sepal.Length Sepal.Width Petal.Length Petal.Width
##   <fct>      <dbl>        <dbl>       <dbl>        <dbl>       <dbl>
## 1 setosa        NA        0.352       0.379        0.174       0.105
## 2 versicolor    NA        0.516       0.314        0.470       0.198
## 3 virginica     NA        0.636       0.322        0.552       0.275
```

```
iris %>%
  group_by(Species) %>%
  summarize(across(is.numeric, sd), n = n())
## # A tibble: 3 x 6
##   Species    Sepal.Length Sepal.Width Petal.Length Petal.Width     n
##   <fct>             <dbl>       <dbl>        <dbl>       <dbl> <int>
## 1 setosa            0.352       0.379        0.174       0.105    50
## 2 versicolor        0.516       0.314        0.470       0.198    50
## 3 virginica         0.636       0.322        0.552       0.275    50
```

Otras funciones de dplyr + across

`across` se puede usar también en otras funciones. Ejemplos:

- Seleccionar filas donde las columnas Ozone **y** Solar.R son `NA`

```
filter(airquality, across(c(Ozone, Solar.R), is.na))
##   Ozone Solar.R Wind Temp Month Day
## 1    NA      NA 14.3   56     5   5
## 2    NA      NA  8.0   57     5  27
```

```
# version sin across, equivalente
filter(airquality, is.na(Ozone), is.na(Solar.R))
##   Ozone Solar.R Wind Temp Month Day
## 1    NA      NA 14.3   56     5   5
## 2    NA      NA  8.0   57     5  27
```

- Calcular el logaritmo de todas las columnas numéricas

```
iris %>%
  mutate(across(is.numeric, log)) %>%
  head()
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1     1.629241    1.252763    0.3364722  -1.6094379  setosa
## 2     1.589235    1.098612    0.3364722  -1.6094379  setosa
## 3     1.547563    1.163151    0.2623643  -1.6094379  setosa
## 4     1.526056    1.131402    0.4054651  -1.6094379  setosa
## 5     1.609438    1.280934    0.3364722  -1.6094379  setosa
## 6     1.686399    1.360977    0.5306283  -0.9162907  setosa
```

Operaciones de conjuntos

- `dplyr` implementa la lógica de operaciones con conjuntos sobre tibbles
  - `intersect(x,y)` : Filas que aparecen tanto en x como en y
  - `union(x,y)` : Filas que aparecen en x, en y, o en ambos
  - `setdiff(x,y)` : Filas que aparecen en x, pero no en y

```
x <- tibble(
  x1=c("A","B","C"),
  x2=1:3
)
y <- tibble(
  x1=c("B","C","D"),
  x2=2:4
)
dplyr::intersect(x,y)
dplyr::union(x,y)
dplyr::setdiff(x,y)
```

Añadir filas

- `dplyr` implementa las funciones `bind_rows` y `bind_cols` para añadir filas o columnas a un tibble, respectivamente
- En `bind_rows` las columnas se combinan por nombre y las columnas que no están en alguno de los dataframes se rellenan con NAs

```
bind_rows(
  c(a = 1, b = 2),
  tibble(saludo="hola", a = 3:4, b = 5:6),
  c(a = 7, b = 8)
)
## # A tibble: 4 x 3
##       a     b saludo
##   <dbl> <dbl> <chr>
## 1     1     2 <NA>
## 2     3     5 hola
## 3     4     6 hola
## 4     7     8 <NA>
```

Añadir columnas

- En `bind_cols` se unen las subtablas por posición -> todos los dataframes deben tener el mismo número de filas
  - Para unir por valores, usar `join`.

```
# Both have to be tibbles
bind_cols(
  tibble(a = 3:4, b = c("a", "b")),
  tibble(logical = c(T, F))
)
## # A tibble: 2 x 3
##       a b     logical
##   <int> <chr> <lgl>
## 1     3 a     TRUE
## 2     4 b     FALSE
```