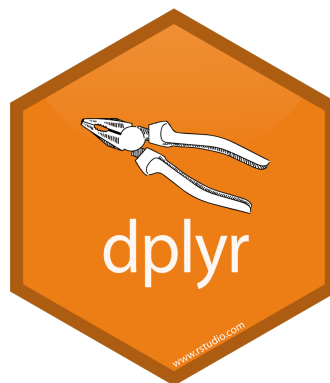


GUÍA DPLYR

José Manuel Pinzón Mendivelso



UNIVERSIDAD DE
LA SALLE



Facultad de Economía, Empresa y Desarrollo Sostenible
Universidad de La Salle
Bogotá, Colombia
2021

Índice

Paquete Dplyr	2
Preparar el paquete	2
Cargar y ajustar bases de datos	2
Generalidades.	4
select.	5
Extraer columnas.	5
Eliminar columnas.	5
distinct.	6
count.	6
arrange.	7
Ordenar por valores.	7
Ordenar alfabéticamente.	7
filter.	9
Filtro para una variable con datos tipo caracter.	9
Filtro para una variable de tipo numérico.	9
Filtro con varias condiciones.	10
mutate.	13
group_by & summarize.	14
Ejercicios.	15

NOTA ACLARATORIA: Esta guía aún se encuentra en construcción, no es la versión final, razón por la cual está sujeta a modificaciones. **Esta guía es exclusivamente para uso pedagógico**

Paquete Dplyr

En esta guía aprenderá a manipular datos por medio del paquete *dplyr*. Este paquete incluye funciones que permiten filtrar, segmentar, resumir y transformar datos. Dplyr también permite escribir el código de forma indentada por medio del operador “pipe” (`%>%`) (Se explicará mientras se desarrolla la guía), permitiendo que el código resulte mucho más legible y fácil de entender, además que permite concatenar varias funciones o comandos en un sólo código. Tenga en cuenta que las funciones de esta librería siempre devolverán un objeto o estructura de dato de tipo `DataFrame`.

En esta guía se explicarán las siguientes funciones:

- **select:** Devuelve un conjunto de columnas.
- **distinct:** Indica los valores o niveles de su variable.
- **count:** Cuenta el número de observaciones que hay en cada grupo o nivel de la variable.
- **arrange:** Reordena filas de un `DataFrame`.
- **filter:** Devuelve un conjunto de filas según una o varias condiciones lógicas.
- **mutate:** Añade nuevas variables/columnas por medio de operaciones matemáticas.
- **summarise/summarize:** Genera resúmenes de diferentes variables en el data frame.
- **group_by:** Agrupa valores del dataframe según las columnas seleccionadas.
- **rename:** Renombra variables en una `DataFrame`.

Para el desarrollo de esta guía se va a descargar la información directamente desde internet, para ello solo se necesita tener los links de descarga en un formato que R permita importar. En este caso, se usarán archivos `.csv`.

Preparar el paquete

En primera instancia, tiene que instalar el paquete, recuerde que puede hacerlo desde la pestaña *Packages* o puede hacerlo haciendo uso del código `install.packages()`

```
install.packages("dplyr") #Recuerde que solo se instala la primera vez.
```

Tenga en cuenta que siempre que se va a trabajar con un paquete que no viene por defecto con R hay que llamarlo para poder usarlo. Para llamarlo haga uso del comando `library`.

```
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
```

Cargar y ajustar bases de datos

Los datos con los cuales se va a trabajar se pueden descargar de los siguientes enlaces:

- https://covid.ourworldindata.org/data/ecdc/full_data.csv
- <https://covid.ourworldindata.org/data/ecdc/locations.csv>.

Si usted quiere cargar directamente la información a R puede hacerlo empleando el comando `read.csv()`, en los paréntesis deberá colocar el link de descarga entre comillas. A continuación encontrará la descripción de las variables y la carga de las bases de datos.

El DataFrame **Reporte_C** está compuesto por 59.354 observaciones y 10 variables:

- **date:** Fecha del reporte.
- **location:** País.
- **new_cases:** Nuevos casos positivos de COVID reportados en esa fecha.
- **new_deaths:** Nuevas muertes por COVID reportados en esa fecha.
- **total_cases:** Casos totales positivos de COVID reportados hasta esa fecha.
- **total_deaths:** Muertes totales a causa de COVID reportados hasta esa fecha.
- **weekly_cases:** Casos totales positivos de COVID reportados en esa semana.
- **weekly_deaths:** Muertes totales a causa de COVID reportados en esa semana.

```
Reporte_C <- read.csv(
  "https://covid.ourworldindata.org/data/ecdc/full_data.csv") #Cargar base de datos.
str(Reporte_C) #Características DF.
```

```
## 'data.frame':    59354 obs. of  10 variables:
## $ date          : chr  "2019-12-31" "2020-01-01" "2020-01-02" "2020-01-03" ...
## $ location      : chr  "Afghanistan" "Afghanistan" "Afghanistan" "Afghanistan" ...
## $ new_cases     : int   0 0 0 0 0 0 0 0 0 0 ...
## $ new_deaths    : int   0 0 0 0 0 0 0 0 0 0 ...
## $ total_cases   : int  NA NA NA NA NA NA NA NA NA NA ...
## $ total_deaths  : int  NA NA NA NA NA NA NA NA NA NA ...
## $ weekly_cases  : num   NA NA NA NA NA NA 0 0 0 0 ...
## $ weekly_deaths : num   NA NA NA NA NA NA 0 0 0 0 ...
## $ biweekly_cases : num   NA NA NA NA NA NA NA NA NA NA ...
## $ biweekly_deaths : num   NA NA NA NA NA NA NA NA NA NA ...
```

También se tiene la base de datos **Info_P**, la cual está compuesta por 214 observaciones y 5 variables que contienen información sobre los países.

- **countriesAndTerritories:** Territorio.
- **location:** País.
- **continent:** Continente.
- **population_year:** Año.
- **population:** Población.

```
Info_P <- read.csv(
  "https://covid.ourworldindata.org/data/ecdc/locations.csv") #Cargar base de datos.
glimpse(Info_P) #Características DF.
```

```
## Rows: 214
## Columns: 5
## $ countriesAndTerritories <chr> "Afghanistan", "Albania", "Algeria", "Andorra"~
## $ location               <chr> "Afghanistan", "Albania", "Algeria", "Andorra"~
## $ continent              <chr> "Asia", "Europe", "Africa", "Europe", "Africa"~
## $ population_year        <int> 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020~
## $ population             <int> 38928341, 2877800, 43851043, 77265, 32866268, ~
```

Ahora se van a unificar las bases de datos, con el fin que poder tener toda la información resumida en un DF. Esto se logra empleando la función **merge**. En primer lugar se deben colocar los archivos a unificar, posteriormente se debe indicar la variable en común sobre la cual se debe unificar toda la información. Para este caso, note que los dos DF tienen la variable *location*, por lo tanto, se indicará a R que tome en cuenta esta variable por medio del argumento *by*.

```
COVID=merge(Reporte_C,Info_P, by="location") #Unificar.
glimpse(COVID) #Características COVID.
```

```
## Rows: 59,019
## Columns: 14
## $ location      <chr> "Afghanistan", "Afghanistan", "Afghanistan", "~
## $ date          <chr> "2020-10-21", "2020-10-22", "2020-10-23", "202~
## $ new_cases     <int> 88, 135, 116, 61, 81, 65, 199, 113, 0, 123, 15~
## $ new_deaths    <int> 2, 2, 4, 2, 4, 3, 8, 7, 0, 3, 4, 0, 5, 3, 4, 6~
## $ total_cases   <int> 40375, 40510, 40626, 40687, 40768, 40833, 4103~
## $ total_deaths  <int> 1499, 1501, 1505, 1507, 1511, 1514, 1522, 1529~
## $ weekly_cases  <dbl> 381, 484, 600, 614, 627, 633, 745, 770, 635, 6~
## $ weekly_deaths <dbl> 19, 20, 24, 22, 23, 22, 25, 30, 28, 27, 29, 25~
## $ biweekly_cases <dbl> 827, 894, 933, 984, 1065, 1034, 1104, 1151, 11~
## $ biweekly_deaths <dbl> 30, 31, 33, 34, 38, 37, 42, 49, 48, 51, 51, 48~
## $ countriesAndTerritories <chr> "Afghanistan", "Afghanistan", "Afghanistan", "~
## $ continent     <chr> "Asia", "Asia", "Asia", "Asia", "Asia", "Asia"~
## $ population_year <int> 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020~
## $ population    <int> 38928341, 38928341, 38928341, 38928341, 389283~
class(COVID$date) #Tipo de dato variable date.
```

```
## [1] "character"
```

Ahora se tiene toda la información en el objeto **COVID**.

Note que las fechas (Variable date) se encuentran en formato carácter, las fechas se pueden manipular mejor cuando se trabajan como tipo **date**, para hacer el cambio se usa el comando **as.Date** de la siguiente manera.

```
COVID$date = as.Date(COVID$date) #Cambiar tipo de datos.
class(COVID$date) #Tipo de dato variable date.
```

```
## [1] "Date"
```

Generalidades.

- Hay dos formas para utilizar las funciones que hacen parte de *dplyr*.
 - La primera forma es escribiendo el código normal. En donde se recibe como primera entrada de la función el nombre del Data Frame a manipular, y las siguientes son opciones propias de cada comando.
 - La segunda manera es escribir el código de manera indentada, haciendo uso del operador *pipe* (`%>%`), el cual permite escribir varios comandos de manera vertical en un sólo código. Para este caso, en primera instancia se indica la base de datos que se va a manipular, luego se coloca el *pipe* (`%>%`), y por último se colocan las funciones a usar.
- El objeto de retorno es un nuevo data frame.
- Los data frames deben estar bien organizados/estructurados, es decir, cada columna debe representar una variable y cada fila al menos una observación.

Nota: Para facilidad y mejor entendimiento de los códigos se recomienda usar pipes, sin embargo en esta guía se explicarán de las dos maneras

select.

El comando `select()` permite eliminar o extraer las variables especificadas del DataFrame original. A continuación se harán las dos maneras de las que se habló anteriormente.

Extraer columnas.

Para seleccionar las columnas que se quieren trabajar simplemente tiene que mencionarlas separandolas por comas, observe los siguientes códigos y sus resultados.

```
## Forma 1
base_1 <- select(COVID, location,date,new_cases) #Primero la base de datos COVID, luego las columnas.
head(base_1,2) #Resultado
```

```
##      location      date new_cases
## 1 Afghanistan 2020-10-21        88
## 2 Afghanistan 2020-10-22       135
```

```
## Forma 2
base_1= COVID %>% #Base de datos.
  select(location,date,new_deaths,total_deaths,population)#Función y columnas
head(base_1,2) #Resultado.
```

```
##      location      date new_deaths total_deaths population
## 1 Afghanistan 2020-10-21         2         1499   38928341
## 2 Afghanistan 2020-10-22         2         1501   38928341
```

Eliminar columnas.

Si usted desea eliminar algunas columnas el procedimiento es practicamente el mismo al anterior, con la única diferencia es que para este caso se debe agregar el símbolo menos (-) antes de las variables o columnas a eliminar.

```
## Forma 1
base_2 <- select(COVID, -date, -new_cases, -new_deaths, -biweekly_cases, -biweekly_deaths,
  -countriesAndTerritories, -population_year, -continent )
head(base_2,2) #Resultado
```

```
##      location total_cases total_deaths weekly_cases weekly_deaths population
## 1 Afghanistan    40375         1499         381         19   38928341
## 2 Afghanistan    40510         1501         484         20   38928341
```

```
## Forma 2
base_2 <- COVID%>%
  select(-total_cases, -new_cases, -new_deaths, -biweekly_cases, -biweekly_deaths,
    -weekly_cases, -weekly_deaths, -population_year, -countriesAndTerritories)
head(base_2,2) #Resultado
```

```
##      location      date total_deaths continent population
## 1 Afghanistan 2020-10-21         1499      Asia   38928341
## 2 Afghanistan 2020-10-22         1501      Asia   38928341
```

distinct.

La función `distinct` permite identificar cuales son los valores u observaciones con los que cuenta cada variable. Para hacer uso de esta solo necesita especificar la base de datos y la variable.

#Forma 1

```
distinct(COVID, continent)
```

```
##      continent
## 1      Asia
## 2     Europe
## 3     Africa
## 4 North America
## 5 South America
## 6     Oceania
## 7
```

#Forma2

```
COVID %>%
  distinct(continent)
```

```
##      continent
## 1      Asia
## 2     Europe
## 3     Africa
## 4 North America
## 5 South America
## 6     Oceania
## 7
```

count.

Este comando es similar a `distinct`, con la diferencia de que este cuenta el número de veces que se repite esa observación.

#Forma 1

```
tail(count(COVID, location),3)
```

```
##      location      n
## 212     Yemen 234
## 213    Zambia 256
## 214  Zimbabwe 254
```

#Forma 2

```
N_Paises <- COVID%>%
  count(location)
```

```
head(N_Paises,3)
```

```
##      location      n
## 1 Afghanistan 335
## 2     Albania 266
## 3     Algeria 335
```

arrange.

Ordena el DataFrame dependiendo de los valores de alguna columna en específico. Estos valores se pueden ordenar de forma numérica o alfabética.

Como la base de datos cuenta con muchas variables se seleccionaran unas pocas para que se vea mejor al imprimir los resultados. Se van a combinar los comandos **select** y **arrange**.

Ordenar por valores.

Por defecto, la función **arrange** organiza de menor a mayor, si usted necesita la base de datos ordenada de mayor a menor agregue un signo menos antes de la variable.

A continuación encontrará ejemplos en donde se ordenará el DataFrame dependiendo de los valores de la variable *new_cases*.

#Forma 1 - Sin usar pipe. Variable new_cases de menor a mayor.

```
arrange_1 <- arrange(select(COVID, location, date, new_cases, total_cases), new_cases)

head(arrange_1, 4)
```

```
##      location      date new_cases total_cases
## 1    Ecuador 2020-09-07     -8261      109784
## 2    Ecuador 2020-05-07     -2461       29420
## 3    Ecuador 2020-05-09     -1480       28818
## 4 Luxembourg 2020-08-28     -1385        6543
```

#Forma 2 - Usando pipe. Variable new_cases de mayor a menor.

```
arrange_2 <- COVID %>%
  select(location, date, new_cases, total_cases) %>%
  arrange(-new_cases)

head(arrange_2, 4)
```

```
##      location      date new_cases total_cases
## 1 United States 2020-11-28    207913    13091758
## 2 United States 2020-11-21    196117    11913944
## 3 United States 2020-11-20    188020    11717827
## 4 United States 2020-11-26    186589    12777754
```

En caso de que haya un empate, es decir, observaciones con un mismo valor, usted puede definir otra variable para hacer el “Desempate” y que se ordene según la otra variable que se definió. Para ello simplemente escriba el nombre de la columna del desempate como tercer argumento para la forma 1 o como segundo argumento de la forma 2 de la función **arrange**.

En la siguiente sección se explicará un ejemplo para cuando haya un empate al ordenar alfabéticamente. Funcionan de la misma manera.

Ordenar alfabéticamente.

También es posible ordenar el DataFrame de manera alfabética, se hace de la misma manera que en el anterior ejercicio. Pero para este caso, las observaciones de las variables categóricas se repiten mucho, por lo que se van a resolver los empates de acuerdo con una segunda variable de la manera que se mencionó anteriormente.

En el siguiente ejemplo se ordenará el DF dependiendo de la variable *location* de manera alfabética, cuando se repita el país se ordenará con la variable *new_deaths*.

#Forma 1. Sin usar pipe. Ordenando con las variables location y new_deaths.

```
Alfa <- arrange(select(COVID,location,date,new_deaths,total_deaths), location, -new_deaths)
```

```
head(Alfa,4)
```

##	location	date	new_deaths	total_deaths
## 1	Afghanistan	2020-07-07	56	920
## 2	Afghanistan	2020-07-16	49	1094
## 3	Afghanistan	2020-06-19	42	546
## 4	Afghanistan	2020-07-06	38	864

#Forma 2. Usando pipe. Ordenando con las variables location y new_deaths

```
Alfa2 <- COVID %>%  
  select(location,date,new_deaths,total_deaths)%>%  
  arrange(location,-new_deaths)  
head(Alfa2,4)
```

##	location	date	new_deaths	total_deaths
## 1	Afghanistan	2020-07-07	56	920
## 2	Afghanistan	2020-07-16	49	1094
## 3	Afghanistan	2020-06-19	42	546
## 4	Afghanistan	2020-07-06	38	864

filter.

Así como la función `select` es utilizada para seleccionar columnas, la función **filter** hace lo propio con las filas del `DataFrame`, se filtran las observaciones que cumplan con las condiciones especificadas.

Esta función trabaja con condiciones lógicas, en donde se utilizan los operadores lógicos, algunos de ellos son:

- `<` (Menor que)
- `<=` (Menor o igual que)
- `>` (Mayor que)
- `>=` (Mayor o igual que)
- `==` (Igual a).

En primera instancia se especifica la base de datos, luego la variable en la cual se quiere buscar el dato y por último la condición. Cuando usted quiere buscar en una variable que tenga formato carácter es necesario que ponga entre comillas el valor objetivo.

Filtro para una variable con datos tipo carácter.

A continuación verá dos ejemplos, en el primero se filtrará la información de Colombia y en el segundo la de Alemania.

Con el fin de que los resultados se impriman de una manera más agradable para el lector, se volverá a utilizar el comando **select** en los ejemplos para seleccionar unas pocas variables.

##Forma 1

```
Colombia <- filter(select(COVID,location,date,total_cases),location == "Colombia")
head(Colombia,3)
```

```
##  location      date total_cases
## 1 Colombia 2020-08-21      513719
## 2 Colombia 2020-08-20      502178
## 3 Colombia 2020-07-24      226373
```

##Forma 2

```
Colombia2 <- COVID%>%
  select(location, date, total_cases)%>%
  filter(location == "Germany")
head(Colombia2,3)
```

```
##  location      date total_cases
## 1 Germany 2020-06-14      186269
## 2 Germany 2020-06-15      186461
## 3 Germany 2020-06-11      185416
```

Filtro para una variable de tipo numérico.

Ahora se mostrarán 2 ejemplos para casos numéricos. En primer lugar se hará un filtro para las observaciones que tengan un valor mayor a 18.000 en la variable `new_cases`. Para el segundo caso, se seleccionarán las variables que tengan un valor menor a 200 en la variable `new_cases`.

#Forma 1

```
Fil_num <- filter(select(COVID,location,date,new_cases), new_cases > 18000)
head(Fil_num)
```

```
##      location      date new_cases
## 1 Argentina 2020-10-22    18326
## 2  Belgium 2020-10-21    19265
## 3  Belgium 2020-10-24    18876
## 4  Belgium 2020-10-27    19868
## 5  Belgium 2020-10-28    22210
## 6   Brazil 2020-11-28    34130
```

#Forma 2

```
Fil_num2 <- COVID%>%
  select(location, date, new_cases)%>%
  filter(new_cases < 200)
head(Fil_num2,3)
```

```
##      location      date new_cases
## 1 Afghanistan 2020-10-21      88
## 2 Afghanistan 2020-10-22     135
## 3 Afghanistan 2020-10-23     116
```

Filtro con varias condiciones.

En este comando se pueden filtrar por varias condiciones al tiempo, en estos casos se usa &(y) cuando se tienen que cumplir estrictamente todas las condiciones especificadas y |(o) cuando quiere filtrar las filas que cumplan con al menos criterio.

Filtro con | (o) Para filtrar datos de una sola variable. Para los siguientes ejemplos se les agregará la función **distinct**, con el fin de que se visualice de mejor manera los datos que se han filtrado.

#Forma 1

```
Países1 <-select(filter(COVID, location %in% c("Colombia", "Vanuatu", "Belgium")),
  location,date,new_cases) #Sin distinct
head(Países1,3)
```

```
##      location      date new_cases
## 1  Belgium 2020-05-05      670
## 2  Belgium 2020-05-06      547
## 3  Belgium 2020-05-07      552
```

```
Países1 <-distinct(select(filter(COVID, location %in% c("Colombia", "Vanuatu", "Belgium")),
  ,location,date,new_cases),location) #Con distinct
head(Países1)
```

```
##      location
## 1  Belgium
## 2 Colombia
## 3  Vanuatu
```

Note cuando se agrega la función **distinct** se muestra que a nivel de la variable *location* solo aparecen los países que se filtraron en el código.

Forma 2

```
Países2 = COVID%>%
  select(location,date,new_cases,new_deaths,total_deaths,population)%>%
  filter(location == "Germany" | location == "Colombia" | location == "France")
```

```
head(Paises2,3)
```

```
##   location      date new_cases new_deaths total_deaths population
## 1 Colombia 2020-08-21   11541      204      16183   50882884
## 2 Colombia 2020-08-20   13056      360      15979   50882884
## 3 Colombia 2020-07-24    7945      315      7688   50882884
```

```
Paises2 = COVID%>%
  select(location,date,new_cases,new_deaths,total_deaths,population)%>%
  filter(location == "Germany" | location == "Colombia" | location == "France")%>%
  distinct(location)
head(Paises2)
```

```
##   location
## 1 Colombia
## 2  France
## 3  Germany
```

Ahora se van a filtrar datos de dos o más variables. Para el primer caso, se filtrará toda la información del país Andorra y toda la información existente para la fecha 2020-10-05. En el segundo caso se filtrará la información de Alemania, además los datos que tengan “South America” en la variable “continent”.

Recuerde que el operador |(o) filtra las observaciones que cumplan con una de las condiciones especificadas.

#Forma 1

```
Andorra_Date <-select(filter(COVID, location == "Andorra" | date == "2020-10-05"),
  location,date,new_cases)

head(Andorra_Date)
```

```
##   location      date new_cases
## 1 Afghanistan 2020-10-05      44
## 2  Albania 2020-10-05     149
## 3  Algeria 2020-10-05     141
## 4  Andorra 2020-06-04       7
## 5  Andorra 2020-06-05       1
## 6  Andorra 2020-06-06       0
```

#Forma 2

```
Alemania_Continent = COVID%>%
  filter(location == "Germany" | continent == "South America")%>%
  select(location,date,new_deaths,total_deaths,continent)

head(Alemania_Continent,4)
```

```
##   location      date new_deaths total_deaths  continent
## 1 Argentina 2020-08-24      53      6848 South America
## 2 Argentina 2020-08-25     231      7079 South America
## 3 Argentina 2020-08-15     182      5428 South America
## 4 Argentina 2020-08-16     137      5565 South America
```

Note que para el ejemplo de Andorra se muestran solo una vez Afganistan, Albania y Algeria, por lo que solo hay una observación de esos países para la fecha especificada. Mientras que Andorra aparece varias veces porque se filtró toda la información de este país, sin importar su fecha.

Para mostrar que los datos se filtraron correctamente en el ejemplo de Alemania y America del sur, se va a aplicar la función **distinct** sobre la variable *continent*.

```
Continent_dis = COVID%>%
  filter(location == "Germany" | continent == "South America")%>%
  select(location,date,new_deaths,total_deaths,continent)%>%
  distinct(continent)
print(Continent_dis)
```

```
##          continent
## 1 South America
## 2          Europe
```

Observe que ahora solo hay 2 continentes, “South America” que fue el dato filtrado de la variable *continent* y “Europe”, que es el continente del país filtrado (“Germany”).

Filtro con & (y) Ahora se realizarán ejemplos con el operador **&**, el cual se toma como una “y”.

A continuación se harán dos ejemplos, en el primero se van a filtrar los datos del continente “Oceania” para la fecha 2020-06-14. Para el segundo ejemplo se filtrarán las observaciones que contengan el dato “South America” en la variable *continent* para la fecha 2020-05-20 y que además el dato de la variable *new_cases* sea mayor a 3000.

Tenga presente que el operador & solo filtra las filas que cumplen estrictamente con todas las condiciones.

```
# Forma 1
America2 <- select(filter(COVID,date == "2020-06-14" & continent == "Oceania"), location,
                    date,new_cases, new_deaths, population)
head(America2)
```

```
##          location      date new_cases new_deaths population
## 1      Australia 2020-06-14        12          0   25499881
## 2           Fiji 2020-06-14          0          0    896444
## 3 French Polynesia 2020-06-14          0          0    280904
## 4           Guam 2020-06-14          0          0    168783
## 5 New Caledonia 2020-06-14          0          0    285491
## 6    New Zealand 2020-06-14          0          0   4822233
```

```
# Forma2
America3 <- COVID %>%
  filter(date == "2020-05-20" & continent == "South America" & new_cases >= 3000)%>%
  select(location,date,new_cases, new_deaths, population)
head(America3)
```

```
##    location      date new_cases new_deaths population
## 1   Brazil 2020-05-20    17408     1179  212559409
## 2    Chile 2020-05-20     3520         31   19116209
## 3     Peru 2020-05-20     4550        125   32971846
```

mutate.

Ahora se explicará como computar transformaciones de variables en un DataFrame por medio de la función **mutate**. Esta puede ser una herramienta muy útil, ya que algunas veces se tiene la necesidad de hacer nuevos calculos a partir de otras variables u otros datos ya existentes en la base de datos, esta función facilita realizar este tipo de operaciones.

A continuación observará un ejemplo de como hacer uso de esta función, en donde se creará el porcentaje de la población que se ha contagiado de covid en cada país, para ello se dividirán el total de los casos (*total_cases*) por la población (*population*). Luego de esto se seleccionarán algunas columnas con la función **select** y posteriormente se ordenará el DataFrame de mayor a menor con los datos de la variable *Nueva_V*.

Forma 1

```
Porcentaje <- arrange(select(mutate(COVID, Nueva_V = (total_cases/population)*100), location,
                             date, total_cases, population, Nueva_V),-Nueva_V)
head(Porcentaje,3)
```

```
##   location      date total_cases population  Nueva_V
## 1 Andorra 2020-11-29      6670      77265 8.632628
## 2 Andorra 2020-11-28      6610      77265 8.554973
## 3 Andorra 2020-11-27      6534      77265 8.456610
```

Forma 2

```
Porcentaje <- COVID%>%
  mutate(Nueva_V = (total_cases/population)*100)%>%
  select(location, date, total_cases, population, Nueva_V)%>%
  arrange(-Nueva_V)
head(Porcentaje,3)
```

```
##   location      date total_cases population  Nueva_V
## 1 Andorra 2020-11-29      6670      77265 8.632628
## 2 Andorra 2020-11-28      6610      77265 8.554973
## 3 Andorra 2020-11-27      6534      77265 8.456610
```

Con el comando **mutate()** también se pueden encadenar la creación de varias variables en una misma sentencia o código.

Ahora se realizará un ejemplo en donde se crearán dos variables nuevas, una con el porcentaje total de personas muertas sobre la población (*Porcentaje_1*) y otro con el porcentaje de personas muertas sobre el total de las personas contagiadas (*Porcentaje_2*) para la última fecha de la base de datos. Sobre esta base de datos se seleccionarán las variables *location*, *date*, *Porcentaje1*, *Porcentaje2* y luego se ordenará mediante la segunda variable creada.

```
COVID_Ultimo_dia = COVID %>%
  filter(date == tail(COVID$date,1)) %>%
  mutate(Porcentaje1=(total_deaths/population)*100,
         Porcentaje2 = (total_deaths/total_cases)*100)%>%
  select(location,date,Porcentaje1,Porcentaje2)%>%
  arrange(-Porcentaje2)
head(COVID_Ultimo_dia,3)
```

```
##   location      date Porcentaje1 Porcentaje2
## 1   Yemen 2020-11-29 0.002061962  28.472222
## 2  Mexico 2020-11-29 0.081793801   9.581233
## 3 Montserrat 2020-11-29 0.020004001   7.692308
```

group_by & summarize.

Para finalizar se explicará el uso de las funciones `group_by` y `summarize` o `summarise` (Son prácticamente lo mismo), usar estos dos comandos juntos se complementan muy bien al momento de querer resumir datos. Primero se explicará la funcionalidad de cada comando y luego se harán ejemplos de como usarlos juntos.

1. **group_by:** Este comando agrupa los valores de todas las observaciones de una variable o columna. Usted puede agrupar los datos con más de una variable.
2. **summarise:** Se usa principalmente para calcular estadísticas de la base de datos. Crea un nuevo dataframe específicamente con los datos a resumir.

Ahora que tiene una idea de como funcionan estos comandos se va a hacer un ejercicio en el cual se agrupará la información por continentes y luego se determinará la media o promedio y varianza de los nuevos casos por día para cada uno de los continentes.

Como el dataframe con tiene espacios vacíos se hará uso del argumento `na.omit` para omitir estos datos vacíos y que no hayan inconvenientes para calcular las estadísticas.

```
COVID %>%
  group_by(continent) %>%
  summarize(media_continente = mean(na.omit(new_cases)),
            Varianza = var(na.omit(new_cases)))
```

```
## # A tibble: 7 x 3
##   continent      media_continente  Varianza
##   <chr>          <dbl>         <dbl>
## 1 ""            10.9           758.
## 2 "Africa"       152.          498724.
## 3 "Asia"        1189.         42098194.
## 4 "Europe"      1109.         15133595.
## 5 "North America" 1644.        112014269.
## 6 "Oceania"      22.1           6694.
## 7 "South America" 3127.        63217476.
```

Ejercicios.

1. Haga un código que seleccione las variables location, date, population y new_cases
2. Escriba un código que filtre los datos de Colombia y de Brasil.
3. Realice un código que cuente el número de datos que hay para cada continente.
4. Construya un df llamado P_Oceania con los países de Oceanía.
5. Construya un df que contenga todos los países de América (Note que América está dividida en la variable *continent*)
6. Cree una nueva base de datos con las variables location, date, total_cases, total_deaths, filtre toda la información de Colombia y ordénela por fecha.
7. Hallar el porcentaje de casos nuevos para el día 30 de agosto de 2020.
8. Determine el porcentaje de nuevas muertes por continente el día 8 de septiembre de 2020.
9. Construya un df con la media y la desviación estándar de la variable new_deaths para cada día.
10. Haga un código que filtre la información de los continentes África, Asia y Europa, luego determine el promedio de nuevos casos para cada uno de los continentes y luego ordénelo de mayor a menor según ese promedio.
11. Investigue que hacen las funciones *sample_n* y *sample_frac*.

```
COVID%>%
  filter(continent %in% c("Africa", "Asia", "Europe"))%>%
  group_by(continent)%>%
  summarise(Promedio = mean(na.omit(new_cases)))%>%
  arrange(-Promedio)
```

```
## # A tibble: 3 x 2
##   continent Promedio
##   <chr>         <dbl>
## 1 Asia          1189.
## 2 Europe        1109.
## 3 Africa         152.
```