Procesamiento de Datos

Angela Di Serio

Contents

Tidyverse

La palabra **tidyverse** hace referencia a una nueva forma de afrontar el análisis de datos en R. Se hace uso de un grupo de paquetes que trabajan bajo ciertos principios, como por ejemplo, la forma de estructurar los datos.

Los principales paquetes del tidyverse son:

- readr para importar datos
- tidyr para convertir los datos a tidy data
- dplyr para manipular datos
- ggplot2 para hacer gráficos
- stringr para trabajar con cadenas

Instalación de Tidyverse

Instalar el paquete tidyverse en Amazon WorkSpaces usando la función install.packages("tidyverse"), esta instalación puede tomar mucho tiempo asi que para nuestros ejercicios podemos instalar individualmente los principales paquetes de tidyverse:

- install.packages("readr")
- install.packages("tidyr")
- install.packages("dplyr")
- install.packages("ggplot2")
- install.packages("stringr")

dplyr

dplyr es un paquete que permite manipular datos de forma intuitiva. Tiene un grupo de **funciones** o **verbos**. Cada uno de ellos hace "una sola cosa", así que para realizar transformaciones complejas hay que ir concatenando instrucciones sencillas. Esto se hace con el operador pipe (%>%). Similar al operador | de Linux.

Todas las funciones tienen una estructura o comportamiento similar:

- el primer argumento siempre es un dataframe
- los siguientes argumentos describen qué hacer con los datos
- el resultado es siempre un nuevo dataframe

Las principales funciones son:

- filter() : permite seleccionar filas que cumplen con una o varias condiciones
- arrange(): reordena las filas
- rename() : cambia los nombres de las columnas o variables
- select() : selecciona columnas
- mutate() : crea nuevas variables
- summarise() : resume (colapsa) unos cuantos valores a uno sólo. Por ejemplo, calcula la media, moda, etc... de un conjunto de valores
- group_by() : permite agrupar filas en función de una o varias condiciones

Operador pipe %>%

Este operador es básico en el tidyverse, ya que permite encadenar llamadas a funciones para así realizar de forma sencilla transformaciones de datos complejas (similar al pipe | de Linux).

Este operador pasa el elemento que está a su izquierda como un argumento de la función que tiene a la derecha.

```
library(tidyr)
```

```r
head(iris, n = 4)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
```

```r
iris %>% head(n = 4)  # %>% pasa lo que hay a la izquierda como argumento de la función
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
```

filter()

Esta función se utiliza para seleccionar filas de un dataframe. Se seleccionan las filas que cumplen una determinada condición o criterio lógico.

```r
library(gapminder)
```

```r
df = gapminder
head(df)
```

```
## # A tibble: 6 x 6
##   country     continent  year lifeExp      pop gdpPercap
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779.
## 2 Afghanistan Asia       1957    30.3  9240934      821.
## 3 Afghanistan Asia       1962    32.0 10267083      853.
## 4 Afghanistan Asia       1967    34.0 11537966      836.
## 5 Afghanistan Asia       1972    36.1 13079460      740.
## 6 Afghanistan Asia       1977    38.4 14880372      786.
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
spain = df %>% filter(country == "Spain")
spain
```

```
## # A tibble: 12 x 6
##    country continent  year lifeExp      pop gdpPercap
##    <fct>   <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Spain   Europe     1952    64.9 28549870     3834.
##  2 Spain   Europe     1957    66.7 29841614     4565.
##  3 Spain   Europe     1962    69.7 31158061     5694.
##  4 Spain   Europe     1967    71.4 32850275     7994.
##  5 Spain   Europe     1972    73.1 34513161    10639.
##  6 Spain   Europe     1977    74.4 36439000    13237.
##  7 Spain   Europe     1982    76.3 37983310    13926.
##  8 Spain   Europe     1987    76.9 38880702    15765.
##  9 Spain   Europe     1992    77.6 39549438    18603.
## 10 Spain   Europe     1997    78.8 39855442    20445.
## 11 Spain   Europe     2002    79.8 40152517    24835.
## 12 Spain   Europe     2007    80.9 40448191    28821.
```

Noten que cuando usamos %>% no hay necesidad de indicar que **country** es un campo del objeto **df.**

```
x = df %>% filter(lifeExp >=  29 , lifeExp <= 32)
x = df %>% filter(lifeExp >=  29 &  lifeExp <= 32)
x = df %>% filter(between(lifeExp, 29, 32))
x
```

```
## # A tibble: 10 x 6
##    country      continent  year lifeExp      pop gdpPercap
##    <fct>        <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan  Asia       1957    30.3  9240934      821.
##  2 Afghanistan  Asia       1962    32.0 10267083      853.
##  3 Angola       Africa     1952    30.0  4232095     3521.
##  4 Angola       Africa     1957    32.0  4561361     3828.
##  5 Burkina Faso Africa     1952    32.0  4469979      543.
##  6 Cambodia     Asia       1977    31.2  6978607      525.
##  7 Gambia       Africa     1952    30     284320      485.
##  8 Mozambique   Africa     1952    31.3  6446316      469.
##  9 Sierra Leone Africa     1952    30.3  2143249      880.
## 10 Sierra Leone Africa     1957    31.6  2295678     1004.
```

arrange()

Esta función se utiliza para reordenar las filas de un dataframe (df).

```
df = gapminder

#- ordena las filas de MENOR a mayor según los valores de la v. lifeExp
aa = df %>% arrange(lifeExp)

#- ordena las filas de MAYOR a menor según los valores de la v. lifeExp
aa = df %>% arrange(desc(lifeExp))

#- ordenada las filas de MENOR a mayor según los valores de la v. lifeExp.
#- Si hay empates se resuelve con la variable "pop"
aa = df %>% arrange(lifeExp, pop)

head(aa)
```

```
## # A tibble: 6 x 6
##   country      continent  year lifeExp     pop gdpPercap
##   <fct>        <fct>     <int>   <dbl>   <int>     <dbl>
## 1 Rwanda       Africa     1992    23.6 7290203      737.
## 2 Afghanistan  Asia       1952    28.8 8425333      779.
## 3 Gambia       Africa     1952    30    284320      485.
## 4 Angola       Africa     1952    30.0 4232095     3521.
## 5 Sierra Leone Africa     1952    30.3 2143249      880.
## 6 Afghanistan  Asia       1957    30.3 9240934      821.
```

select()

Se utiliza para seleccionar columnas de un dataframe.

```
#- Se lee como: "Take el df gapminder, then select the variables year and lifeExp"
aa = df %>% select(year, lifeExp)
aa = df %>% select(c(year, lifeExp))

head(aa)
```

```
## # A tibble: 6 x 2
##    year lifeExp
##   <int>   <dbl>
## 1  1952    28.8
## 2  1957    30.3
## 3  1962    32.0
## 4  1967    34.0
## 5  1972    36.1
## 6  1977    38.4
```

mutate()

Sirve para crear nuevas variables (columnas).

```
df = gapminder

# Creamos la variable: GDP = pop*gdpperCap
aa = df %>% mutate(GDP = pop*gdpPercap)
head(aa)
```

```
## # A tibble: 6 x 7
##   country     continent  year lifeExp      pop gdpPercap          GDP
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>        <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779.  6567086330.
## 2 Afghanistan Asia       1957    30.3  9240934      821.  7585448670.
## 3 Afghanistan Asia       1962    32.0 10267083      853.  8758855797.
## 4 Afghanistan Asia       1967    34.0 11537966      836.  9648014150.
## 5 Afghanistan Asia       1972    36.1 13079460      740.  9678553274.
## 6 Afghanistan Asia       1977    38.4 14880372      786. 11697659231.
```

summarise()

Sirve para RESUMIR (o "colapsar filas"). Coge un grupo de valores como input y devuelve un solo valor; por ejemplo, la media aritmética (o el mínimo, o el máximo) de un grupo de valores.

```
aa = df %>% summarise(media = mean(lifeExp))

#- retornará un único valor: el número de filas
aa = df %>% summarise(NN = n())  #- retornará un único valor: el número de filas

#- retornará un único valor: la desviación típica de la v. "lifeExp"
aa = df %>% summarise(desviacion_tipica = sd(lifeExp))

#- retornará un único valor: el máximo de la variable "pop"
aa = df %>% summarise(max(pop))
```

group_by()

En análisis de datos, muchas operaciones (media, desviación, etc.) queremos calcularlas para distintos grupos (hombre, mujer). Esta función permite hacerlo.

group_by() coge un dataframe y lo convierte en un "dataframe agrupado". En ese nuevo "dataframe agrupado", las operaciones que hagamos con summarise() se harán por separado para cada uno de los grupos que hayamos definido.

Si, por ejemplo, agrupamos un dataframe por continente, al ejecutar summarise, nos retornará una fila con el resultado para cada continente.

```r
aa = df %>% group_by(continent) %>%  summarise(NN = n())
aa
```

```
## # A tibble: 5 x 2
##   continent    NN
##   <fct>     <int>
## 1 Africa      624
## 2 Americas    300
## 3 Asia        396
## 4 Europe      360
## 5 Oceania      24
```

```r
df = read.csv("personal.csv",header=TRUE, sep=",")
```

```r
library(dplyr)
df %>% filter(edad > 40) %>% group_by(estado_civil) %>%
  summarise(media = mean(edad))
```

```
## # A tibble: 3 x 2
##   estado_civil media
##   <chr>        <dbl>
## 1 C               54
## 2 D               45
## 3 V               61
```

La anterior línea de código R hace:

- coge los datos del dataframe df
- selecciona (o filtra) las filas que cumplen que el valor de edad es mayor que 40
- agrupa los datos por la variable estado_civil y
- calcula la media de edad

Con esta sintáxis que permite el operador pipe, no necesitamos anidar funciones, sino que las instrucciones van una después de otra. Es mucho más fácil de leer y de escribir.

tidyr - Tidy data

La mayoría de datos son de tipo tabular; es decir, organizados en filas y columnas. En R este tipo de datos se almacenan en dataframes (o tibbles). En esencia, un dataframe será tidy si cada columna es una variable

y cada fila es una unidad de análisis (persona, país, región etc. . .); es decir, cada celda contiene el valor de una variable para una unidad de análisis.

Un ejemplo de datos no tidy

año	Pedro	Carla	Mario
2014	100	400	200
2015	500	600	700
2016	200	250	900

Un ejemplo tidy pero ancho

```
data_2 = data.frame(nombre = c("Pedro", "Carla", "Mario"),
                    W_2014 = c(100, 400, 200),
                    W_2015 = c(500, 600, 700),
                    W_2016 = c(200, 250, 900) )
```

nombre	W_2014	W_2015	W_2016
Pedro	100	500	200
Carla	400	600	250
Mario	200	700	900

Un ejemplo de datos tidy y long

```
data_3 = data.frame(nombre = c("Pedro", "Carla", "Mario", "Pedro", "Carla", "Mario", "Pedro", "Carla",

data_3
```

```
##   nombre year salario
## 1  Pedro 2014     100
## 2  Carla 2014     400
## 3  Mario 2014     200
## 4  Pedro 2015     500
## 5  Carla 2015     600
## 6  Mario 2015     700
## 7  Pedro 2016     200
## 8  Carla 2016     250
## 9  Mario 2016     900
```

Los paquetes del tidyverse trabajan mejor con tidy data en formato “long”.

¿Qué hacemos si tenemos un dataframe en formato wide? Debemos pasarlo a long. Para ello usaremos la función gather() del paquete tidyr.

wide a long - gather()

La función gather() convierte dataframes de formato wide a long.

```
library(tidyr)
data_2
```

```
##   nombre W_2014 W_2015 W_2016
## 1  Pedro    100    500    200
## 2  Carla    400    600    250
## 3  Mario    200    700    900
```

```
data_wide = data_2   #- data_2 está en formato ancho (wide)
data_long = data_wide %>% gather(periodo, salario, 2:4)

data_long
```

```
##   nombre periodo salario
## 1  Pedro  W_2014     100
## 2  Carla  W_2014     400
## 3  Mario  W_2014     200
## 4  Pedro  W_2015     500
## 5  Carla  W_2015     600
## 6  Mario  W_2015     700
## 7  Pedro  W_2016     200
## 8  Carla  W_2016     250
## 9  Mario  W_2016     900
```

Los periodos aparecen como W_2014, W_2015 y W_2016 y muy probablemente nos interese tenerlos como 2014, 2015 y 2016.

```
library(stringr)
data_long = data_long %>% mutate(periodo = str_replace(periodo, "W_", "" ))

data_long
```

```
##   nombre periodo salario
## 1  Pedro    2014     100
## 2  Carla    2014     400
## 3  Mario    2014     200
## 4  Pedro    2015     500
## 5  Carla    2015     600
## 6  Mario    2015     700
## 7  Pedro    2016     200
## 8  Carla    2016     250
## 9  Mario    2016     900
```

Y si los datos queremos pasarlos de long a wide usaremos la función spread().

long a wide - spread()

Pasar pasar de long a wide, tidyr tiene la función spread()

```
data_wide2 = data_long %>% spread(periodo, salario)
data_wide2
```

```
##   nombre 2014 2015 2016
## 1  Carla  400  600  250
## 2  Mario  200  700  900
## 3  Pedro  100  500  200
```

spread(data, key, value, . . .)

- data es un data frame
- key nombre de columnas o posiciones
- value el valor que será reportado en las posiciones