

dplyr (continuación)

Entornos de Análisis de Datos: R

Alberto Torres

2020-01-14

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 índices 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 hwy fl c
<chr> <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <
1 audi a4 1.8 1999 4 auto(l~ f 18 29 p c
2 audi a4 1.8 1999 4 manual~ f 21 29 p c
3 volkswagen jetta 2.5 2008 5 auto(s~ f 21 29 r c
4 volkswagen jetta 2.5 2008 5 manual~ f 21 29 r c
5 audi a4 2.8 1999 6 auto(l~ f 16 26 p c
6 audi a4 2.8 1999 6 manual~ f 18 26 p c
7 audi a6 quattro 4.2 2008 8 auto(s~ 4 16 23 p n
8 chevrolet c1500 subur~ 5.3 2008 8 auto(l~ r 14 20 r s

Select con group_by

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

```
dim(mpg)
data <- mpg %>%
  group_by(cyl) %>%
  select(cty)
dim(data)
```

```
data <- mpg %>%
  group_by(cyl) %>%
  select(cty)
glimpse(data)
## Observations: 234
## Variables: 2
## Groups: cyl [4]
## $ cyl <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 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()` ordena por la(s) variable(s) especificadas como parámetros.

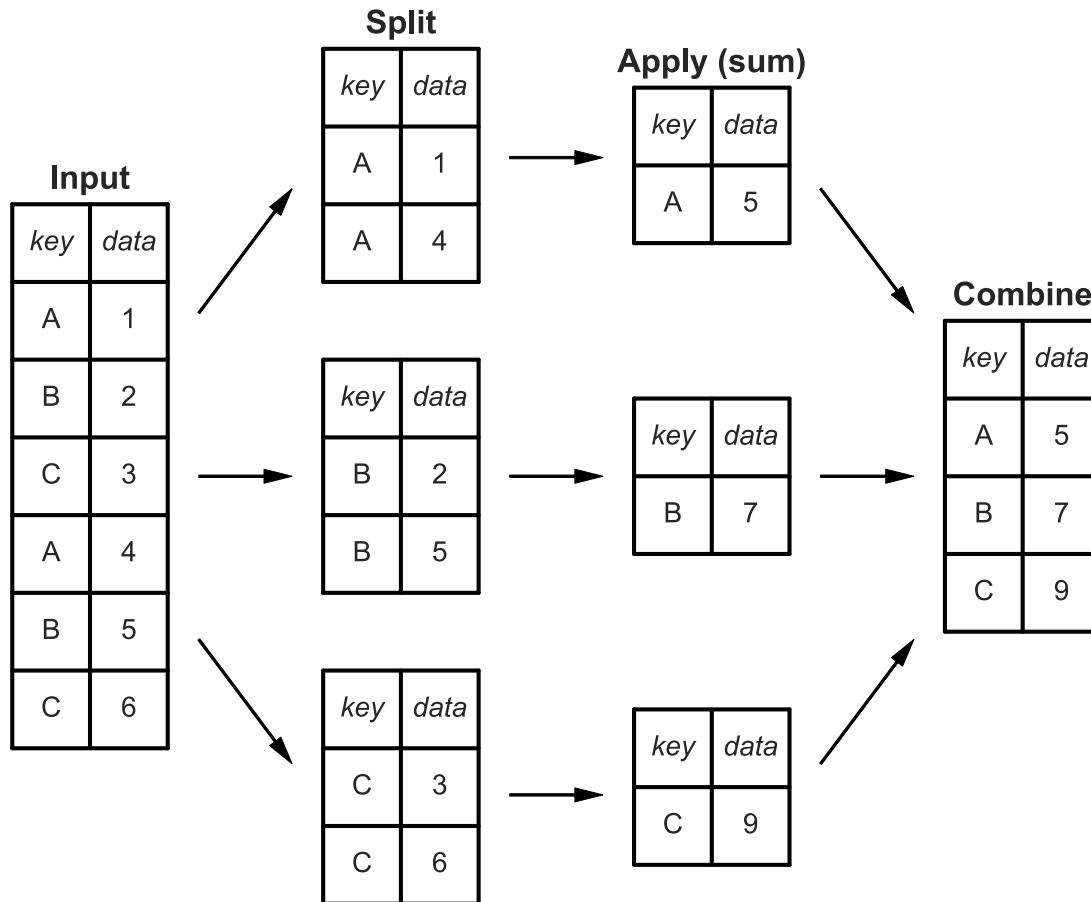
```
data <- mpg %>%
  group_by(cyl) %>%
  arrange(manufacturer)
glimpse(data)
## Observations: 234
## Variables: 11
## Groups: cyl [4]
## $ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi", "a
## $ model <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro", "a
## $ displ <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, 2.8
## $ year <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 2008
## $ cyl <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, 8, 8
## $ trans <chr> "auto(l5)", "manual(m5)", "manual(m6)", "auto(av)", "auto(
## $ drv <chr> "f", "f", "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", "4"
## $ cty <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17, 15
## $ hwy <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25, 25
## $ fl <chr> "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p"
## $ class <chr> "compact", "compact", "compact", "compact", "compact", "cc
```

summarize con group_by

Un `summarize()` sobre un data frame agrupado devuelve otro con tantas filas como grupos (valores distintos de la/s variable/s usadas para agrupar).

```
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
```

Metodología split-apply-combine



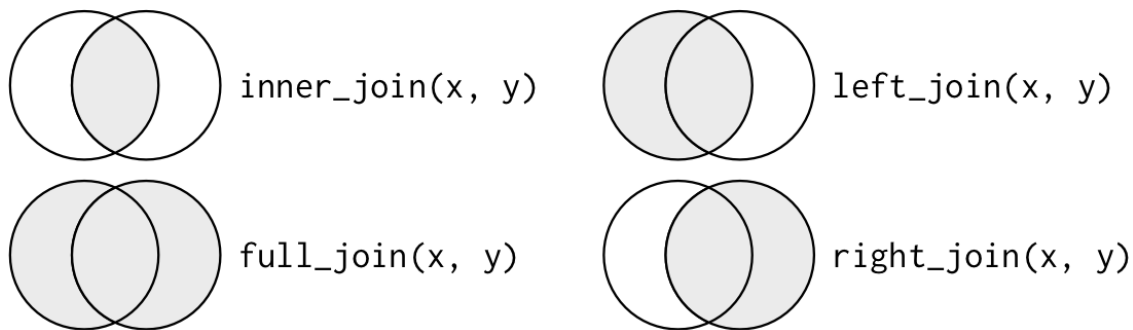
mutate con group_by

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

```
data <- mpg %>%
  group_by(cyl) %>%
  mutate(avg_cty = mean(cty))
glimpse(data)
## Observations: 234
## Variables: 12
## Groups: cyl [4]
## $ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi", "a
## $ model <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro", "a
## $ displ <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, 2.8
## $ year <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 2008
## $ cyl <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, 8, 8
## $ trans <chr> "auto(l5)", "manual(m5)", "manual(m6)", "auto(av)", "auto(
## $ drv <chr> "f", "f", "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", "4"
## $ cty <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17, 15
## $ hwy <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25, 25
## $ fl <chr> "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p"
## $ class <chr> "compact", "compact", "compact", "compact", "compact", "cc
## $ avg_cty <dbl> 21.01235, 21.01235, 21.01235, 21.01235, 16.21519, 16.21519
```


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]



Equivalencia con SQL

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

[R for Data Science]

Ejemplo

```
t4a <- gather(table4a, key = "year", value = "cases", num_range("", 1999:2000))
head(t4a,4)
## # A tibble: 4 x 3
##   country    year  cases
##   <chr>    <chr> <int>
## 1 Afghanistan 1999     745
## 2 Brazil      1999   37737
## 3 China       1999  212258
## 4 Afghanistan 2000    2666
t4b <- gather(table4b, key = "YEAR", value = "population", `1999`:`2000`)
head(t4b,4)
## # A tibble: 4 x 3
##   country    YEAR  population
##   <chr>    <chr>    <int>
## 1 Afghanistan 1999  19987071
## 2 Brazil      1999  172006362
## 3 China       1999  1272915272
## 4 Afghanistan 2000   20595360
```

Ejemplo (cont.)

```
inner_join(t4a, t4b, by=c("year" = "YEAR", "country"))  
## # 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
```

Operaciones de conjuntos con dplyr

- `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 y/o columnas en dplyr

- `dplyr` implementa las funciones `bind_rows` y `bind_cols` para añadir filas o columnas a un tibble, respectivamente.
- Las funciones de `dplyr` son **más eficientes** que las funciones `rbind` y `cbind` de R base.
- 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="ho1a", 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 ho1a  
## 3     4     6 ho1a  
## 4     7     8 <NA>
```

Añadir filas y/o columnas en dplyr (cont.)

- 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
```

Operar en múltiples columnas

- dplyr tiene **variantes** de sus funciones principales que operan sobre múltiples columnas
- La selección de columnas puede ser:
 - Todas, funciones que terminan en `_all`
 - Con un predicado, funciones que terminan en `_if`
 - Vector con nombres, posiciones o función `vars()`


```

summarize_all(mpg, funs(sum(is.na(.))))
## # A tibble: 1 x 11
##   manufacturer model displ  year   cyl trans  drv   cty   hwy   fl class
##   <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
## 1           0     0     0     0     0     0     0     0     0     0     0

```

```

mutate_if(mpg, is.numeric, log)
## # A tibble: 234 x 11
##   manufacturer model displ  year   cyl trans  drv   cty   hwy fl class
##   <chr> <chr> <dbl> <dbl> <dbl> <chr> <chr> <dbl> <dbl> <chr> <chr>
## 1 audi a4 0.588 7.60 1.39 auto(15) f 2.89 3.37 p c
## 2 audi a4 0.588 7.60 1.39 manual(m~ f 3.04 3.37 p c
## 3 audi a4 0.693 7.60 1.39 manual(m~ f 3.00 3.43 p c
## 4 audi a4 0.693 7.60 1.39 auto(av) f 3.04 3.40 p c
## 5 audi a4 1.03 7.60 1.79 auto(15) f 2.77 3.26 p c
## 6 audi a4 1.03 7.60 1.79 manual(m~ f 2.89 3.26 p c
## 7 audi a4 1.13 7.60 1.79 auto(av) f 2.89 3.30 p c
## 8 audi a4 quatt~ 0.588 7.60 1.39 manual(m~ 4 2.89 3.26 p c
## 9 audi a4 quatt~ 0.588 7.60 1.39 auto(15) 4 2.77 3.22 p c
## 10 audi a4 quatt~ 0.693 7.60 1.39 manual(m~ 4 3.00 3.33 p c
## # ... with 224 more rows

```