Uso de la librería dplyr

Code

`dplyr` aplicaciones

Es una librería centrada en agrupación de datos y selección de los mismos.

Las operaciones centrales que podemos realizar:

1. select() seleccionar columnas
2. filter() seleccionar filas
3. arrange() ordenar los datos
4. mutate() transformar una columna
5. summarise() calcular características estadísticas sobre variables
6. group_by() agrupar los datos por una clave, habitualmente una o varias columnas
7. left_join() ensamblar dataframes

Veamos con ejemplos lo que puede hacer:

Primero, instalamos y cargamos la librería.

Hide

```
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
```

Cargamos los datos y observamos las primeras filas.

Hide

```
data("airquality")
head(airquality, 10) # variables
```

	Ozone <int>	Solar.R <int>	Wind <dbl>	Temp <int>	Month <int>	Day <int>
1	41	190	7.4	67	5	1
2	36	118	8.0	72	5	2
3	12	149	12.6	74	5	3
4	18	313	11.5	62	5	4
5	*NA*	*NA*	14.3	56	5	5

	Ozone <int>	Solar.R <int>	Wind <dbl>	Temp <int>	Month <int>	Day <int>
6	28	*NA*	14.9	66	5	6
7	23	299	8.6	65	5	7
8	19	99	13.8	59	5	8
9	8	19	20.1	61	5	9
10	*NA*	194	8.6	69	5	10

1-10 of 10 rows

select Seleccionamos variables específicas de nuestro data.frame

Hide

```
select(airquality, Ozone, Solar.R, Wind) # seleccionamos las variables Ozone, Sola
r.R y Wind
```

Ozone <int>	Solar.R <int>	Wind <dbl>
41	190	7.4
36	118	8.0
12	149	12.6
18	313	11.5
NA	*NA*	14.3
28	*NA*	14.9
23	299	8.6
19	99	13.8
8	19	20.1
NA	194	8.6

1-10 of 153 rows Previous **1** 2 3 4 5 6 … 16 Next

filter Filtramos los registros de nuestro data.frame según los valores que tomen ciertas variables.

Hide

```
filter(airquality, Temp > 70) # filtramos los registros con temperatura superior a
70
```

Ozone <int>	Solar.R <int>	Wind <dbl>	Temp <int>	Month <int>	Day <int>
36	118	8.0	72	5	2
12	149	12.6	74	5	3

Ozone <int>	Solar.R <int>	Wind <dbl>	Temp <int>	Month <int>	Day <int>
7	*NA*	6.9	74	5	11
11	320	16.6	73	5	22
45	252	14.9	81	5	29
115	223	5.7	79	5	30
37	279	7.4	76	5	31
NA	286	8.6	78	6	1
NA	287	9.7	74	6	2
NA	186	9.2	84	6	4

1-10 of 120 rows Previous **1** 2 3 4 5 6 … 12 Next

Hide

```
filter(airquality, Temp > 80 & Month > 5) # igual pero añadimos la condición mes superior a 5
```

Ozone <int>	Solar.R <int>	Wind <dbl>	Temp <int>	Month <int>	Day <int>
NA	186	9.2	84	6	4
NA	220	8.6	85	6	5
29	127	9.7	82	6	7
NA	273	6.9	87	6	8
71	291	13.8	90	6	9
39	323	11.5	87	6	10
NA	259	10.9	93	6	11
NA	250	9.2	92	6	12
23	148	8.0	82	6	13
NA	138	8.0	83	6	30

1-10 of 67 rows Previous **1** 2 3 4 5 6 7 Next

arrange Ordenamos los registros en base a una o varias variables.

Hide

```
arrange(airquality, desc(Month), Day) # ordena según mes (de manera descendente) y por día (de manera ascendente)
```

Ozone <int>	Solar.R <int>	Wind <dbl>	Temp <int>	Month <int>	Day <int>
96	167	6.9	91	9	1
78	197	5.1	92	9	2
73	183	2.8	93	9	3
91	189	4.6	93	9	4
47	95	7.4	87	9	5
32	92	15.5	84	9	6
20	252	10.9	80	9	7
23	220	10.3	78	9	8
21	230	10.9	75	9	9
24	259	9.7	73	9	10

1-10 of 153 rows Previous **1** 2 3 4 5 6 … 16 Next

mutate Modificamos los valores de una variable.

En este caso creamos una nueva variable a partir de otra: pasamos la variable temperatura, que estaba en grados Fahrenheit, a grados Celsius.

Hide

```
mutate(airquality, TempInC = (Temp - 32) * 5 / 9) # Creamos la variable TempInC a partir de la variable Temp
```

Ozone <int>	Solar.R <int>	Wind <dbl>	Temp <int>	Month <int>	Day <int>	TempInC <dbl>
41	190	7.4	67	5	1	19.44444
36	118	8.0	72	5	2	22.22222
12	149	12.6	74	5	3	23.33333
18	313	11.5	62	5	4	16.66667
NA	*NA*	14.3	56	5	5	13.33333
28	*NA*	14.9	66	5	6	18.88889
23	299	8.6	65	5	7	18.33333
19	99	13.8	59	5	8	15.00000
8	19	20.1	61	5	9	16.11111
NA	194	8.6	69	5	10	20.55556

1-10 of 153 rows Previous **1** 2 3 4 5 6 … 16 Next

summarise Calcula características estadísticas sobre variables.

Hide

```
summarise(airquality, mean(Temp, na.rm = TRUE)) # calcula la media de la temperatu
ra ignorando valores faltantes
```

mean(Temp, na.rm = TRUE) <dbl>
77.88235

1 row

group_by En multitud de ocasiones buscaremos obtener datos agregados según los valores que tome cierta variable.

Los datos agregados serán resultado de ejercer operaciones sobre el resto de operaciones. Por ejemplo, podemos agrupar por la variable Month y tomar la media de la temperatura para cada mes utilizando la función summarise

Hide

```
summarise(group_by(airquality, Month), mean(Temp, na.rm = TRUE)) # agrupa por mes
y calcula la temperatura media en dicho mes
```

Month <int>	mean(Temp, na.rm = TRUE) <dbl>
5	65.54839
6	79.10000
7	83.90323
8	83.96774
9	76.90000

5 rows

left_join Nos permite enlazar varias tablas de manera horizontal (i.e. a nivel de registros).

Supongamos que tenemos una tabla “master” donde nos viene el ID de una persona junto a su nombre y también tenemos una tabla “colores” con el ID de la persona y su color favorito.

Si cruzamos la tabla “master” con la tabla “colores” por la variable ID, obtendremos una tabla con el ID, el nombre y el color favorito de la persona.

Creamos los dos data.frame:

Hide

```
IDs <- c("0001", "0002", "0003")
nombres <- c("Álvaro", "Miriam", "Laura")
colores_fav <- c("Azul marino", "Lila", "Azul turquesa")

master <- data.frame(IDs, nombres)
colores <- data.frame(IDs, colores_fav)

master
```

IDs <chr>	nombres <chr>
0001	Álvaro
0002	Miriam
0003	Laura

3 rows

Hide

```
colores
```

IDs <chr>	colores_fav <chr>
0001	Azul marino
0002	Lila
0003	Azul turquesa

3 rows

Ahora, realizaremos un join de la tabla master con la tabla colores:

Hide

```
left_join(master, colores, by="IDs")
```

IDs <chr>	nombres <chr>	colores_fav <chr>
0001	Álvaro	Azul marino
0002	Miriam	Lila
0003	Laura	Azul turquesa

3 rows

Más funciones La librería dplyr cuenta con múltiples funciones. Una función de uso muy habitual es **count**.

Hide

```
count(airquality, Month) #cuenta el número de registros para cada valor del variab
le Month
```

Month <int>	n <int>
5	31
6	30
7	31
8	31
9	30

5 rows

Con el operador **pipe** `%>%` se pueden hacer las transformaciones anteriores en forma de secuencia, esto es una sintaxis muy cómoda para trabajar sobre las tablas

Hide

```
airquality %>%
    filter(Month != 5) %>% # filtra por meses distintos de 5
    group_by(Month) %>% # agrupa por mes
    summarise(mean(Temp, na.rm = TRUE)) # calcula la media
```

Month <int>	mean(Temp, na.rm = TRUE) <dbl>
6	79.10000
7	83.90323
8	83.96774
9	76.90000

4 rows