Workshop: paquete dplyr

Analytics Research Lab

Introducción

El presente documento tiene como objetivo ayudarte a mejorar tus habilidades en el software Rstudio. Es una guía practica que te impulsará a desarrollar ejercicios de manera autonoma. A continuación encontrarás las principales funciones contenidas en el apquete dplyr y sus argumentos. Inicialmente debes instalar el paquete: install.packages("dplyr") y luego cargarlo: library(dplyr). El paquete dplyr permite manipular y operar data frames. Entre las principales funciones de este paquete podemos encontrar: *filter* que nos permite seleccionar filas, *select* nos ayuda a seleccionar columnas de interés, *arrange* nos permite ordenar, entre otras.

```
library(dplyr)
```

Es posible que cuando instales y cargues el paquete, aparezcan waring messages en color rojo, esto es debido a que el paquete tiene funciones con el mismo nombre en otros paquetes. Si tienes dudas sobre algunas de las funciones, puedes intentar lo siguiente con cada función y obtener más información

```
?select
```

Principales funciones del paquete dplyr

Select

La función *select* te permite seleccionar columnas específicas de tu base de datos y te devuelve un nuevo data frame. Para este ejemplo, lo primero que haremos es instalar y cargar el paquete con el data frame que vamos a utilizar

```
library(devtools)
```

Una vez instalado y cargado el paquete devtools, podemos observar la base de datos. Esta base de datos contiene información sobre tormentas, el año, dia, hora, ubicación, etc. escribiendo el nombre de la base de datos storms[1:2,] le estamos pidiendo que nos muestre las 2 primeras observaciones y todas las variables. Es importante recordar que las filas son observaciones y las columnas son variables.

```
storms[1:2,]
```

Vamos a seleccionar unicamente las variables name, que hace referencia al nombre de la tormenta y pressure. Utilizamos la función *select* y el primer argumento será el nombre del objeto donde están nuestros datos, en este caso, storms, luego, separado de comas, indicamos las columnas que queremos obtener.

```r
select(storms, name, pressure)[1:2,]
```

```
## # A tibble: 2 x 2
##   name  pressure
##   <chr>    <int>
## 1 Amy       1013
## 2 Amy       1013
```

En caso de queramos excluir columnas de nuestro data frame, podemos utilizar un *guión* - y si queremos sleccionar un rango, podemos utilizar *dos puntos* : Por ejemplo, si ahora deseo eliminar las columnas desde category hasta hu_diameter:

```r
select(storms,-(category:hu_diameter))[1:2,]
```

```
## # A tibble: 2 x 8
##   name   year month   day  hour   lat  long status
##   <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>
## 1 Amy    1975     6    27     0  27.5   -79 tropical depression
## 2 Amy    1975     6    27     6  28.5   -79 tropical depression
```

Otra de las funciones que pueden facilitar el trabajo con la función select es *starts_with*, la cual nos permite seleccionar columnas teniendo en cuenta la letra con la que inicie su nombre:

```r
select(storms,starts_with("w"))[1:2,]
```

```
## # A tibble: 2 x 1
##    wind
##   <int>
## 1    25
## 2    25
```

Así mismo, podemos usar la función *ends_with()* para indicarle que seleccione las variables cuyo nombre termina con una letra indicada.

Filter

Esta función nos permite filtrar observaciones según una condición. Por ejemplo, si ahora quiero obtener unicamente las observaciones en las que el viento alcanzó una velocidad mayor o igual a 50, utilizo la función *filter()* cuyos argumentos son el nombre del objeto que contiene los datos y la condición.

```r
filter(storms,wind>=50)[1:2,]
```

```
## # A tibble: 2 x 13
##   name   year month   day  hour   lat  long status category  wind pressure
##   <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>  <ord>    <int>    <int>
## 1 Amy    1975     6    29    18  33.8 -72.8 tropi~ 0           50      998
## 2 Amy    1975     6    30     0  34.3 -71.6 tropi~ 0           50      998
## # ... with 2 more variables: ts_diameter <dbl>, hu_diameter <dbl>
```

Podemos incluir varias condiciones en la función: Por ejemplo, si ahora queremos aquellas observaciones de Alberto, Alex y Allison donde el viento es mayor o igual a 50 utilizamos %in% que significa "pertenece al conjunto"

```
filter(storms, wind >= 50, name %in% c("Alberto", "Alex", "Allison"))[1:2,]
```

```
## # A tibble: 2 x 13
##   name   year month   day  hour   lat  long status category  wind pressure
##   <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>  <ord>    <int>    <int>
## 1 Albe~  1982     6     3    12  23.2 -84.2 tropi~ 0           50      995
## 2 Albe~  1982     6     3    18  24   -83.6 hurri~ 1           75      985
## # ... with 2 more variables: ts_diameter <dbl>, hu_diameter <dbl>
```

Otra forma de filtrar varias condiciones es utilizar *&*. Por ejemplo, si quiero unicamente observaciones con el viento mayor o igual a 50 y una presión menor a 1010, obtenemos:

```
filter(storms, wind>=50 & pressure<1010)[1:2,]
```

```
## # A tibble: 2 x 13
##   name   year month   day  hour   lat  long status category  wind pressure
##   <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>  <ord>    <int>    <int>
## 1 Amy    1975     6    29    18  33.8 -72.8 tropi~ 0           50      998
## 2 Amy    1975     6    30     0  34.3 -71.6 tropi~ 0           50      998
## # ... with 2 more variables: ts_diameter <dbl>, hu_diameter <dbl>
```

Operador pipe %>%

Este operador permite realizar varias operaciones del paquete dplyr de manera consecutiva. Por ejemplo, nos permite combinar la función filter y select sin necesidad de hacerlas por separado. Si quiero obtener el nombre de aquellas tormentas en las que el viento fue mayor o igual a 50

```
(storms %>%
  filter(wind>=50) %>%
  select(name,wind))[1:2,]
```

```
## # A tibble: 2 x 2
##   name   wind
##   <chr> <int>
## 1 Amy      50
## 2 Amy      50
```

Función Arrange

Esta función nos permite ordenar las observaciones de una base de datos, teniendo en cuenta una o varias variables. Por defecto, ordena de forma ascendente. En caso de que quiera ordenar de manera descendente, debo agregar desc(). Si por ejemplo quiero ordenarlas teniendo en cuenta el viento:

```
arrange(storms, wind)[1:2,]
```

```
## # A tibble: 2 x 13
##   name   year month   day  hour   lat  long status category  wind pressure
##   <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>  <ord>    <int>    <int>
## 1 Bonn~  1986     6    28     6  36.5 -91.3 tropi~ -1          10     1013
## 2 Bonn~  1986     6    28    12  37.2 -90   tropi~ -1          10     1012
## # ... with 2 more variables: ts_diameter <dbl>, hu_diameter <dbl>
```

Con esta función también podemos ordenar teniendo en cuenta varias variables. Por ejemplo, si ahora quiero ordenar ascendente según el viento, pero me interesa la presión descentende:

```
arrange(storms, wind,desc(pressure))[1:2,]
```

```
## # A tibble: 2 x 13
##   name   year month   day  hour   lat  long status category  wind pressure
##   <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>  <ord>    <int>    <int>
## 1 AL03~  1987     8    17     0  31.4 -82.9 tropi~ -1          10     1015
## 2 AL03~  1987     8    17     6  31.8 -82.3 tropi~ -1          10     1015
## # ... with 2 more variables: ts_diameter <dbl>, hu_diameter <dbl>
```

A continuación, para trabajar las ultimas funciones claves del paquete **dplyr**, utilizaremos la base de datos " vehicle" del paquete *fueleconomy*, para cargar los datos realizamos lo siguiente:

```
install.packages("fueleconomy")
```

```
library(fueleconomy)
vehicles <- fueleconomy::vehicles
```

La base de datos cuenta con información sobre el rendimiento de combustible de ciertos vehiculos, cuenta con aproximadamente 34,000 datos. Observamos los tipos de variables:

```
glimpse(vehicles)
```

Las variables de interés serán el numero de cilindros (cyl), la cilindrada o cilindraje del vehiculo (displ), las millas por galón en autopista (hwy) y en ciudad (cty). Para mayor información de la base de datos y sus variables:

```
?vehicles
```

Continuando con las funciones.

Mutate

Esta función nos permite crear variables nuevas en función de las ya existentes, tambien permite modificar variables de un data set. Si queremos construir una nueva variable de cilindraje (displ) medida en mililitros hariamos lo siguiente:

```
(vehicles %>% mutate("displ_ml"=displ*1000) %>% select(1,6,displ, displ_ml))[1:3,]
```

```
## # A tibble: 3 x 4
##      id trans           displ displ_ml
##   <dbl> <chr>           <dbl>    <dbl>
## 1 13309 Automatic 4-spd   2.2     2200
## 2 13310 Manual 5-spd      2.2     2200
## 3 13311 Automatic 4-spd   3       3000
```

Ahora, si deseamos convertir las variables hwy y cty de millas por galón a **kilometros por litro**, realizariamos:

```
(vehicles %>% mutate("hwy"=hwy*0.425, "cty"=cty*0.425) %>% select(1,6,hwy, cty))[1:3,]
```

```
## # A tibble: 3 x 4
##      id trans             hwy   cty
##   <dbl> <chr>           <dbl> <dbl>
## 1 13309 Automatic 4-spd  11.0  8.5
## 2 13310 Manual 5-spd     11.9  9.35
## 3 13311 Automatic 4-spd  11.0  7.65
```

Si deseamos que las variables nuevas se creen en un nuevo data set, podemos utilizar de la misma forma la función ***trasnmute()***

```
(vehicles %>% transmute("hwy"=hwy*0.425, "cty"=cty*0.425))[1:3,]
```

Summarise

Esta función permite **resumir** todo el data set en una o mas variables de interes, utilizando alguna función en particular. Algunas funciones de interes para resumir un data set son la media, mediana, quantiles, maximos, minimos, etc. Si deseamos hallar el valor minimo, maximo y el promedio de millas por galón en autopista para todos los vehiculos, realizamos:

```
(vehicles %>% summarise("Mínimo"=min(hwy, na.rm = T),"Media"=mean(hwy, na.rm = T), "Máximo"=max(hwy, na
```

```
## # A tibble: 1 x 3
##   Mínimo Media Máximo
##    <dbl> <dbl>  <dbl>
## 1      9  23.6    109
```

```
# el parametro na.rm=T omite los valores faltantes de la variable
```

group_by

Todas las funciones que se han mecionado, se pueden acompañar de group_by() para realizar analisis mas complejos. La función crea grupos en el data set dependiendo de las variables que se escojan. Por ejemplo, si deseamos agrupar los vehiculos que tengan el mismo numero de cilindros y luego hallar la frecuencia absoluta de estos grupos, al igual que la respectiva mediana de la variable *displ*, realizamos lo siguiente:

```
(vehicles %>% group_by(cyl) %>% summarise("Frecuencia absoluta"=n(),
                                          "Mediana"=median(displ, na.rm = T)))
```

```
## # A tibble: 10 x 3
##      cyl `Frecuencia absoluta` Mediana
##    <dbl>                 <int>   <dbl>
##  1     2                    45     1.3
##  2     3                   182     1
##  3     4                 12381     2
##  4     5                   718     2.5
##  5     6                 11885     3.4
##  6     8                  7550     5.2
##  7    10                   138     5.2
##  8    12                   478     5.9
##  9    16                     7     8
## 10    NA                    58     1.3
```

Conclusión y recomendaciones

Para terminar, el paquete dplyr ofrece una gama de funciones que nos permiten domar los datos, es decir, estudiarlos, modificarlos, interpretarlos y obtener conclusiones a partir de estos. Si deseamos obtener resultados cada vez mas especificos y utiles, es primordial utilizar las funciones en conjunto y de manera secuencial. El operador pipe "%>%" y la función group_by(), son aspectos que al dominarlos nos permitirán obtener mejores hallazgos. Es recomendable practicar las funciones aqui presentadas, pueden obtener bases de datos para practicar en el paquete *datasets* (para mas info del paquete revisar: https://stat.ethz.ch/R-manual/R-patched/library/datasets/html/00Index.html). Recomendamos **realizar los problemas sugeridos**.

. # **Probemas sugeridos**

1. (storms) Crea un nuevo data frame que muestre el nombre y la velocidad del viento de aquellas tormentas que ocurrieron del año 1974 a 1976 y fueron clasificadas como depresión tropical.
2. (storms) Organizar el data set teniendo en cuenta la velocidad del viento, de mayor a menor.
3. (storms) Crea una nueva variable categorica, la cual, dependiendo de la hora del dia, tenga el valor "Mañana" (0-12h) o "Tarde" (12-24h). **Pista:** Usar la función if_else() del paquete dplyr
4. (vehicles) Hallar los cuartiles de la variable cty. **Pista:** Realizar un "resumen" del data set y utilizar la función quantiles().
5. (vehicles) Excluyendo los vehiculos de años anteriores a 2005, agrupar dependiendo de la clase del vehiculo y del numero de cilindros. Hallar el valor maximo de cilindraje (displ) para cada grupo y ordenar de manera descendente.

La respuesta del punto 5. debe verse se la siguiente manera:

```
## # A tibble: 105 x 3
## # Groups:   class [24]
##    class                       cyl Cilindraje_max
##    <chr>                     <dbl>          <dbl>
##  1 Two Seaters                  10            8.4
##  2 Standard Pickup Trucks 2WD   10            8.3
##  3 Two Seaters                  16            8
##  4 Compact Cars                  8            7
##  5 Two Seaters                   8            7
##  6 Large Cars                    8            6.8
##  7 Midsize Cars                  8            6.8
```

```
##  8 Vans, Cargo Type              10            6.8
##  9 Vans, Passenger Type          10            6.8
## 10 Compact Cars                  12            6.7
## # ... with 95 more rows
```

Bibliografía recomendada

libro programación en R: https://rsanchezs.gitbooks.io/rprogramming/content/chapter9/arrange.html

Libro R for data science: https://es.r4ds.hadley.nz/transform.html

Página Web: https://swcarpentry.github.io/r-novice-gapminder-es/13-dplyr/