Manipulación de Datos con dplyr

Rubén Sierra Serrano

2024-03-23

Índice

Transformación de datos

A lo largo del siguiente documento, se trabajará con el conjunto de datos *counties*, el cual corresponde a un censo de los Estados Unidos del año 2015.

La función **glimpse()** del paquete dplyr permite visualizar los primeros valores de todas las variables de un DataFrame, así como el tipo de dato de cada una.

```
library(dplyr)
glimpse(counties)
```

```
## Rows: 3,138
## Columns: 40
## $ census_id          <chr> "1001", "1003", "1005", "1007", "1009", "1011", "10~
## $ state              <chr> "Alabama", "Alabama", "Alabama", "Alabama", "Alabam~
## $ county             <chr> "Autauga", "Baldwin", "Barbour", "Bibb", "Blount", ~
## $ region             <chr> "South", "South", "South", "South", "South", "South~
## $ metro              <chr> "Metro", "Metro", "Nonmetro", "Metro", "Metro", "No~
## $ population         <dbl> 55221, 195121, 26932, 22604, 57710, 10678, 20354, 1~
## $ men                <dbl> 26745, 95314, 14497, 12073, 28512, 5660, 9502, 5627~
## $ women              <dbl> 28476, 99807, 12435, 10531, 29198, 5018, 10852, 603~
## $ hispanic           <dbl> 2.6, 4.5, 4.6, 2.2, 8.6, 4.4, 1.2, 3.5, 0.4, 1.5, 7~
## $ white              <dbl> 75.8, 83.1, 46.2, 74.5, 87.9, 22.2, 53.3, 73.0, 57.~
## $ black              <dbl> 18.5, 9.5, 46.7, 21.4, 1.5, 70.7, 43.8, 20.3, 40.3,~
## $ native             <dbl> 0.4, 0.6, 0.2, 0.4, 0.3, 1.2, 0.1, 0.2, 0.2, 0.6, 0~
## $ asian              <dbl> 1.0, 0.7, 0.4, 0.1, 0.1, 0.2, 0.4, 0.9, 0.8, 0.3, 0~
## $ pacific            <dbl> 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0~
## $ citizens           <dbl> 40725, 147695, 20714, 17495, 42345, 8057, 15581, 88~
## $ income             <dbl> 51281, 50254, 32964, 38678, 45813, 31938, 32229, 41~
## $ income_err         <dbl> 2391, 1263, 2973, 3995, 3141, 5884, 1793, 925, 2949~
## $ income_per_cap     <dbl> 24974, 27317, 16824, 18431, 20532, 17580, 18390, 21~
## $ income_per_cap_err <dbl> 1080, 711, 798, 1618, 708, 2055, 714, 489, 1366, 15~
## $ poverty            <dbl> 12.9, 13.4, 26.7, 16.8, 16.7, 24.6, 25.4, 20.5, 21.~
## $ child_poverty      <dbl> 18.6, 19.2, 45.3, 27.9, 27.2, 38.4, 39.2, 31.6, 37.~
## $ professional       <dbl> 33.2, 33.1, 26.8, 21.5, 28.5, 18.8, 27.5, 27.3, 23.~
## $ service            <dbl> 17.0, 17.7, 16.1, 17.9, 14.1, 15.0, 16.6, 17.7, 14.~
## $ office             <dbl> 24.2, 27.1, 23.1, 17.8, 23.9, 19.7, 21.9, 24.2, 26.~
## $ construction       <dbl> 8.6, 10.8, 10.8, 19.0, 13.5, 20.1, 10.3, 10.5, 11.5~
## $ production         <dbl> 17.1, 11.2, 23.1, 23.7, 19.9, 26.4, 23.7, 20.4, 24.~
## $ drive              <dbl> 87.5, 84.7, 83.8, 83.2, 84.9, 74.9, 84.5, 85.3, 85.~
## $ carpool            <dbl> 8.8, 8.8, 10.9, 13.5, 11.2, 14.9, 12.4, 9.4, 11.9, ~
## $ transit            <dbl> 0.1, 0.1, 0.4, 0.5, 0.4, 0.7, 0.0, 0.2, 0.2, 0.2, 0~
## $ walk               <dbl> 0.5, 1.0, 1.8, 0.6, 0.9, 5.0, 0.8, 1.2, 0.3, 0.6, 1~
## $ other_transp       <dbl> 1.3, 1.4, 1.5, 1.5, 0.4, 1.7, 0.6, 1.2, 0.4, 0.7, 1~
## $ work_at_home       <dbl> 1.8, 3.9, 1.6, 0.7, 2.3, 2.8, 1.7, 2.7, 2.1, 2.5, 1~
## $ mean_commute       <dbl> 26.5, 26.4, 24.1, 28.8, 34.9, 27.5, 24.6, 24.1, 25.~
## $ employed           <dbl> 23986, 85953, 8597, 8294, 22189, 3865, 7813, 47401,~
## $ private_work       <dbl> 73.6, 81.5, 71.8, 76.8, 82.0, 79.5, 77.4, 74.1, 85.~
## $ public_work        <dbl> 20.9, 12.3, 20.8, 16.1, 13.5, 15.1, 16.2, 20.8, 12.~
## $ self_employed      <dbl> 5.5, 5.8, 7.3, 6.7, 4.2, 5.4, 6.2, 5.0, 2.8, 7.9, 4~
## $ family_work        <dbl> 0.0, 0.4, 0.1, 0.4, 0.4, 0.0, 0.2, 0.1, 0.0, 0.5, 0~
## $ unemployment       <dbl> 7.6, 7.5, 17.6, 8.3, 7.7, 18.0, 10.9, 12.3, 8.9, 7.~
## $ land_area          <dbl> 594.44, 1589.78, 884.88, 622.58, 644.78, 622.81, 77~
```

Para seleccionar variables dentro del DataFrame se puede emplear la función **select()**.

```
counties %>%
  select(state, county, income, poverty)
```

```
## # A tibble: 3,138 x 4
##    state   county   income poverty
##    <chr>   <chr>     <dbl>   <dbl>
##  1 Alabama Autauga   51281    12.9
##  2 Alabama Baldwin   50254    13.4
##  3 Alabama Barbour   32964    26.7
##  4 Alabama Bibb      38678    16.8
##  5 Alabama Blount    45813    16.7
##  6 Alabama Bullock   31938    24.6
##  7 Alabama Butler    32229    25.4
##  8 Alabama Calhoun   41703    20.5
##  9 Alabama Chambers  34177    21.6
## 10 Alabama Cherokee  36296    19.2
## # i 3,128 more rows
```

La función count()

La función **count()** cuenta el número de observaciones de un DataFrame.

```
counties %>%
  count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1  3138
```

Se puede especificar un atributo dentro del DataFrame para conocer el número de observaciones que pertenecen a cada valor de la variable.

```
counties %>%
  count(state)
```

```
## # A tibble: 50 x 2
##    state           n
##    <chr>       <int>
##  1 Alabama        67
##  2 Alaska         28
##  3 Arizona        15
##  4 Arkansas       75
##  5 California     58
##  6 Colorado       64
##  7 Connecticut     8
##  8 Delaware        3
##  9 Florida        67
## 10 Georgia       159
## # i 40 more rows
```

count() tiene un parámetro `sort` que permite ordenar las observaciones según la cantidad de ocurrencias del valor en el atributo especificado.

```
counties %>%
  count(state, sort = TRUE)
```

```
## # A tibble: 50 x 2
##    state              n
##    <chr>          <int>
##  1 Texas            253
##  2 Georgia          159
##  3 Virginia         133
##  4 Kentucky         120
##  5 Missouri         115
##  6 Kansas           105
##  7 Illinois         102
##  8 North Carolina   100
##  9 Iowa              99
## 10 Tennessee         95
## # i 40 more rows
```

Otro parámetro importante de **count()** es `wt`, que se utiliza para especificar una variable que contiene pesos para cada observación. Esto significa que en lugar de contar cada observación como una unidad, **count()** utiliza los valores de la variable especificada en `wt` para ponderar las observaciones.

```
counties %>%
  count(state, wt = population, sort = TRUE)
```

```
## # A tibble: 50 x 2
##    state                n
##    <chr>            <dbl>
##  1 California    38421464
##  2 Texas         26538497
##  3 New York      19673174
##  4 Florida       19645772
##  5 Illinois      12873761
##  6 Pennsylvania  12779559
##  7 Ohio          11575977
##  8 Georgia       10006693
##  9 Michigan       9900571
## 10 North Carolina 9845333
## # i 40 more rows
```

Las funciones slice_min() y slice_max()

Ambas funciones se utilizan para seleccionar filas de un DataFrame basadas en los valores mínimos o máximos de una variable específica.

- **slice_min()**: Esta función devuelve las filas con los valores mínimos de la variable especificada.
- **slice_max()**: Esta función devuelve las filas con los valores máximos de la variable especificada.

Ambas funciones toman como argumentos el DataFrame y la variable por la cual se desea realizar la selección. Además, pueden tomar un argumento adicional opcional `n` para especificar cuántas filas deseas mantener. Si no se especifica este argumento, la función devolverá todas las filas que tienen el valor mínimo o máximo en la variable especificada.

```
counties_selected <- counties %>% select(state, county,
                                         population, unemployment, income)

counties_selected %>%
  group_by(state) %>%
  slice_max(population, n = 1)
```

```
## # A tibble: 50 x 5
## # Groups:   state [50]
##    state       county                 population unemployment income
##    <chr>       <chr>                       <dbl>        <dbl>  <dbl>
##  1 Alabama     Jefferson                  659026          9.1  45610
##  2 Alaska      Anchorage Municipality     299107          6.7  78326
##  3 Arizona     Maricopa                  4018143          7.7  54229
##  4 Arkansas    Pulaski                    390463          7.5  46140
##  5 California  Los Angeles              10038388         10    56196
##  6 Colorado    El Paso                    655024          8.4  58206
##  7 Connecticut Fairfield                  939983          9    84233
##  8 Delaware    New Castle                 549643          7.4  65476
##  9 Florida     Miami-Dade                2639042         10    43129
## 10 Georgia     Fulton                     983903          9.9  57207
## # i 40 more rows
```

```
counties_selected %>%
  group_by(state) %>%
  slice_min(unemployment, n = 1)
```

```
## # A tibble: 51 x 5
## # Groups:   state [50]
##    state       county                      population unemployment income
##    <chr>       <chr>                            <dbl>        <dbl>  <dbl>
##  1 Alabama     Shelby                          203530          5.5  70187
##  2 Alaska      Aleutians West Census Area        5684          2.1  84306
##  3 Arizona     Maricopa                       4018143          7.7  54229
##  4 Arkansas    Benton                          238198          4.2  56239
##  5 California  Marin                           258349          5.7  93257
##  6 Colorado    Jackson                           1335          1.5  46014
##  7 Connecticut Middlesex                       165165          6    79893
##  8 Delaware    New Castle                      549643          7.4  65476
```

```
##  9 Florida     Monroe                          75901          6    57290
## 10 Georgia     Bacon                           11222          4.4  37162
## # i 41 more rows
```

Seleccionar y transformar datos

La función select()

Anteriormente se ha mencionado que la función **select()** puede ser empleada para seleccionar atributos dentro de un DataFrame. Sin embargo, dicha función ofrece varias configuraciones adicionales para facilitar la selección de múltiples atributos de forma simultánea con los llamados `select_helpers`.

select_helper	**Descripción**
-var	Selecciona todas las variables excepto var
:	Selecciona un rango
contains()	Selecciona variables cuyo nombre contiene la cadena de texto
ends_with()	Selecciona variables cuyo nombre termina con la cadena de caracteres
everything()	Selecciona todas las columnas
matches()	Selecciona las variables cuyos nombres coinciden con una expresión regular
num_range()	Selecciona las variables por posición
one_of()	Selecciona variables cuyos nombres están en un grupo de nombres
start_with()	Selecciona variables cuyos nombres empiezan con la cadena de caracteres

Para más información: **?select_helpers**

Algunos ejemplos de uso:

```
counties %>%
  select(state, county, population, professional:production) %>%
  arrange(desc(service))
```

```
## # A tibble: 3,138 x 8
##    state   county population professional service office construction production
##    <chr>   <chr>       <dbl>        <dbl>   <dbl>  <dbl>        <dbl>      <dbl>
##  1 Missis~ Tunica      10477         23.9    36.6   21.5          3.5       14.5
##  2 Texas   Kinney       3577         30      36.5   11.6         20.5        1.3
##  3 Texas   Kenedy        565         24.9    34.1   20.5         20.5        0
##  4 New Yo~ Bronx     1428357         24.3    33.3   24.2          7.1       11
##  5 Texas   Brooks       7221         19.6    32.4   25.3         11.1       11.5
##  6 Colora~ Fremo~      46809         26.6    32.2   22.8         10.7        7.6
##  7 Texas   Culbe~       2296         20.1    32.2   24.2         15.7        7.8
##  8 Califo~ Del N~      27788         33.9    31.5   18.8          8.9        6.8
##  9 Minnes~ Mahno~       5496         26.8    31.5   18.7         13.1        9.9
## 10 Virgin~ Lanca~      11129         30.3    31.2   22.8          8.1        7.6
## # i 3,128 more rows
```

```r
counties %>%
  select(state, county, population, ends_with("work")) %>%
  filter(public_work > 50)
```

```
## # A tibble: 7 x 6
##   state        county            population private_work public_work family_work
##   <chr>        <chr>                  <dbl>        <dbl>       <dbl>       <dbl>
## 1 Alaska       Lake and Peninsu~       1474         42.2        51.6         0.2
## 2 Alaska       Yukon-Koyukuk Ce~       5644         33.3        61.7         0
## 3 California   Lassen                 32645         42.6        50.5         0.1
## 4 Hawaii       Kalawao                   85         25          64.1         0
## 5 North Dakota Sioux                   4380         32.9        56.8         0.1
## 6 South Dakota Todd                    9942         34.4        55           0.8
## 7 Wisconsin    Menominee               4451         36.8        59.1         0.4
```

La función rename()

La función **rename()** sirve para renombrar columnas.

```r
counties %>%
  count(state) %>%
  rename(num_counties = n)
```

```
## # A tibble: 50 x 2
##    state       num_counties
##    <chr>              <int>
##  1 Alabama               67
##  2 Alaska                28
##  3 Arizona               15
##  4 Arkansas              75
##  5 California            58
##  6 Colorado              64
##  7 Connecticut            8
##  8 Delaware               3
##  9 Florida               67
## 10 Georgia              159
## # i 40 more rows
```

Se puede renombrar directamente al seleccionar los atributos con **select()**

```
counties %>%
  select(state, county, poverty_rate = poverty)
```

```
## # A tibble: 3,138 x 3
##    state   county   poverty_rate
##    <chr>   <chr>           <dbl>
##  1 Alabama Autauga          12.9
##  2 Alabama Baldwin          13.4
##  3 Alabama Barbour          26.7
##  4 Alabama Bibb             16.8
##  5 Alabama Blount           16.7
##  6 Alabama Bullock          24.6
##  7 Alabama Butler           25.4
##  8 Alabama Calhoun          20.5
##  9 Alabama Chambers         21.6
## 10 Alabama Cherokee         19.2
## # i 3,128 more rows
```

La función relocate()

La función **relocate()** es empleada para cambiar de forma sencilla la posición de un atributo dentro del DataFrame. Para ello, emplea los parámetros `.before` y `.after` así como los `select_helpers`.

```
counties %>%
  relocate(region, .before = state)
```

```
## # A tibble: 3,138 x 40
##    census_id region state   county   metro population   men women hispanic white
##    <chr>     <chr>  <chr>   <chr>    <chr>      <dbl> <dbl> <dbl>    <dbl> <dbl>
##  1 1001      South  Alabama Autauga  Metro      55221 26745 28476      2.6  75.8
##  2 1003      South  Alabama Baldwin  Metro     195121 95314 99807      4.5  83.1
##  3 1005      South  Alabama Barbour  Nonm~      26932 14497 12435      4.6  46.2
##  4 1007      South  Alabama Bibb     Metro      22604 12073 10531      2.2  74.5
##  5 1009      South  Alabama Blount   Metro      57710 28512 29198      8.6  87.9
##  6 1011      South  Alabama Bullock  Nonm~      10678  5660  5018      4.4  22.2
##  7 1013      South  Alabama Butler   Nonm~      20354  9502 10852      1.2  53.3
##  8 1015      South  Alabama Calhoun  Metro     116648 56274 60374      3.5  73
##  9 1017      South  Alabama Chambers Nonm~      34079 16258 17821      0.4  57.3
## 10 1019      South  Alabama Cherokee Nonm~      26008 12975 13033      1.5  91.7
## # i 3,128 more rows
## # i 30 more variables: black <dbl>, native <dbl>, asian <dbl>, pacific <dbl>,
## #   citizens <dbl>, income <dbl>, income_err <dbl>, income_per_cap <dbl>,
## #   income_per_cap_err <dbl>, poverty <dbl>, child_poverty <dbl>,
## #   professional <dbl>, service <dbl>, office <dbl>, construction <dbl>,
## #   production <dbl>, drive <dbl>, carpool <dbl>, transit <dbl>, walk <dbl>,
## #   other_transp <dbl>, work_at_home <dbl>, mean_commute <dbl>, ...
```

```
counties %>%
  relocate(state, .after = region)
```

```
## # A tibble: 3,138 x 40
##    census_id county   region state   metro population   men women hispanic white
##    <chr>     <chr>    <chr>  <chr>   <chr>      <dbl> <dbl> <dbl>    <dbl> <dbl>
##  1 1001      Autauga  South  Alabama Metro      55221 26745 28476      2.6  75.8
##  2 1003      Baldwin  South  Alabama Metro     195121 95314 99807      4.5  83.1
##  3 1005      Barbour  South  Alabama Nonm~      26932 14497 12435      4.6  46.2
##  4 1007      Bibb     South  Alabama Metro      22604 12073 10531      2.2  74.5
##  5 1009      Blount   South  Alabama Metro      57710 28512 29198      8.6  87.9
##  6 1011      Bullock  South  Alabama Nonm~      10678  5660  5018      4.4  22.2
##  7 1013      Butler   South  Alabama Nonm~      20354  9502 10852      1.2  53.3
##  8 1015      Calhoun  South  Alabama Metro     116648 56274 60374      3.5  73
##  9 1017      Chambers South  Alabama Nonm~      34079 16258 17821      0.4  57.3
## 10 1019      Cherokee South  Alabama Nonm~      26008 12975 13033      1.5  91.7
## # i 3,128 more rows
## # i 30 more variables: black <dbl>, native <dbl>, asian <dbl>, pacific <dbl>,
## #   citizens <dbl>, income <dbl>, income_err <dbl>, income_per_cap <dbl>,
## #   income_per_cap_err <dbl>, poverty <dbl>, child_poverty <dbl>,
## #   professional <dbl>, service <dbl>, office <dbl>, construction <dbl>,
## #   production <dbl>, drive <dbl>, carpool <dbl>, transit <dbl>, walk <dbl>,
## #   other_transp <dbl>, work_at_home <dbl>, mean_commute <dbl>, ...
```

```
counties_selected %>%
  relocate(state, .after = last_col())
```

```
## # A tibble: 3,138 x 5
##    county   population unemployment income state
##    <chr>         <dbl>        <dbl>  <dbl> <chr>
##  1 Autauga       55221          7.6  51281 Alabama
##  2 Baldwin      195121          7.5  50254 Alabama
##  3 Barbour       26932         17.6  32964 Alabama
##  4 Bibb          22604          8.3  38678 Alabama
##  5 Blount        57710          7.7  45813 Alabama
##  6 Bullock       10678         18    31938 Alabama
##  7 Butler        20354         10.9  32229 Alabama
##  8 Calhoun      116648         12.3  41703 Alabama
##  9 Chambers      34079          8.9  34177 Alabama
## 10 Cherokee      26008          7.9  36296 Alabama
## # i 3,128 more rows
```