

Introducción a R y Tidyverse

Sesión 04

Laboratorio de Innovación en Salud

2022-06-04

 @healthinnovation
 @innovalab_imt
 innovalab.info

Contenidos

- Selección de variables de interés
- Uso de condicionales para filtrar casos
- Aspectos adicionales sobre la creación de variables
- Solicitar resumen de datos (summarise)
- Considerar variables para subdividir análisis (group_by)

Exploración competencial

Uso de filter()

¿Para qué sirve? Ejemplo

- Ayuda a crear un subconjunto de datos con todas las filas que cumplan tus condiciones.
- Pueden incluirse varias condiciones dentro de un filtro.
- Funciones y operadores útiles:
 - De comparación: `==`, `<`, `>`, `≤`, `≥`, `≠`,
`%in%`, `is.na`, `!is.na`
 - De lógica: `&`, `|`, `xor`, `!`, `any()`, `all()`

```
library(tidyverse)
nycflights13 :: flights
```

```
## # A tibble: 336,776 × 19
##       year month   day dep_time
##       <int> <int> <int>     <int>
## 1 2013     1     1      1     517
## 2 2013     1     1      1     533
## 3 2013     1     1      1     542
## 4 2013     1     1      1     544
## 5 2013     1     1      1     554
## 6 2013     1     1      1     554
## 7 2013     1     1      1     555
## 8 2013     1     1      1     557
```

Uso de filter()

¿Para qué sirve?

Ejemplo

Esta data muestra información de la NOAA acerca de tormentas desde 1975 hasta el 2020.

```
storms %>%
  count(status, category)

## # A tibble: 8 × 3
##   status          category     n
##   <chr>            <ord>    <int>
## 1 hurricane        1        1933
## 2 hurricane        2         749
## 3 hurricane        3         434
## 4 hurricane        4         411
## 5 hurricane        5          86
## 6 tropical depression -1       2898
```

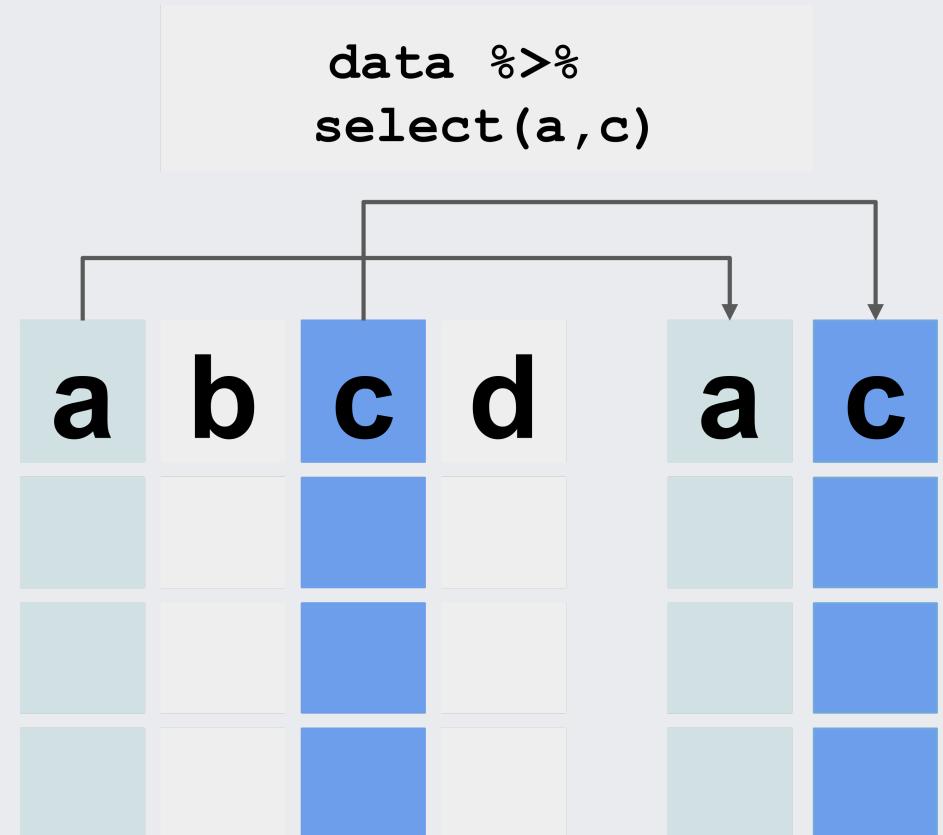
Si quisieramos solo trabajar con los huracanes de categoría 5, tendríamos que hacer lo siguiente con la función `filter()`:

```
storms %>%
  filter(status = "hurricane",
         category = 5)

## # A tibble: 86 × 13
##   name      year month day hour lat
##   <chr>    <dbl> <dbl> <int> <dbl> <dbl>
## 1 Anita    1977    9     2     0  24.6
## 2 Anita    1977    9     2     6  24.2
## 3 David    1979    8     30    6  16
## 4 David    1979    8     30   12  16.3
```

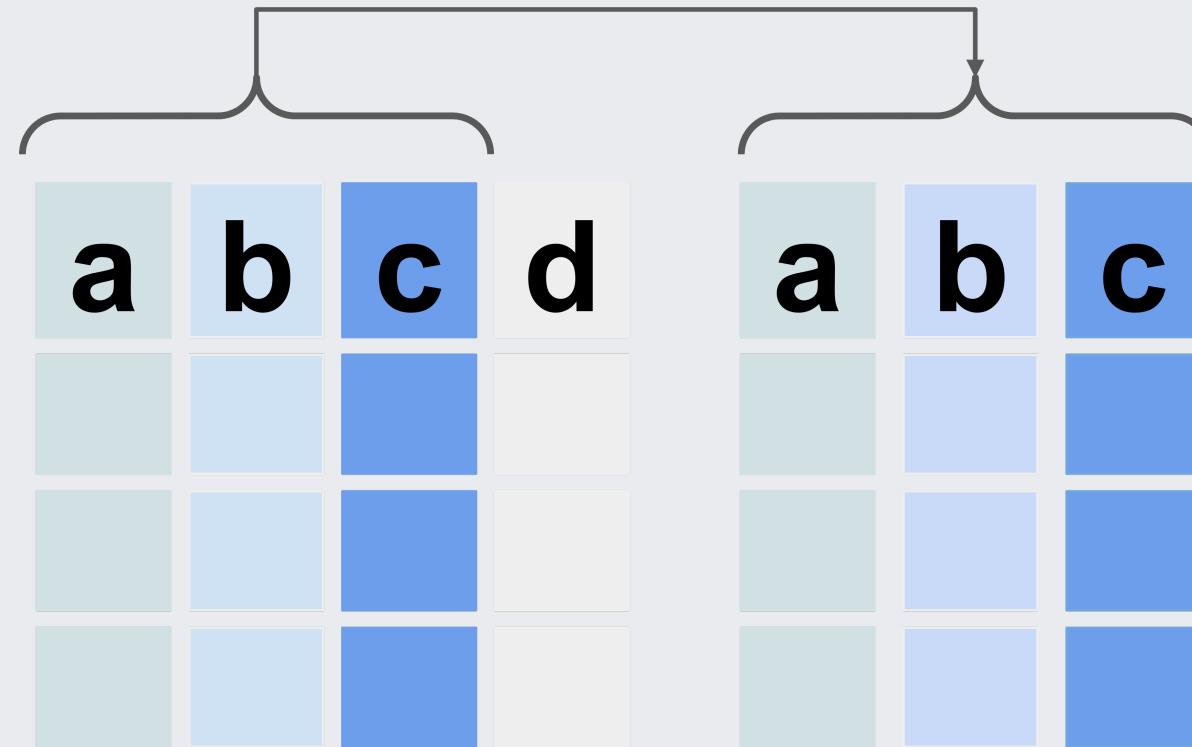
Uso de select()

- Esta función permite **seleccionar** las variables de interés.
- Puede ser útil previo a un análisis de datos o solo si se desea tener una data menos larga.
- Para usarlo se puede nombrar a las variables dentro de la función `select()` y/o usar el comando `:` para indicar **hasta**.



Uso de select()

```
data %>%  
  select(a:c)
```



Uso de select()

El uso de las funciones con **pipe (%>%)** permite anidarlas una tras otra. Además, si las columnas están una a lado de otra, podemos usar : para seleccionar varios al mismo tiempo.

```
storms %>%
  filter(status = "hurricane",
         category = 5) %>%
  select(name, year, month, day, hour, wind)
```

```
## # A tibble: 86 × 6
##   name    year  month  day  hour  wind
##   <chr>  <dbl> <dbl> <int> <dbl> <int>
## 1 Anita   1977     9     2     0    140
## 2 Anita   1977     9     2     6    150
## 3 David   1979     8    30     6    140
## 4 David   1979     8    30    12    145
## 5 David   1979     8    30    18    150
## 6 David   1979     8    31     0    145
## 7 David   1979     8    31     6    145
## ...
```

```
storms %>%
  filter(status = "hurricane",
         category = 5) %>%
  select(name:hour, wind)
```

```
## # A tibble: 86 × 6
##   name    year  month  day  hour  wind
##   <chr>  <dbl> <dbl> <int> <dbl> <int>
## 1 Anita   1977     9     2     0    140
## 2 Anita   1977     9     2     6    150
## 3 David   1979     8    30     6    140
## 4 David   1979     8    30    12    145
## 5 David   1979     8    30    18    150
## 6 David   1979     8    31     0    145
## 7 David   1979     8    31     6    145
## ...
```

¡Hazlo tú mismo!



Para este ejercicio usaremos la ENDES 2020. Específicamente el módulo de cuestionario individual para mujeres de 12 a 49 años (RECH94). Sobre este módulo se requiere lo siguiente:

1. Importar el archivo .sav que se encuentra dentro de la carpeta `data/` en un objeto llamado `rech94`.
2. Usando el archivo PDF llamado `Diccionario - REC94.pdf` dentro de la carpeta `data/`, selecciona las siguientes variables: Identificador del cuestionario, meses de embarazo en última revisión prenatal, razón por la que no acudió a un hospital para dar a luz, prueba de anemia durante embarazo, diagnóstico de anemia, tratamiento con hierro y consumo del mismo.
3. De forma similar con el archivo `RECH1.sav`, importarlo dentro de R.
4. Usando el documento `Diccionario - REC1.pdf`, filtrar únicamente los casos de mujeres que tengan edades entre los 12 y 49 años.

10 : 00

Mutate II

dplyr::
mutate()



Esta función del paquete **dplyr** nos permite agregar o modificar las variables actuales en un conjunto de datos (dataframe). Tal y como ya lo vimos en la [sesión 03](#), esta función puede hacer cambios sobre una o múltiples variables existentes, así como crear varias a la vez separándolas con una coma (Y. Wendy Huynh, 2019).

```
data %>%  
  mutate(  
    var1 = old_var*2,  
    var2 = old_var*3,  
    var3 = var1*var2  
)
```

Mutate II: Uso de across()

Haremos la prueba con la base `storm`. Usando la función `across()` para aplicar una función a múltiples variables a la vez. En esta ocasión transformaremos en **factores** a las variables `status` y `category` de tormentas.

```
storms %>%
  select(name:pressure) %>%
  mutate(
    status = as.factor(status),
    category = as.factor(category)
  ) %>%
  glimpse()

## #> #> #> #>
```

```
storms %>%
  select(name:pressure) %>%
  mutate(
    across(c(status, category), as.factor)
  ) %>%
  glimpse()

## #> #> #> #>
```

```
## #> #> #> #>
```

Mutate II: lubridate

Muchas veces durante el pre-procesamiento de datos tenemos que enfrentarnos a manejo de fechas (crear, transformar, etc.). Para estos fines podemos usar las funciones del paquete **lubridate**.

Para tener una mejor apreciación de la variable de fecha que crearemos, usaremos **relocate()** para reposicionar a la variable creada.

```
storms %>%  
  select(name:pressure)
```

	## # A tibble: 11,859 × 11
	## 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
## 2	Amy 1975 6 27 6 28.5 -79 tropical
## 3	Amy 1975 6 27 12 29.5 -79 tropical
## 4	Amy 1975 6 27 18 30.5 -79 tropical
## 5	Amy 1975 6 28 0 31.5 -78.8 tropical
## 6	Amy 1975 6 28 6 32.4 -78.7 tropical
## 7	Amy 1975 6 28 12 33.3 -78 tropical

Mutate II: lubridate

Muchas veces durante el pre-procesamiento de datos tenemos que enfrentarnos a manejo de fechas (crear, transformar, etc.). Para estos fines podemos usar las funciones del paquete **lubridate**.

Para tener una mejor apreciación de la variable de fecha que crearemos, usaremos **relocate()** para reposicionar a la variable creada.

```
storms %>%  
  select(name:pressure) %>%  
  mutate(  
    date = lubridate::make_date(year,  
  ))
```

	## # A tibble: 11,859 × 12
	## 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
## 2	Amy 1975 6 27 6 28.5 -79 tropical
## 3	Amy 1975 6 27 12 29.5 -79 tropical
## 4	Amy 1975 6 27 18 30.5 -79 tropical
## 5	Amy 1975 6 28 0 31.5 -78.8 tropical
## 6	Amy 1975 6 28 6 32.4 -78.7 tropical
## 7	Amy 1975 6 28 12 33.3 -78 tropical

Mutate II: lubridate

Muchas veces durante el pre-procesamiento de datos tenemos que enfrentarnos a manejo de fechas (crear, transformar, etc.). Para estos fines podemos usar las funciones del paquete **lubridate**.

Para tener una mejor apreciación de la variable de fecha que crearemos, usaremos **relocate()** para reposicionar a la variable creada.

```
storms %>%  
  select(name:pressure) %>%  
  mutate(  
    date = lubridate::make_date(year,  
  ) %>%  
  relocate(date, .before = year)
```

	## # A tibble: 11,859 × 12
	## name date year month day hour lat lon
	## <chr> <date> <dbl> <dbl> <int> <dbl> <dbl> <dbl>
1	## 1 Amy 1975-06-27 1975 6 27 0 27.5 -79
2	## 2 Amy 1975-06-27 1975 6 27 6 28.5 -79
3	## 3 Amy 1975-06-27 1975 6 27 12 29.5 -79
4	## 4 Amy 1975-06-27 1975 6 27 18 30.5 -79
5	## 5 Amy 1975-06-28 1975 6 28 0 31.5 -78.
6	## 6 Amy 1975-06-28 1975 6 28 6 32.4 -78.
7	## 7 Amy 1975-06-28 1975 6 28 12 33.3 -78

Uso de group_by()

Esta función permite poder indicarle a R que las operaciones que hagamos en adelante, sean divididas y aplicadas por las agrupaciones que sea haya formado. Es decir, que se puede hacer la agrupación por una o múltiples variables.

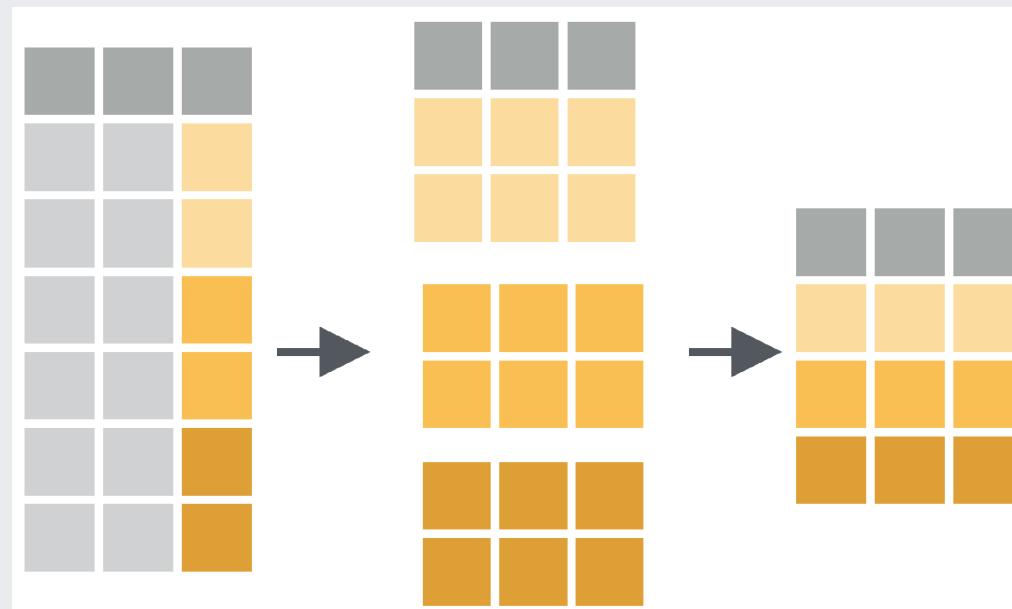
```
storms %>%  
  filter(year >= 2010) %>%  
  group_by(name)  
  
## # A tibble: 3,608 × 13  
## # Groups:   name [112]  
##       name   year month   day hour   lat  
##   <chr> <dbl> <dbl> <int> <dbl> <dbl>  
## 1 Alex    2010     6    25    18    16.4  
## 2 Alex    2010     6    26     0    16.6  
## 3 Alex    2010     6    26     6    16.7  
## 4 Alex    2010     6    26    12    16.9  
## 5 Alex    2010     6    26    18    17.2
```

La función `group_by()` solo configura como será el tratamiento de la data en adelante. No observará ningún cambio perceptible en los datos con excepción de:

```
# Groups:   name [112]
```

Uso de summarise()

Esta función permite obtener algún resumen de los datos, como la media, desviación estándar, valores máximos, mínimos, calculos de prueba estadística, y la mayoría de funciones de esta naturaleza. El uso conjunto de `group_by()` y `summarise()` es una combinación perfecta.



```
mtcars %>%  
  group_by(cyl) %>%  
  summarise(  
    avg = mean(mpg)  
  )
```

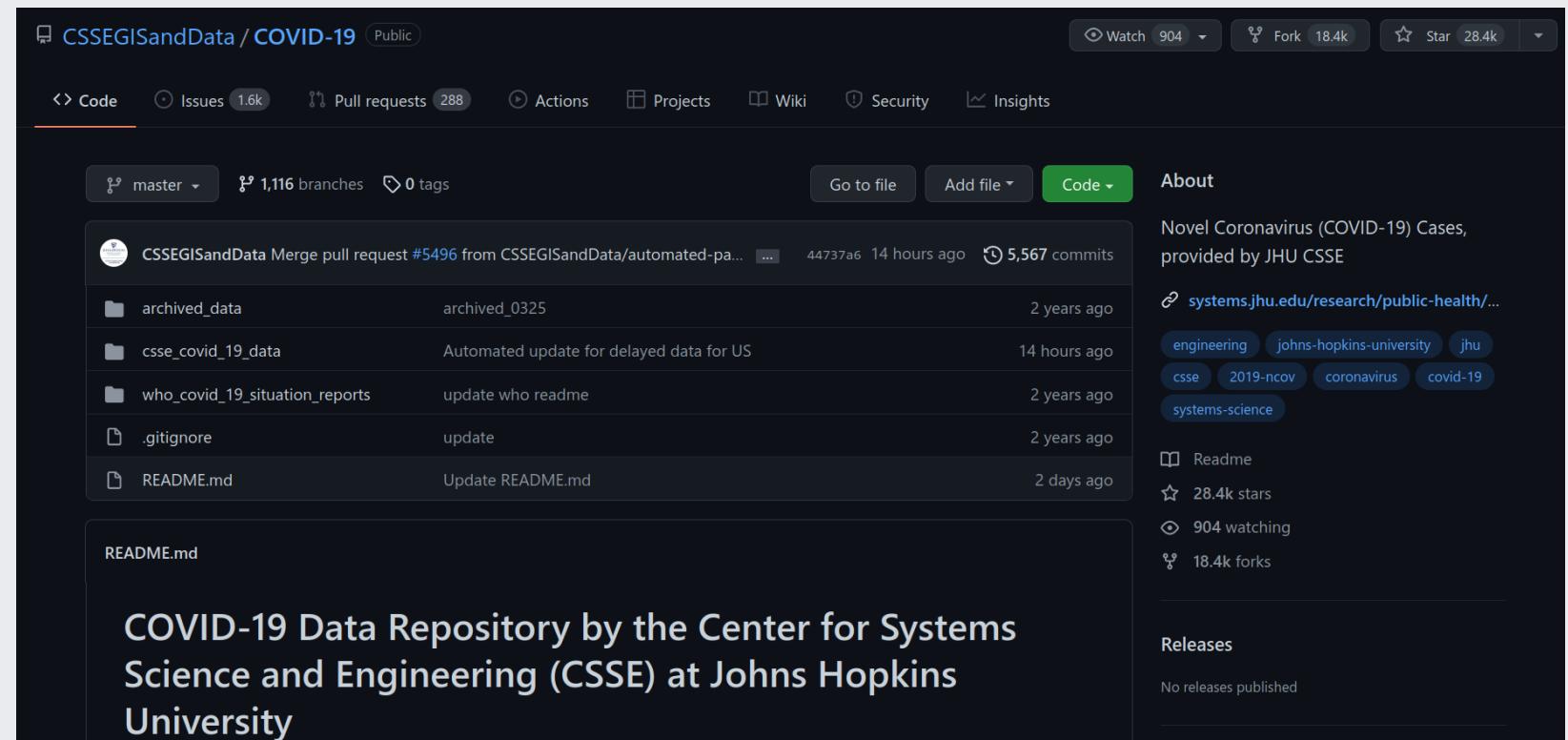
Data transformation with dplyr:: CHEAT SHEET

Uso de summarise()

Intro Importación

Usaremos la misma BD de COVID-19 que empleamos en la sesión 03.

Este dataset está alojada en el siguiente repositorio de GitHub ([click aquí](#)).



The screenshot shows the GitHub repository page for CSSEGISandData/COVID-19. The repository has 904 watchers, 18.4k forks, and 28.4k stars. It contains 1,116 branches and 0 tags. The master branch is selected. The repository description is: "Novel Coronavirus (COVID-19) Cases, provided by JHU CSSE". The README.md file states: "COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University".

Uso de summarise()

Intro Importación

Mediante la función `read_csv` podremos importar la data directamente desde **Github**.

```
covid19 ← read_csv("https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/covid19.csv")  
  
## # A tibble: 285 × 868  
##   `Province/State` `Country/Region`   Lat   Long `1/22/20` `1/23/20` `1/24/20` `1/25/20`  
##   <chr>           <chr>       <dbl>  <dbl>     <dbl>     <dbl>     <dbl>     <dbl>  
## 1 <NA>            Afghanistan  33.9  67.7      0        0        0        0  
## 2 <NA>            Albania     41.2  20.2      0        0        0        0  
## 3 <NA>            Algeria    28.0  1.66      0        0        0        0  
## 4 <NA>            Andorra    42.5  1.52      0        0        0        0  
## 5 <NA>            Angola     -11.2 17.9      0        0        0        0  
## 6 <NA>            Antarctica -71.9  23.3      0        0        0        0  
## 7 <NA>            Antigua and Barbados 17.1 -61.8      0        0        0        0
```

Uso de summarise()

Recordar que primero debemos *pivotear* la data para poder trabajar con ella.
Obtendremos resúmenes de casos por muerte COVID-19 por países.

```
covid19      ## # A tibble: 285 × 868
#>   `Province/State`    `Country/Region`     Lat    Lon
#>   <chr>                <chr>            <dbl>   <dbl>
#> 1 <NA>                Afghanistan       33.9   67.7
#> 2 <NA>                Albania          41.2   20.2
#> 3 <NA>                Algeria          28.0   1.6
#> 4 <NA>                Andorra          42.5   1.5
#> 5 <NA>                Angola           -11.2  17.9
#> 6 <NA>                Antarctica      -71.9  23.3
#> 7 <NA>                Antigua and Bar... 17.1  -61.8
#> 8 <NA>                Argentina        -38.4  -63.6
#> 9 <NA>                Armenia          40.1   45.0
#> 10 Australian Capital Te... Australia      -35.5  149.
#> # ... with 275 more rows, and 864 more variables:
```

Uso de summarise()

Recordar que primero debemos *pivotear* la data para poder trabajar con ella. Obtendremos resúmenes de casos por muerte COVID-19 por países.

```
covid19 %>%
  select(-c(Lat:Long)) %>%
  pivot_longer(
    cols = -c(`Province/State`:`Count`),
    names_to = "fecha",
    values_to = "fallecidos"
  )
```

A tibble: 246,240 × 4
`Province/State` `Country/Region` fecha fallecidos
<chr> <chr> <chr> <dbl>
1 <NA> Afghanistan 1/22/20 0
2 <NA> Afghanistan 1/23/20 0
3 <NA> Afghanistan 1/24/20 0
4 <NA> Afghanistan 1/25/20 0
5 <NA> Afghanistan 1/26/20 0
6 <NA> Afghanistan 1/27/20 0
7 <NA> Afghanistan 1/28/20 0
8 <NA> Afghanistan 1/29/20 0
9 <NA> Afghanistan 1/30/20 0
10 <NA> Afghanistan 1/31/20 0
... with 246,230 more rows

Uso de summarise()

Recordar que primero debemos *pivotear* la data para poder trabajar con ella.
 Obtenremos resúmenes de casos por muerte COVID-19 por países.

```
covid19 %>%
  select(-c(Lat:Long)) %>%
  pivot_longer(
    cols = -c(`Province/State`:`Count`),
    names_to = "fecha",
    values_to = "fallecidos"
  ) %>%
  group_by(`Country/Region`)
  #> # A tibble: 246,240 × 4
  #> # Groups:   Country/Region [199]
  #> #   `Province/State` `Country/Region` fecha   fallecido
  #> #   <chr>          <chr>       <chr>     <dbl>
  #> #   1 <NA>          Afghanistan 1/22/20
  #> #   2 <NA>          Afghanistan 1/23/20
  #> #   3 <NA>          Afghanistan 1/24/20
  #> #   4 <NA>          Afghanistan 1/25/20
  #> #   5 <NA>          Afghanistan 1/26/20
  #> #   6 <NA>          Afghanistan 1/27/20
  #> #   7 <NA>          Afghanistan 1/28/20
  #> #   8 <NA>          Afghanistan 1/29/20
  #> #   9 <NA>          Afghanistan 1/30/20
  #> #  10 <NA>          Afghanistan 1/31/20
  # ... 
```

Uso de summarise()

Recordar que primero debemos *pivotear* la data para poder trabajar con ella.
 Obtenremos resúmenes de casos por muerte COVID-19 por países.

```
covid19 %>%
  select(-c(Lat:Long)) %>%
  pivot_longer(
    cols = -c(`Province/State`:`Count`),
    names_to = "fecha",
    values_to = "fallecidos"
  ) %>%
  group_by(`Country/Region`) %>%
  summarise(
    M = mean(fallecidos, na.rm = TRUE),
    DE = sd(fallecidos, na.rm = TRUE),
    Max = max(fallecidos, na.rm = TRUE),
    Min = min(fallecidos, na.rm = TRUE)
  )
## # A tibble: 199 × 5
##   `Country/Region`      M     DE     Max     Min
##   <chr>          <dbl>  <dbl>  <dbl>  <dbl>
## 1 Afghanistan     3754.  2962.  7708     0
## 2 Albania        1723.  1314.  3497     0
## 3 Algeria         3451.  2332.  6875     0
## 4 Andorra          96.3   46.8   155     0
## 5 Angola           806.   727.  1900     0
## 6 Antarctica       0      0      0      0
## 7 Antigua and Barbuda  46.2   51.4   138     0
## 8 Argentina        63440. 49301. 128889    0
## 9 Armenia           3800.  3023.  8625     0
## 10 Australia        207.   543.  3480     0
## # ... with 189 more rows
```

Uso de summarise()

Recordar que primero debemos *pivotear* la data para poder trabajar con ella.
 Obtenremos resúmenes de casos por muerte COVID-19 por países.

```
covid19 %>%
  select(-c(Lat:Long)) %>%
  pivot_longer(
    cols = -c(`Province/State`:`Count`),
    names_to = "fecha",
    values_to = "fallecidos"
  ) %>%
  group_by(`Country/Region`) %>%
  summarise(
    M = mean(fallecidos, na.rm = TRUE),
    DE = sd(fallecidos, na.rm = TRUE),
    Max = max(fallecidos, na.rm = TRUE),
    Min = min(fallecidos, na.rm = TRUE)
  ) %>%
  arrange(desc(M))
```

	## # A tibble: 199 × 5	M	DE	Max	Min
	## `Country/Region`	<dbl>	<dbl>	<dbl>	<dbl>
## 1	US	481819.	322299.	1008422	0
## 2	Brazil	343401.	248850.	666971	0
## 3	India	251630.	198932.	524677	0
## 4	Mexico	172562.	116524.	324966	0
## 5	Russia	132836.	127534.	371703	0
## 6	Peru	127690.	76774.	213228	0
## 7	Italy	90448.	52966.	166875	0
## 8	Colombia	70824.	53720.	139867	0
## 9	Iran	70048.	49947.	141321	0
## 10	Indonesia	64863	61869.	156604	0
	## # ... with 189 more rows				

Uso de summarise()

Guardaremos la data pivoteada en un objeto llamado covid19_tidy, para no extender demasiado el código. Además, veamos lo sencillo que es poder introducir un filter a los resultados obtenidos.

```
covid19_tidy
## # A tibble: 246,240 × 4
##   `Province/State` `Country/Region` fecha   fallecidos
##   <chr>           <chr>        <chr>     <dbl>
## 1 <NA>            Afghanistan 1/22/20      0
## 2 <NA>            Afghanistan 1/23/20      0
## 3 <NA>            Afghanistan 1/24/20      0
## 4 <NA>            Afghanistan 1/25/20      0
## 5 <NA>            Afghanistan 1/26/20      0
## 6 <NA>            Afghanistan 1/27/20      0
## 7 <NA>            Afghanistan 1/28/20      0
## 8 <NA>            Afghanistan 1/29/20      0
## 9 <NA>            Afghanistan 1/30/20      0
## 10 <NA>           Afghanistan 1/31/20      0
## # ... with 246,230 more rows
```

Uso de summarise()

Guardaremos la data pivoteada en un objeto llamado covid19_tidy, para no extender demasiado el código. Además, veamos lo sencillo que es poder introducir un filter a los resultados obtenidos.

```
covid19_tidy %>%  
  group_by(`Country/Region`)  
  
## # A tibble: 246,240 × 4  
## # Groups:   Country/Region [199]  
##   `Province/State` `Country/Region` fecha   fallecido  
##   <chr>           <chr>       <chr>     <dbl>  
## 1 <NA>             Afghanistan 1/22/20  
## 2 <NA>             Afghanistan 1/23/20  
## 3 <NA>             Afghanistan 1/24/20  
## 4 <NA>             Afghanistan 1/25/20  
## 5 <NA>             Afghanistan 1/26/20  
## 6 <NA>             Afghanistan 1/27/20  
## 7 <NA>             Afghanistan 1/28/20  
## 8 <NA>             Afghanistan 1/29/20  
## 9 <NA>             Afghanistan 1/30/20  
## 10 <NA>            Afghanistan 1/31/20  
## # ... with 246,230 more rows, and 1 more variable:  
## #   last_update <date>
```

Uso de summarise()

Guardaremos la data pivoteada en un objeto llamado covid19_tidy, para no extender demasiado el código. Además, veamos lo sencillo que es poder introducir un filter a los resultados obtenidos.

```
covid19_tidy %>%
  group_by(`Country/Region`) %>%
  summarise(
    M = mean(fallecidos, na.rm = TRUE),
    DE = sd(fallecidos, na.rm = TRUE),
    Max = max(fallecidos, na.rm = TRUE),
    Min = min(fallecidos, na.rm = TRUE)
  )
```

	## # A tibble: 199 × 5	## `Country/Region`	M	DE	Max	Min
		## <chr>	<dbl>	<dbl>	<dbl>	<dbl>
## 1	Afghanistan	3754.	2962.	7708	0	
## 2	Albania	1723.	1314.	3497	0	
## 3	Algeria	3451.	2332.	6875	0	
## 4	Andorra	96.3	46.8	155	0	
## 5	Angola	806.	727.	1900	0	
## 6	Antarctica	0	0	0	0	
## 7	Antigua and Barbuda	46.2	51.4	138	0	
## 8	Argentina	63440.	49301.	128889	0	
## 9	Armenia	3800.	3023.	8625	0	
## 10	Australia	207.	543.	3480	0	
	## # ... with 189 more rows					

Uso de summarise()

Guardaremos la data pivoteada en un objeto llamado covid19_tidy, para no extender demasiado el código. Además, veamos lo sencillo que es poder introducir un filter a los resultados obtenidos.

```
covid19_tidy %>%
  group_by(`Country/Region`) %>%
  summarise(
    M = mean(fallecidos, na.rm = TRUE),
    DE = sd(fallecidos, na.rm = TRUE),
    Max = max(fallecidos, na.rm = TRUE),
    Min = min(fallecidos, na.rm = TRUE)
  ) %>%
  arrange(desc(M))
```

	## # A tibble: 199 × 5	## `Country/Region`	M	DE	Max	Min
		## <chr>	<dbl>	<dbl>	<dbl>	<dbl>
## 1	## # A tibble: 199 × 5	## US	481819.	322299.	1008422	0
## 2	## `Country/Region`	## Brazil	343401.	248850.	666971	0
## 3	## <chr>	## India	251630.	198932.	524677	0
## 4	## M	## Mexico	172562.	116524.	324966	0
## 5	## DE	## Russia	132836.	127534.	371703	0
## 6	## Max	## Peru	127690.	76774.	213228	0
## 7	## Min	## Italy	90448.	52966.	166875	0
## 8		## Colombia	70824.	53720.	139867	0
## 9		## Iran	70048.	49947.	141321	0
## 10		## Indonesia	64863	61869.	156604	0
		## # ... with 189 more rows				

Uso de summarise()

Guardaremos la data pivoteada en un objeto llamado covid19_tidy, para no extender demasiado el código. Además, veamos lo sencillo que es poder introducir un filter a los resultados obtenidos.

```
covid19_tidy
## # A tibble: 246,240 × 4
##   `Province/State` `Country/Region` fecha   fallecidos
##   <chr>           <chr>        <chr>     <dbl>
## 1 <NA>            Afghanistan 1/22/20      0
## 2 <NA>            Afghanistan 1/23/20      0
## 3 <NA>            Afghanistan 1/24/20      0
## 4 <NA>            Afghanistan 1/25/20      0
## 5 <NA>            Afghanistan 1/26/20      0
## 6 <NA>            Afghanistan 1/27/20      0
## 7 <NA>            Afghanistan 1/28/20      0
## 8 <NA>            Afghanistan 1/29/20      0
## 9 <NA>            Afghanistan 1/30/20      0
## 10 <NA>           Afghanistan 1/31/20      0
## # ... with 246,230 more rows
```

Uso de summarise()

Guardaremos la data pivoteada en un objeto llamado covid19_tidy, para no extender demasiado el código. Además, veamos lo sencillo que es poder introducir un filter a los resultados obtenidos.

```
covid19_tidy %>%  
  mutate(  
    fecha = lubridate::mdy(fecha)  
  )  
  
## # A tibble: 246,240 × 4  
##   `Province/State` `Country/Region` fecha  
##   <chr>           <chr>          <date>  
## 1 <NA>             Afghanistan  2020-01-22  
## 2 <NA>             Afghanistan  2020-01-23  
## 3 <NA>             Afghanistan  2020-01-24  
## 4 <NA>             Afghanistan  2020-01-25  
## 5 <NA>             Afghanistan  2020-01-26  
## 6 <NA>             Afghanistan  2020-01-27  
## 7 <NA>             Afghanistan  2020-01-28  
## 8 <NA>             Afghanistan  2020-01-29  
## 9 <NA>             Afghanistan  2020-01-30  
## 10 <NA>            Afghanistan  2020-01-31  
## # ... with 246,230 more rows, and 1 more variable:  
## #   fallecidos <dbl>
```

Uso de summarise()

Guardaremos la data pivoteada en un objeto llamado covid19_tidy, para no extender demasiado el código. Además, veamos lo sencillo que es poder introducir un filter a los resultados obtenidos.

```
covid19_tidy %>%  
  mutate(  
    fecha = lubridate::mdy(fecha)  
  ) %>%  
  filter(fecha > "2022-01-01")
```

```
## # A tibble: 43,605 × 4  
##   `Province/State` `Country/Region` fecha  
##   <chr>           <chr>          <date>  
## 1 <NA>             Afghanistan  2022-01-02  
## 2 <NA>             Afghanistan  2022-01-03  
## 3 <NA>             Afghanistan  2022-01-04  
## 4 <NA>             Afghanistan  2022-01-05  
## 5 <NA>             Afghanistan  2022-01-06  
## 6 <NA>             Afghanistan  2022-01-07  
## 7 <NA>             Afghanistan  2022-01-08  
## 8 <NA>             Afghanistan  2022-01-09  
## 9 <NA>             Afghanistan  2022-01-10  
## 10 <NA>            Afghanistan  2022-01-11  
## # ... with 43,595 more rows, and 1 more variable:  
## #   fallecidos <dbl>
```

Uso de summarise()

Guardaremos la data pivoteada en un objeto llamado covid19_tidy, para no extender demasiado el código. Además, veamos lo sencillo que es poder introducir un filter a los resultados obtenidos.

```
covid19_tidy %>%  
  mutate(  
    fecha = lubridate::mdy(fecha)  
  ) %>%  
  filter(fecha > "2022-01-01") %>%  
  group_by(`Country/Region`)
```

```
## # A tibble: 43,605 × 4  
## # Groups:   Country/Region [199]  
##   `Province/State` `Country/Region` fecha  
##   <chr>           <chr>          <date>  
## 1 <NA>             Afghanistan  2022-01-02  
## 2 <NA>             Afghanistan  2022-01-03  
## 3 <NA>             Afghanistan  2022-01-04  
## 4 <NA>             Afghanistan  2022-01-05  
## 5 <NA>             Afghanistan  2022-01-06  
## 6 <NA>             Afghanistan  2022-01-07  
## 7 <NA>             Afghanistan  2022-01-08  
## 8 <NA>             Afghanistan  2022-01-09  
## 9 <NA>             Afghanistan  2022-01-10  
## 10 <NA>            Afghanistan  2022-01-11  
## # ... with 43,595 more rows, and 1 more variable:
```

Uso de summarise()

Guardaremos la data pivoteada en un objeto llamado covid19_tidy, para no extender demasiado el código. Además, veamos lo sencillo que es poder introducir un filter a los resultados obtenidos.

```
covid19_tidy %>%
  mutate(
    fecha = lubridate::mdy(fecha)
  ) %>%
  filter(fecha > "2022-01-01") %>%
  group_by(`Country/Region`) %>%
  summarise(
    M = mean(fallecidos, na.rm = TRUE),
    DE = sd(fallecidos, na.rm = TRUE),
    Max = max(fallecidos, na.rm = TRUE),
    Min = min(fallecidos, na.rm = TRUE)
  )
## # A tibble: 199 × 5
##   `Country/Region`      M       DE     Max     Min
##   <chr>          <dbl>    <dbl>    <dbl>    <dbl>
## 1 Afghanistan  7585.   122.    7708    7361
## 2 Albania      3435.   88.1    3497    3220
## 3 Algeria      6758.   181.    6875    6291
## 4 Andorra       150.    4.17    155     140
## 5 Angola        1890.   25.6    1900    1772
## 6 Antarctica     0       0       0       0
## 7 Antigua and Barbuda 133.    5.91    138     119
## 8 Argentina     125564. 3874.  128889  117204
## 9 Armenia       8421.   254.    8625    7977
## 10 Australia     707.   1020.   3480      1
## # ... with 189 more rows
```

Uso de summarise()

Guardaremos la data pivoteada en un objeto llamado covid19_tidy, para no extender demasiado el código. Además, veamos lo sencillo que es poder introducir un filter a los resultados obtenidos.

```
covid19_tidy %>%
  mutate(
    fecha = lubridate::mdy(fecha)
  ) %>%
  filter(fecha > "2022-01-01") %>%
  group_by(`Country/Region`) %>%
  summarise(
    M = mean(fallecidos, na.rm = TRUE),
    DE = sd(fallecidos, na.rm = TRUE),
    Max = max(fallecidos, na.rm = TRUE),
    Min = min(fallecidos, na.rm = TRUE)
  ) %>%
  arrange(desc(M))
```

## # A tibble: 199 × 5	## `Country/Region`	M	DE	Max	Min
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>
## 1 US		950206.	55279.	1008422	825847
## 2 Brazil		649643.	16471.	666971	619401
## 3 India		512237.	13753.	524677	481893
## 4 Russia		348769.	21122.	371703	304284
## 5 Mexico		317212.	8749.	324966	299544
## 6 Peru		210005.	3547.	213228	202782
## 7 Italy		155761.	8632.	166875	137646
## 8 Indonesia		151164.	5193.	156604	144097
## 9 Iran		137613.	3739.	141321	131680
## 10 Colombia		137575.	3267.	139867	130026
## # ... with 189 more rows					

¡Hazlo tú mismo!

1. Replica los procedimientos observados en las diapositivas
2. Genera nuevas solicitudes en summarise que consideres pertinente.
3. Usa nuevas reglas de filtrado

08 : 00

Retroalimentación

¡Gracias!

✉ imt.innovlab@oficinas-upch.pe

🗣 [@healthinnovation](https://twitter.com/@healthinnovation)

🐦 [@innovalab_imt](https://twitter.com/@innovalab_imt)

Estas diapositivas fueron creadas mediante el paquete `xaringan` y `xaringanthemer`.