Dplyr

El *Tidyverse*.

El ***Tidyverse*** es un conjunto de paquetes / librerías con una filosofía común, como es el uso de ciertas estructuras gramaticales, que facilitan muchas de las tareas y análisis que podrían hacerse con el lenguaje R estándar. Uno de esos paquetes es ggplot2, que proporciona unas herramientas muy flexibles para visualizar gráficamente conjuntos de datos. Otro de estos paquetes es el que se va a comentar aquí, dplyr, que proporciona una gramática más sencilla que la del lenguaje R convencional para manipular los objetos de estructuras de datos conocidos como ***data frames***.

Los *data frames* son estructuras en las que se guardan datos de modo que, por columnas, se disponen las variables del análisis; y por filas los casos que conforman la muestra / población.

Obteniendo los datos.

Vamos a suponer que trabajamos dentro de un proyecto que hemos creado previamente, de nombre "explora". Dentro de la carpeta del proyecto guardaremos el script llamado "explora_dplyr.R" y el archivo de Microsoft® Excel® llamado "eolica_20.xlsx". Si abrimos este último archivo, comprobaremos que se compone de tres hojas. La primera muestra el criterio de búsqueda de casos en la base de datos Sabi®; la segunda recoge la descripción de las variables consideradas; y la tercera (hoja "Top 20") guarda los datos que debemos importar desde R-Studio. Estos datos se corresponden con diferentes variables económico-financieras de las 20 empresas productoras de electricidad mediante generación eólica con mayor volumen de activo total.

Luego cerraremos el archivo de Microsoft® Excel®, "eolica_20.xlsx", y volveremos a R-Studio. Después, abriremos nuestro script "explora_dplyr.R" con File → Open File… Este script contiene el programa que vamos a ir ejecutando en la práctica.

La primera línea / instrucción en los scripts suele ser:

```
rm(list = ls())
```

La instrucción tiene como objeto limpiar el *Environment* (memoria) de objetos de anteriores sesiones de trabajo.

Para importar los datos que hay en la hoja "Top 20" del archivo de Microsoft® Excel® llamado "eolica_20.xlsx", ejecutaremos el código:

```
# DATOS

library(readxl)
eolica_20 <- read_excel("eolica_20.xlsx", sheet = "Top 20")
```

Podemos observar como en el *Environment* ya aparece un objeto. Este objeto es una estructura de datos tipo *data frame*, se llama "eolica_20" (si importamos mediante las facilidades de R-Studio, el *data frame*, por defecto, adopta el nombre del archivo del que proceden los datos, aunque podemos cambiar el nombre si queremos), y contiene 11 columnas. R ha considerado la primera columna como una variable de tipo cualitativo, como comprobaremos haciendo un summary():

```
summary (eolica_20)
```

En realidad, la primera columna no es una variable, sino que está formada por el nombre (identificador) de los diferentes casos u observaciones. Para evitar que R tome los nombres de los casos como una variable más, podemos redefinir nuestro *data frame* diciéndole que tome esa primera columna como los *nombres de los individuos*:

```
eolica_20 <- data.frame(eolica_20, row.names = 1)
```

En la línea anterior hemos asignado al *data frame* "eolica_20" los propios datos de "eolica_20"; pero indicando que la primera columna de datos no es una variable; sino el nombre de los casos. Si hacemos ahora un summary():

```
summary (eolica_20)
```

Vemos que ya no aparece NOMBRE como variable y que, en el *Environment*, se recoge el *data frame* "eolica_20" con 20 casos y con 10 variables.

Cargando *dplyr*.

A continuación, cargaremos el paquete dplyr. Si nunca antes se ha utilizado este paquete, cuando lo intentemos activar con la función library() nos dará un error o nos dirá que previamente hay que importarlo. En ese caso, iremos a la ventana inferior-derecha y pulsaremos la pestaña "Packages", pulsaremos en "Install", y emergerá una ventana donde dejaremos el "repositorio" que viene por defecto y, en el campo "Packages", escribiremos el nombre del "paquete" (en nuestro caso dplyr). Una vez descargado el "paquete", podremos ejecutar el código sin problemas.:

```
#Cargando dplyr

library (dplyr)
```

Para entender mejor la **gramática** que siguen las funciones o instrucciones a las que da acceso dplyr, hay que tener en cuenta lo siguiente:

- El primer argumento que tiene una función de dplyr es el *data frame* con el que se va a trabajar.
- Los otros argumentos describen qué hay que hacer con el *data frame* especificado en el primer argumento. Es posible referirse a las columnas (variables) del *data frame* con su nombre, **sin utilizar el operador $**.
- El valor de retorno es un **nuevo** *data frame*.

En los siguientes apartados practicaremos con algunas de las principales funciones que aporta dplyr.

Seleccionando columnas de un *data frame*.

La función clave de dplyr para seleccionar una o varias columnas (variables) de un data frame es la función select().

Así, vamos a imaginar por ejemplo que queremos eliminar de nuestro *data frame* la variable (de *tipo* "carácter") MATRIZ. Podremos ejecutar la asignación:

```
#Seleccionando variables

eolica_20 <-select(eolica_20, -MATRIZ)
summary (eolica_20)
```

Podemos verificar que, en el *Environment,* el *data frame* ha pasado a tener una variable menos (9), ya que hemos eliminado la variable MATRIZ. Es decir, con el guión "-" se pueden eliminar directamente variables de un *data frame*.

Ahora, suponemos que queremos visualizar las variables del *data frame* "eolica_20": ACTIVO, FPIOS, LIQUIDEZ, MARGEN, SOLVENCIA y APALANCA (es decir, todas las variables menos RES, RENECO, RENFIN). Para ello, ejecutaremos el código:

```
select(eolica_20, ACTIVO, FPIOS, LIQUIDEZ, MARGEN, SOLVENCIA,
APALANCA)
```

Con lo anterior, obtendremos en la *consola*:

```
                                ACTIVO       FPIOS LIQUIDEZ   MARGEN
Holding De Negocios De GAS SL.  13492812.0 6904824.000    1.020   91.152
Global Power Generation SA.      2002458.0 1740487.000    2.006   22.403
Naturgy Renovables SLU           1956869.0  318475.000    1.263   20.442
EDP Renovables España SLU        1275939.0  726783.000    1.596   47.193
Corporacion Acciona Eolica SL     864606.0  136064.000    0.788   20.091
Saeta Yield SA.                   796886.4  665319.556    2.687   16.258
Elawan Energy SL.                 443467.0  186302.006    0.595  208.357
Olivento SL                       381207.0   58340.998    0.771   16.629
Parque Eolico La Boga SL.         303904.4   29316.797    1.407    1.001
Naturgy Wind, S.L.                273542.0   28418.000    1.364   39.575
Viesgo Renovables SL.             269730.0  177707.000    0.272   11.818
Al-Andalus Wind Power SL          249853.8   21466.121    1.550   12.582
Innogy Spain SA.                  230338.5   85447.212    1.416  -18.025
Guzman Energia SL                 190287.0  -77532.698    0.078  -19.193
Acciona Eolica Del Levante SL     188354.0   21769.000    2.855   27.520
Biovent Energia SA                183899.0   70033.000    1.206   22.792
Esquilvent SL                     157630.6   48769.130    5.330   39.476
Eolica La Janda SL                153429.4   25206.748    1.184   38.256
Parque Eolico Santa Catalina SL   147742.5   -1664.755    0.388   31.780
WPD Wind Investment SL.           109023.8  108023.826    0.624 -302.027
                                SOLVENCIA  APALANCA
Holding De Negocios De GAS SL.     51.174    91.964
Global Power Generation SA.        86.917     1.044
Naturgy Renovables SLU             16.274   494.729
EDP Renovables España SLU          56.960    67.028
Corporacion Acciona Eolica SL      15.737   422.263
Saeta Yield SA.                    83.489    17.067
Elawan Energy SL.                  42.010   123.771
Olivento SL                        15.304   534.761
Parque Eolico La Boga SL.           9.646   921.591
Naturgy Wind, S.L.                 10.388   824.537
```

```
Viesgo Renovables SL.              65.883    13.330
Al-Andalus Wind Power SL            8.591  1019.616
Innogy Spain SA.                   37.096   150.688
Guzman Energia SL                 -40.745  -343.542
Acciona Eolica Del Levante SL      11.557   743.754
Biovent Energia SA                 38.082   141.163
Esquilvent SL                      30.938   218.275
Eolica La Janda SL                 16.428   480.122
Parque Eolico Santa Catalina SL    -1.126 -6265.496
WPD Wind Investment SL.            99.082     0.000
```

Como no hemos asignado el resultado de la función a ningún "nombre", R simplemente saca el resultado en pantalla; pero no guarda ningún objeto en el *Environment*. Si un "select()" lo asignamos a un "nombre", se creará un *data frame* con ese nombre, y las variables seleccionadas:

```
eolica_20A <-select(eolica_20, ACTIVO, FPIOS, LIQUIDEZ, MARGEN,
SOLVENCIA, APALANCA)
summary (eolica_20A)
```

Podemos comprobar en el *Environment* cómo hay otro objeto *data frame* llamado "eolica_920A", con 5 variables (y los mismos 20 casos). Este *data frame* lo podríamos haber creado, también, eliminando del *data frame* original ("eolica_20"), las variables que nos sobran:

```
eolica_20A <-select(eolica_20, -RES, -RENECO, -RENFIN)
eolica_20A
```

Vemos cómo se obtiene el mismo resultado. Más aún, si nos fijamos bien, los nombres de todas las variables que hemos excluido empiezan por "RE", a diferencia de las incluidas. Podríamos haber hecho también:

```
eolica_20A <-select(eolica_20, -(starts_with("RE")))
eolica_20A
```

Y de nuevo obtendríamos el mismo resultado. El argumento starts_with() permite seleccionar variables cuyos nombres comienzan por cierta cadena de caracteres. También se puede hacer mismo con los caracteres finales (ends_with()) o contenidos en alguna posición del nombre (contains()).

```
eolica_20A <-select(eolica_20, -(starts_with("RE")))
summary (eolica_20A)
```

Otra posibilidad que tenemos es hacer una copia de un *data frame* rápidamente con el argumento everything(). Por ejemplo:

```
eolica_20_replica <-select(eolica_20, everything())
summary (eolica_20_replica)
```

Se ha creado el *date frame* "eolica_920_replica" que es una copia exacta de "eolica_20".

Seleccionando casos de un *data frame*.

Además de seleccionar variables, con dplyr también se pueden seleccionar casos que cumplan ciertas condiciones. La función para realizar este cometido es filter().

Por ejemplo, si queremos seleccionar las empresas eólicas con un resultado (variable RES) mayor o igual a 50.000 y presentarlas en pantalla, la instrucción será:

```
filter(eolica_20, RES >= 50000)
```

Y en la consola se visualizará:

```
                                RES   ACTIVO   FPIOS RENECO RENFIN LIQUIDEZ MARGEN
Holding De Negocios De GAS SL. 727548 13492812 6904824  5.264 10.287    1.020 91.152
EDP Renovables España SLU       67033  1275939  726783  6.458 11.338    1.596 47.193
                               SOLVENCIA APALANCA
Holding De Negocios De GAS SL.    51.174   91.964
EDP Renovables España SLU         56.960   67.028
```

Se pueden incluir varias condiciones en un mismo filtro. Por ejemplo, vamos a construir un nuevo *data frame* llamado "eolica_20B" con las empresas que posean un resultado mayor o igual a 50000 y una rentabilidad económica (variable RENECO) inferior al 6%:

```
eolica_20B <-filter(eolica_20, RES >= 50000 & RENECO < 6)
eolica_20B
```

En el *Environment* aparecerá el *data frame* "eolica_9B" con solo un caso: la empresa que cumple con ambas condiciones, introducidas mediante el operador lógico relacional "**&**".

Los filtros más usuales son >, <, >=, <=, == (igual, ojo, con dos símbolos de igualdad seguidos), != (no igual). En cuanto a los operadores para ligar filtros o condiciones, tenemos & (y), \ (o), xor (o inclusivo), ! (no), any (cualquiera verdadero), all (todos verdaderos).

Ordenando casos de un *data frame*.

Además de seleccionar determinados casos u observaciones (filas) de un *data frame*, con las funciones de dplyr también se pueden ordenar estos casos a partir de los valores de ciertas variables (columnas). La función a utilizar es arrange(). Esta función, por defecto, ordena los casos de modo **ascendente**. Por ejemplo:

```
arrange(eolica_20, RENECO)
```

En cambio, para ordenar de modo descendente, hay que utilizar el argumento desc():

```
arrange(eolica_20, desc(RENECO))
```

En el supuesto de que, por ejemplo, hubiera varias empresas con la misma rentabilidad económica (RENECO), podría añadirse otro criterio de ordenación con otra variable, que afectaría a tales empresas para deshacer el "empate" en rentabilidad económica. Por ejemplo, para ordenar de modo ascendente por rentabilidad y, en caso de que haya rentabilidades iguales, por liquidez (variable LIQUIDEZ), se ejecutaría:

```
arrange(eolica_20, RENECO, LIQUIDEZ)
```

Obviamente, en este ejemplo concreto el resultado es el mismo que se obtuvo con `arrange(eolica_20, RENECO)`, puesto que no hay rentabilidades iguales entre las 20 empresas de la muestra.

Cambiando el nombre de las variables en un *data frame*.

Con R base es dificultoso cambiar el nombre de una variable (columna de un *data frame*). En cambio, dplyr cuenta con una función que lo hace directamente: la función rename(). Por ejemplo, si queremos cambiar el nombre de la variable SOLVENCIA por SOLVE, simplemente ejecutaremos:

```
#Renombrando variables

eolica_20 <- rename(eolica_20, SOLVE = SOLVENCIA)
summary (eolica_20)
```

Podemos comprobar en el *Environment*, despegando el objeto "eolica_20", cómo ya no aparece la variable SOLVENCIA; pero sí SOLVE en su lugar (obviamente, con los mismos datos). Es necesario tener en cuenta que en el **lado izquierdo** de la igualdad hay que poner el **nuevo nombre**, y en la derecha el antiguo. Además, en el mismo rename() se pueden cambiar los nombres de **varias variables**, separando las igualdades correspondientes con comas.

Añadiendo variables como transformación de otras variables en un *data frame*.

El paquete dplyr permite también añadir a un *data frame* variables que son el resultado de someter a otras variables a diversas transformaciones. La función para realizar este cometido es mutate().

Así, por ejemplo, imaginemos que necesitamos calcular una variable como el cociente entre los resultados obtenidos y el activo. A esta nueva variable la denominaremos RATIO. El código será:

```
# Añadiendo variables como transformacion de otras variables

eolica_20 <- mutate (eolica_20, RATIO = RES / ACTIVO)
summary(eolica_20)
```

En la transformación de variables mediante la función mutate(), se pueden utilizar **funciones integradas en otros paquetes** de R. Por ejemplo, si queremos calcular la variable ACTIVOS_ACUM como la variable que recoge los activos acumulados de las empresas, comenzando por la empresa con menor activo, podríamos utilizar la función cumsum() del paquete bas, y hacer:

```
eolica_20 <- arrange(eolica_20, ACTIVO)
eolica_20
eolica_20 <- mutate (eolica_20, ACTIVOS_ACUM = cumsum(ACTIVO))
eolica_20
```

Podemos verificar cómo se ha integrado en el *data frame* la variable ACTIVOS_ACUM.

Un último ejemplo de adición de una variable que es transformación de otras. En este caso, crearemos la variable "TAM" (tamaño), que es **categórica** (carácter). Esta variable toma valor "G" para las empresas con

un valor de la variable ACTIVO mayor que 1000000, y "P" para las que tengan un valor en la variable ACTIVO menor o igual a 1000000. Para calcular automáticamente esta nueva variable categórica, utilizaremos la función de R base cut(). De este modo, haremos:

```
eolica_20 <- mutate(eolica_20, TAM = cut(ACTIVO, breaks = c(-Inf,
1000000, Inf), labels = c("P", "G")))
eolica_20
```

Podemos advertir cómo la función cut(), que incluimos dentro de nuestra función de dplyr mutate(), tiene, a su vez, varios argumentos: la variable numérica de referencia (ACTIVO); el argumento **"breaks"**, en el que decimos los intervalos en que quedarán divididos los casos (uno de menos infinito a 1000000, y otro de 1000000 a más infinito), y **"labels"**, que es el valor que tomará la variable creada (TAM) según el intervalo en el que se sitúe cada caso de la muestra. Es fácil ver el resultado con:

```
select(eolica_20, ACTIVO, TAM)
```

Cabe destacar que podíamos haber escrito el código para crear la variable TAM de un modo más elegante, utilizando el operador **"*pipe*"** (**%>%**). Este operador permite concatenar una serie de instrucciones de un modo más cómodo:

```
eolica_20 <- eolica_20 %>% mutate(TAM = cut(ACTIVO, breaks = c(-Inf,
1000000, Inf), labels = c("P", "G")))
eolica_20
select(eolica_20, ACTIVO, TAM)
```

Podríamos interpretar la línea de código así: "asigna al *data frame* "eolica_20" sus propios datos, después (%>%) crea la variable TAM con la función cut() y añádela a "eolica_20"".

Extrayendo información de las variables de un *data frame*.

Otra posibilidad que permite dplyr es extraer y sintetizar la información de las variables contenidas en las variables de un *data frame*. Para ello, nos ayudaremos de la función summarise(). Como ejemplo, calculemos la rentabilidad financiera media de las 20 empresas:

```
#Extrayendo información de las variables de un data frame

summarise(eolica_20, RENFIN_media = mean(RENFIN))
```

Obteniéndose como resultado:

```
  RENFIN_media
1     -3.45005
```

A veces, es de gran utilidad combinar summarise() con group_by(), que extrae la información por grupos definidos por una de las variables. Para ilustrarlo, vamos a utilizar la variable recién creada TAM, para hacer dos grupos de empresas: las de menor ("P") y las de mayor ("G") volumen de activo, y calcularemos la media de las rentabilidades para cada grupo:

```
eolica_20 %>% group_by(TAM) %>% summarise(RENFIN_media = mean(RENFIN))
```

Con lo que se obtiene:

```
# A tibble: 2 × 2
  TAM   RENFIN_media
  <fct>        <dbl>
1 P            -6.52
2 G             8.82
```

Hemos utilizado el operador "pipe" **%>%** para concatenar diferentes instrucciones de dplyr: primero agrupar casos, y luego calcular las medias de cada grupo. Es decir, en este caso se podría "traducir" la línea de código como: "Toma el *data frame* "eolica_9", divide a los casos según el valor de la variable TAM, y para cada grupo calcula la media de la variable RENFIN".

Updated: 27/09/2022