

Introducción a dplyr

Armando Ocampo

dplyr

dplyr es una librería que nos permite manipular datos, ordenarlos y extraer información necesaria. Esta paquetería no forma parte de las paqueterías base de R. Por lo que tenemos que descargarla.

```
install.packages("dplyr")
```

Una vez instalada, la llamaremos para comenzar a trabajar.

```
library(dplyr)
```

dplyr pertenece a un conjunto de paqueterías llamado tidyverse, el cual nos permite manipular información. Es bueno que sepas esto, no obstante, el día de hoy solo trabajaremos con la paquetería mencionada.

Las funciones principales de dplyr son `filter()`, `arrange()`, `mutate()`, `group_by()` y `summarise()`. También se conocen como verbos. Además existe un elemento denominado pipe " `%>%` " que nos permite unir dos elementos y trabajar con ellos.

filter()

El primer verbo que utilizaremos se llama `filter()` y nos permite filtrar un conjunto de datos con una característica en específico. Los ejemplos mostrados en este artículo se basaran en información de los data set *iris* y *mtcars*. Antes de comenzar, haremos un resumen de los elementos que contiene cada data set.

```
str(iris)
summary(iris)
glimpse(iris)

str(mtcars)
summary(mtcars)
glimpse(mtcars)
```

Nota__ `glimpse()` es una función de dplyr que permite observar un resumen de la información presente en el data frame de trabajo.

Comenzaremos a trabajar con el dataset de *iris*. De este data set nos quedaremos solo con las especies setosas. En este caso dentro de la función `filter()` se coloca el nombre de la variable y la condición a seguir.

Nota 1: el nombre de la variable debe coincidir con la forma en la cual está escrito en el df.

Nota 2: en caso de realizar una condición con elementos de tipo caracter, este va en comillas. Las comparaciones numéricas van sin comillas

Nota 3: el "pipe" `%>%` , nos permite tomar el objeto de la izquierda y ejercer un verbo o una acción en específico. Esta última, se escribe del lado derecho.

```
iris %>%
  filter(Species == "setosa")

# al igual que en ejercicios previos, al correr esta parte del código
# el resultado se observará en la consola. Si queremos guardar el resultado
# debemos asignarlo a un objeto

df_setosas <- iris %>%
  filter(Species == "setosa")
```

Una de las ventajas de filter es realizar varios filtros en una sola función. A continuación filtraremos las iris de tipo setosas con longitud de sepalo mayor a 4.8. Solo debemos separar cada condición con una coma

```
iris %>%
  filter(Species == "setosa",
         Sepal.Length > 4.8)
```

Siempre que respetemos el uso de comas, podemos realizar tantos filtros como nosotros queramos. Sin correr el siguiente código. ¿Qué crees que filtraría el siguiente argumento?

```
iris %>%
  filter(Species == "virginica",
         Petal.Length > 4.5,
         Petal.Width < 2,
         Sepal.Length > 6,
         Sepal.Width < 2.8)
```

En los ejemplos previos realizamos comparaciones con mayor y menor que e igualdades. No obstante, ¿Cómo puedo hacer una comprativa con rangos? Por ejemplo, quiero obtener las iris versicolor con una longitud de petalo entre 3.5 y 4.8. Para este caso, realizamos comparaciones con los booleanos | OR, & AND o != NOT. Para ello colocamos la primer condición del lado izquierdo el booleano en el centro y la segunda condición a la derecha.

```
iris %>%
  filter(Species == "versicolor",
         Petal.Length > 3.5 & Petal.Length < 4.8)
```

Esta regla se debe seguir cuando utilices los booleanos. Además, hay que recordar que no hay una sola respuesta correcta en R. Como ejemplo filtraremos las especies de iris para generar un df que contenga solo setosas y virginicas a partir de diferentes códigos.

```
iris %>%
  filter(Species == "setosa" | Species == "virginica")

iris %>%
  filter(Species != "versicolor")

iris %>%
  filter(Species %in% c("setosa", "versicolor"))
```

Nota: Para realizar un filtro a partir de un vector de tipo caracter se utiliza %in% en lugar de ==

TAREA con el dataset mtcars responde lo siguiente 1. ¿Cuántos automóviles tienen solo 4 cilindros? 2. ¿Cuántos automóviles tienen motor con forma de V y desplazamiento mayor a 150? 3. ¿Qué automóviles tienen 4 carburadores, 6 cilindros, transmisión manual y menos de 200 caballos de fuerza?

arrange()

El verbo `arrange()` permite ordenar los datos de forma ascendente a partir de una variable. A continuación, ordenaremos los datos de iris a partir de la longitud de sépalo

```
iris

iris %>%
  arrange(Sepal.Length)
```

Para ordenar de forma descendente se agrega la función `desc()` dentro del verbo `arrange()`. Ahora ordenaremos de forma descendente los datos de iris a partir de la longitud de pétalo

```
iris

iris %>%
  arrange(desc(Petal.Length))
```

El pipe `%>%` permite combinar varios verbos de `dplyr`. De esta forma podemos realizar varios verbos y manipular los datos acorde a lo que necesitamos.

Nota: cada verbo se separa con un pipe

Vamos a filtrar los datos para obtener solo las especies `versicolor` y aquellas iris con longitud de pétalo entre 3.5 y 4.8. Por último, ordenaremos los datos de forma descendente a partir del ancho del pétalo.

```
iris %>%
  filter(Species == "versicolor",
         Petal.Length > 3.5 & Petal.Length < 4.8) %>%
  arrange(desc(Petal.Width))
```

Nota: los pipes también permiten separar el código en acciones individuales. Puedes subrayar ciertas partes de tu código y evaluar cada paso. Esto es útil cuando hay algún error en tu script.

mutate()

El verbo `mutate()` genera una nueva variable a partir de una expresión de nuestros datos. Este verbo toma como argumento el nombre de la nueva variable en el lado izquierdo y del lado derecho la expresión de interés. Como ejemplo, crearemos una variable denominada `multiplicación`. La cual se conformará de multiplicar la longitud de sépalo por 5.

Nota 1: recuerda que cada variable de un dataset puede utilizarse como un vector independiente. Y que cada vector puede utilizarse en una función matemática.

Nota 2: recuerda que para asignar un objeto en R puedes utilizar `<-` o `=`. En `mutate()` asignamos a la nueva variable solo con `=`

```
iris

iris %>%
  mutate(multiplicacion = Sepal.Length * 5)

# Este resultado solo estará presente en la consola. Para guardarlo deberás
# asignarlo a un objeto
```