

dplyr

Fundamentos lenguajes: R

Alberto Torres Barrán y Irene Rodríguez Luján

2019-07-08

dplyr

Introducción

- Implementa una gramática para realizar operaciones básicas con data frames
- Muy eficiente
- Operaciones principales: `slice`, `filter`, `select`, `arrange`, `mutate` y `summarize`
- Estas operaciones se pueden componer para realizar otras más complejas

slice

Selecciona filas por su posición

```
slice(mpg, 1:5)
```

```
## # A tibble: 5 x 11
##   manufacturer model displ  year   cyl trans  drv    cty   hwy fl    class
##   <chr>         <chr> <dbl> <int> <int> <chr>  <chr> <int> <int> <chr>  <chr>
## 1 audi         a4      1.8  1999     4 auto(...) f      18    29 p    comp...
## 2 audi         a4      1.8  1999     4 manua... f      21    29 p    comp...
## 3 audi         a4      2    2008     4 manua... f      20    31 p    comp...
## 4 audi         a4      2    2008     4 auto(...) f      21    30 p    comp...
## 5 audi         a4      2.8  1999     6 auto(...) f      16    26 p    comp...
```

filter

Selecciona filas por condición

```
filter(mpg, model == "a4")
```

```
## # A tibble: 7 x 11
##   manufacturer model displ  year   cyl trans  drv    cty   hwy fl  class
##   <chr>          <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
## 1 audi          a4     1.8  1999     4 auto(...) f      18    29 p  comp...
## 2 audi          a4     1.8  1999     4 manua... f      21    29 p  comp...
## 3 audi          a4     2    2008     4 manua... f      20    31 p  comp...
## 4 audi          a4     2    2008     4 auto(...) f      21    30 p  comp...
## 5 audi          a4     2.8  1999     6 auto(...) f      16    26 p  comp...
## 6 audi          a4     2.8  1999     6 manua... f      18    26 p  comp...
## 7 audi          a4     3.1  2008     6 auto(...) f      18    27 p  comp...
```

Operadores lógicos

- R implementa todos los operadores relacionales habituales `>`, `<`, `>=`, `<=`, `==`, `!=`
- Los operadores lógicos son la negación `!`, and `&` y or `|`
- El resultado de todas estas operaciones son valores lógicos `TRUE` (T) o `FALSE` (F)

filter (cont.)

- Se pueden combinar multiples condiciones separadas por `,` (and lógico)

```
filter(mpg, model == "a4", cyl >= 5)
```

- También se puede usar explicitamente el operador

```
filter(mpg, model == "a4" & cyl >= 5)
```

- En el caso del or lógico es obligatorio el uso del operador

```
filter(mpg, model == "a4" | model == "mustang")
```

select

Seleccionar variables (columnas) de un data frame

```
select(mpg, model, displ, cyl)
```

```
## # A tibble: 234 x 3
##   model      displ    cyl
##   <chr>      <dbl> <int>
## 1 a4          1.8     4
## 2 a4          1.8     4
## 3 a4           2     4
## 4 a4           2     4
## 5 a4          2.8     6
## 6 a4          2.8     6
## 7 a4          3.1     6
## 8 a4 quattro  1.8     4
## 9 a4 quattro  1.8     4
## 10 a4 quattro  2      4
## # ... with 224 more rows
```

Ignorar variables

Con un `-` se ignoran variables

```
select(mpg, -manufacturer)
```

```
## # A tibble: 234 x 10
##   model      displ  year   cyl trans      drv    cty   hwy fl      class
##   <chr>      <dbl> <int> <int> <chr>    <chr> <int> <int> <chr>  <chr>
## 1 a4          1.8  1999     4 auto(l5)  f       18    29 p    compact
## 2 a4          1.8  1999     4 manual(m5) f       21    29 p    compact
## 3 a4          2    2008     4 manual(m6) f       20    31 p    compact
## 4 a4          2    2008     4 auto(av)   f       21    30 p    compact
## 5 a4          2.8  1999     6 auto(l5)  f       16    26 p    compact
## 6 a4          2.8  1999     6 manual(m5) f       18    26 p    compact
## 7 a4          3.1  2008     6 auto(av)   f       18    27 p    compact
## 8 a4 quattro  1.8  1999     4 manual(m5) 4       18    26 p    compact
## 9 a4 quattro  1.8  1999     4 auto(l5)   4       16    25 p    compact
## 10 a4 quattro  2    2008     4 manual(m6) 4       20    28 p    compact
## # ... with 224 more rows
```


Rangos

Puesto que las variables están ordenadas, se puede seleccionar un rango con :

```
select(mpg, model:trans)
```

```
## # A tibble: 234 x 5
##   model      displ  year   cyl trans
##   <chr>      <dbl> <int> <int> <chr>
## 1 a4          1.8  1999     4 auto(l5)
## 2 a4          1.8  1999     4 manual(m5)
## 3 a4          2    2008     4 manual(m6)
## 4 a4          2    2008     4 auto(av)
## 5 a4          2.8  1999     6 auto(l5)
## 6 a4          2.8  1999     6 manual(m5)
## 7 a4          3.1  2008     6 auto(av)
## 8 a4 quattro  1.8  1999     4 manual(m5)
## 9 a4 quattro  1.8  1999     4 auto(l5)
## 10 a4 quattro  2    2008     4 manual(m6)
## # ... with 224 more rows
```

Funciones auxiliares

Las siguientes funciones se pueden usar dentro de `select()`

- `starts_with()` : empiezan con un prefijo
- `ends_with()` : terminan con un sufijo
- `contains()` : contienen una string
- `matches()` : concuerdan con una expresión regular
- `num_range()` : rango numérico como "X01", "X02", "X03"

arrange

Ordena las filas de un data frame

```
arrange(mpg, desc(year), cyl)
```

```
## # A tibble: 234 x 11
##   manufacturer model displ  year   cyl trans drv   cty   hwy fl   class
##   <chr>         <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
## 1 audi         a4      2   2008     4 manu... f     20    31 p    comp...
## 2 audi         a4      2   2008     4 auto... f     21    30 p    comp...
## 3 audi         a4 q...  2   2008     4 manu... 4     20    28 p    comp...
## 4 audi         a4 q...  2   2008     4 auto... 4     19    27 p    comp...
## 5 chevrolet    mali... 2.4  2008     4 auto... f     22    30 r    mids...
## 6 honda        civic  1.8  2008     4 manu... f     26    34 r    subc...
## 7 honda        civic  1.8  2008     4 auto... f     25    36 r    subc...
## 8 honda        civic  1.8  2008     4 auto... f     24    36 c    subc...
## 9 honda        civic  2    2008     4 manu... f     21    29 p    subc...
## 10 hyundai     sona... 2.4  2008     4 auto... f     21    30 r    mids...
## # ... with 224 more rows
```

mutate

Añade nuevas variables (columnas) al data frame como combinación de las ya existentes

```
mutate(mpg, avg_mpg = (cty+hwy)/2)
```

```
## # A tibble: 234 x 12
##   manufacturer model displ  year   cyl trans drv   cty   hwy fl   class
##   <chr>          <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
## 1 audi          a4      1.8  1999     4 auto... f     18    29 p    comp...
## 2 audi          a4      1.8  1999     4 manu... f     21    29 p    comp...
## 3 audi          a4      2    2008     4 manu... f     20    31 p    comp...
## 4 audi          a4      2    2008     4 auto... f     21    30 p    comp...
## 5 audi          a4      2.8  1999     6 auto... f     16    26 p    comp...
## 6 audi          a4      2.8  1999     6 manu... f     18    26 p    comp...
## 7 audi          a4      3.1  2008     6 auto... f     18    27 p    comp...
## 8 audi          a4 q...  1.8  1999     4 manu... 4     18    26 p    comp...
## 9 audi          a4 q...  1.8  1999     4 auto... 4     16    25 p    comp...
## 10 audi         a4 q...  2    2008     4 manu... 4     20    28 p    comp...
## # ... with 224 more rows, and 1 more variable: avg_mpg <dbl>
```

Operadores y funciones aritméticas

- R implementa los operadores aritméticos habituales
 - suma `+`
 - resta `-`
 - multiplicación `*`
 - división `/`
 - exponenciación `^`
 - división entera `%/%`
 - módulo (resto) `%%`
- También las funciones aritméticas comunes: `log()`, `exp()`, `sin()`, `cos()`, `tan()`, `cumsum()`, `cumprod()`, `abs()`, `sqrt()`, `round()`, `ceiling()`, `floor()`, `trunc()`, ...
- Operan sobre vectores (columnas de un data frame) elemento a elemento

summarize

Colapsa el data frame a una única fila

```
summarize(mpg, max_cyl = max(cyl), avg_cty = mean(cty), min_year = min(year))
```

```
## # A tibble: 1 x 3  
##   max_cyl avg_cty min_year  
##   <int>   <dbl>   <int>  
## 1      8    16.9    1999
```

Funciones de agregación

- Las funciones más comunes para usar dentro de `summarize()` son:
 - Aritméticas: `prod()`, `sum()`
 - Centralidad: `mean()`, `median()`
 - Dispersión: `sd()`, `var()`, `mad()`
 - Rango: `max()`, `min()`, `quantile()`
 - Posición: `first()`, `last()`, `nth()`
 - Lógicas: `any()`, `all()`
 - *Conteo*: `n()`, `n_distinct()` (solo se pueden usar dentro de `summarize()`)
- Todas reducen un vector de números a un único resultado

Concatenación de funciones

- Todas las funciones de `dplyr` toman como primer argumento un data frame y devuelven otro data frame
- Se pueden aplicar de manera consecutiva:

```
arrange(select(filter(mpg, model == "a4"), model, year), year)

arrange(
  select(
    filter(mpg, model == "a4"),
    model, year
  ),
  year
)
```


Concatenación de funciones (cont.)

-Otra opción:

```
df1 <- filter(mpg, model == "a4")  
df2 <- select(df1, model, year)  
df3 <- arrange(df2, year)
```

- Habitualmente no nos interesan los valores intermedios, solo el resultado final

Operador "tubería" (*pipe*)

- La sintaxis es `%>%` y permite reescribir el código anterior como

```
mpg %>%  
  filter(model == "a4") %>%  
  select(model, year) %>%  
  arrange(year)
```

- En general el código `df %>% foo()` es equivalente a `foo(df)`
- Esto permite concatenar funciones sin almacenar resultados intermedios y siguiendo el orden lógico

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

Operaciones agrupadas (cont.)

- `slice()` los índices son relativos al grupo
- `select()` mantiene siempre las variables agrupadas, aunque no se indique explícitamente
- `arrange()` ordena en primer lugar por las variables agrupadas

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
```

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 %>%  
  group_by(cyl) %>%  
  mutate(avg_cty = mean(cty))
```

```
## # A tibble: 234 x 12  
## # Groups:   cyl [4]  
##   manufacturer model displ year   cyl trans  drv    cty   hwy fl   class  
##   <chr>          <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>  
## 1 audi         a4      1.8  1999     4 auto... f      18    29 p    comp...  
## 2 audi         a4      1.8  1999     4 manu... f      21    29 p    comp...  
## 3 audi         a4      2    2008     4 manu... f      20    31 p    comp...  
## 4 audi         a4      2    2008     4 auto... f      21    30 p    comp...  
## 5 audi         a4      2.8  1999     6 auto... f      16    26 p    comp...  
## 6 audi         a4      2.8  1999     6 manu... f      18    26 p    comp...  
## 7 audi         a4      3.1  2008     6 auto... f      18    27 p    comp...  
## 8 audi         a4 q...  1.8  1999     4 manu... 4      18    26 p    comp...  
## 9 audi         a4 q...  1.8  1999     4 auto... 4      16    25 p    comp...  
## 10 audi        a4 q...  2    2008     4 manu... 4      20    28 p    comp...  
## # ... with 224 more rows, and 1 more variable: avg_cty <dbl>
```

joins

- La librería `dplyr` implementa funciones para unir data frames: `inner_join()`, `left_join()`, `right_join()` y `full_join()`
- Diagrama de Venn [R for Data Science]

![Diagrama de Venn](join-venn.png)

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))
t4b <- gather(table4b, key = "YEAR", value = "population", `1999`:`2000`)
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
```

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... f      2.89  3.37 p    comp...
## 2 audi          a4    0.588 7.60  1.39 manu... f      3.04  3.37 p    comp...
## 3 audi          a4    0.693 7.60  1.39 manu... f      3.00  3.43 p    comp...
## 4 audi          a4    0.693 7.60  1.39 auto... f      3.04  3.40 p    comp...
## 5 audi          a4    1.03  7.60  1.79 auto... f      2.77  3.26 p    comp...
## 6 audi          a4    1.03  7.60  1.79 manu... f      2.89  3.26 p    comp...
## 7 audi          a4    1.13  7.60  1.79 auto... f      2.89  3.30 p    comp...
## 8 audi          a4 q... 0.588 7.60  1.39 manu... 4      2.89  3.26 p    comp...
## 9 audi          a4 q... 0.588 7.60  1.39 auto... 4      2.77  3.22 p    comp...
## 10 audi         a4 q... 0.693 7.60  1.39 manu... 4      3.00  3.33 p    comp...
## # ... with 224 more rows
```