Sesion 4

Curso: POL304 - Estadística para el análisis político 2

Jefes de práctica: Alexander Benites, Chiara Zamora y Airám Bello

Ciclo 2023-2

Para esta sesión, vamos a necesitar cargar de nuevo la base de urbanización.

Para quienes exportaron la base de datos, podemos cargarla usando el comando import

```
library(rio)
data="https://github.com/Alexanderbenit7/EAP2_2023-2/raw/main/data/urban.xlsx"
urban=import(data)
```

1. Limpieza de datos

En esta sección, veremos otros caminos para separar los datos.

1.1. Parsers

Los interpretes pueden facilitar en algo, cuando se trate de un único número en la columna. El comando **parse_number** nos permite extraer valores númericos de un string.

```
library(readr)
parse_number(x = "$139,100 32")
```

```
## [1] 139100
```

Pero sólo recupera el primer valor. Confirmando:

```
parse_number(urban$URBANIZATION)
```

```
##   [1]  24.0  47.0  65.0  92.0  89.0  57.0 100.0  30.0  92.0  64.0  47.0  89.0
##  [13]  67.0  52.0  84.0  89.0  27.0  40.0  73.0  97.0  52.0  41.0 100.0  35.0
##  [25]  66.0  47.0  60.0  86.0  40.0  75.0  71.0  20.0  33.0  10.0  22.0  57.0
##  [37]  80.0  60.0 100.0  39.0  27.0  88.0  43.0  74.0  28.0  34.0  61.0  74.0
##  [49]  63.0  49.0  57.0  76.0  70.0  73.0  87.0  87.0  74.0  69.0  66.0  43.0
##  [61]  61.0  39.0  21.0  69.0  17.0  92.0  41.0  52.0  63.0  77.0  52.0  85.0
```

```
##  [73]  57.0  72.0  53.0  74.0  50.0 100.0  61.0  84.0  31.0  93.0  49.0  31.0
##  [85]  34.0  30.0  28.0  47.0 100.0  48.0 100.0  68.0  92.0  29.0  52.0  68.0
##  [97]  67.0  61.0  51.0  92.0  68.0  53.0  66.0  31.0  78.0  58.0  22.0  44.0
## [109]  63.0  81.0  98.0  36.0  31.0  68.0  87.0  25.0  60.0  78.0  14.0  67.0
## [121]  82.0 100.0  67.0  29.0  19.0  70.0  38.0  32.0  94.0  71.0  41.0  42.0
## [133]  77.0  22.0  42.0 100.0  57.0  60.0  14.0  56.0  37.0  37.0 100.0  17.0
## [145]  82.0  93.0  65.0  87.0  57.0  16.0  48.0  39.0  91.0  77.0  72.0  36.0
## [157]  81.0  73.0  12.0  60.0  71.0  65.0   0.0  61.0  59.0  98.0  96.0  54.0
## [169]  73.0  18.0  39.0  32.0  28.0  89.0  47.0  23.0  94.0  61.0  82.0  42.0
## [181]  52.0  54.0  38.0 100.0  56.0  48.0  18.0  37.0  61.0  77.0  15.0  43.0
## [193]  75.0  25.0  85.0  73.0  54.0  26.0  25.0  33.0  27.0  42.0   0.0  25.0
## [205]  13.0  67.0  69.0  49.0  92.0  49.0  13.0  68.0  78.0  90.0  82.0  92.0
## [217]  37.0  25.0  93.0  28.0  95.0   0.0  72.0  81.0  48.6  31.0  35.0  37.0
```

En ese sentido, podemos usarlo combinado con la estrategia separadora (la vimos la sesión pasada)

```
library(stringr)
library(magrittr) # para %>%

urban$pop_urb3=str_split(string = urban$URBANIZATION,
          pattern = 'rate of urbanization:',
          simplify = T)[,1]%>%parse_number()

urban$rate_urb3=str_split(string = urban$URBANIZATION,
          pattern = 'rate of urbanization:',
          simplify = T)[,2]%>%parse_number()
```

1.2. Usando el comando substr()

Otra función que nos permite extraer cadenas es **substr()**, cuya síntesis es:

substr("x", "star=", "stop=")

Donde x es un vector caracter o texto del que se desea extraer una subcadena. star indica el orden inicial, es decir, la posición inicial de la que se empezará a extraer. Por último, stop indica el orden final. El orden inicial y final viene expresado por números (de la clase integer - aunque funciona con numéricos).

Veamos un ejemplo, si deseamos extraer desde el tercer al sexto caracter del vector x. La sintaxis sería la siguiente:

```
x= "$139,100"
substr(x,start=4,stop=8)
```

```
## [1] "9,100"
```

gsub()

También es importante la función gsub, que busca un character y lo reemplaza:

```
porcentajes=c('13%','33%','55%')
gsub('%',"",porcentajes) # lo reemplaza por nada ''.
```

```
## [1] "13" "33" "55"
```

En el caso anterior

```
gsub(',|\\$|\\?',"Alex","$139,100?")
```

```
## [1] "Alex139Alex100Alex"
```

str_pad()

Agregamos elementos hasta un número específico de caracteres:

```
#A la derecha:
stringr::str_pad("$139100", 10, side = "right", pad = 0)
```

```
## [1] "$139100000"
```

```
#A la izquierda
stringr::str_pad("$139100", 10, side = "left", pad = 0)
```

```
## [1] "000$139100"
```

EJERCICIO.

Como parte de un proyecto que analiza los procesos de movilidad interna en el país, le piden obtener la tasa de movilidad a nivel distrital, que hace referencia a:

Número de personas que NO vivían en ese distrito hace 5 años / Número total de personas censadas en el distrito

Para ello, puede recurrir a las bases de datos del censo, que se encuentran en el REDATAM del INEI. En ese sentido, debe realizar lo siguiente:

- Elimine filas y columnas innecesarias
- Separe regiones, provincias y departamentos
- Corrija inconsistencia en los ubigeos (algunos casos no están arrancando en 0 cuando deberían)
- Formatee sus variables
- Extraiga la tasa de movilidad distrital

La base de datos debe verse de la siguiente manera:

UBIGEO	REGION	PROVINCIA	DISTRITO	NO_NAC	SI_VIV	NO_VIV	TOTAL_POB	TASA_MOV
010101	Amazonas	Chachapoyas	Chachapoyas	2633	22774	7182	32589	0.220381110
010102	Amazonas	Chachapoyas	Asunción	22	220	20	262	0.076335878
010103	Amazonas	Chachapoyas	Balsas	111	938	87	1136	0.076584507
010104	Amazonas	Chachapoyas	Cheto	44	533	65	642	0.101246106
010105	Amazonas	Chachapoyas	Chiliquin	49	483	53	585	0.090598291
010106	Amazonas	Chachapoyas	Chuquibamba	184	1472	125	1781	0.070185289
010107	Amazonas	Chachapoyas	Granada	31	423	26	480	0.054166667
010108	Amazonas	Chachapoyas	Huancas	31	684	543	1258	0.431637520
010109	Amazonas	Chachapoyas	La Jalca	408	3416	154	3978	0.038712921
010110	Amazonas	Chachapoyas	Leimebamba	284	2850	486	3620	0.134254144

```
dataMov = import("https://github.com/Alexanderbenit7/EAP2_2023-2/raw/main/data/reporte-3.xlsX")
```

```
## New names:
## * `` -> `...2`
## * `` -> `...3`
## * `` -> `...4`
## * `` -> `...5`
## * `` -> `...6`
```

Eliminando columnas innecesarias:

```
dataMov = dataMov[,c(2:6)]
dataMov = dataMov[-c(1:5,1880:1883),]
```

Nombres a las columnas:

```
colnames(dataMov) = c("UBIGEO","UBI","NO_NAC","SI_VIV","NO_VIV")
```

Nos quedamos con el distrito:

```
dataMov$DISTRITO=str_split(dataMov$UBI,'distrito: ',simplify = T) [,2]
dataMov$UBI=str_split(dataMov$UBI,'distrito: ',simplify = T) [,1]
```

```
dataMov$REGION=str_split(dataMov$UBI,', ',simplify = T)[,1]
dataMov$PROVINCIA=str_split(dataMov$UBI,', ',simplify = T)[,2]
```

Agregamos el 0 a los UBIGEOS:

```
dataMov$UBIGEO=stringr::str_pad(dataMov$UBIGEO, 6, side = "left", pad = 0)
```

Reordenando:

```
dataMov = dataMov[,c(1,7,8,6,3,4,5)]
dataMov[,5:7]=lapply(dataMov[,5:7], as.numeric)
str(dataMov)
```

```
## 'data.frame':    1874 obs. of  7 variables:
##  $ UBIGEO   : chr  "010101" "010102" "010103" "010104" ...
##  $ REGION   : chr  "Amazonas" "Amazonas" "Amazonas" "Amazonas" ...
##  $ PROVINCIA: chr  "Chachapoyas" "Chachapoyas" "Chachapoyas" "Chachapoyas" ...
##  $ DISTRITO : chr  "Chachapoyas" "Asunción" "Balsas" "Cheto" ...
##  $ NO_NAC   : num  2633 22 111 44 49 ...
##  $ SI_VIV   : num  22774 220 938 533 483 ...
##  $ NO_VIV   : num  7182 20 87 65 53 ...
```

Total de la población censada por distrito:

```
dataMov$TOTAL_POB = dataMov$NO_NAC+dataMov$SI_VIV+dataMov$NO_VIV
dataMov$TASA_MOV = dataMov$NO_VIV/dataMov$TOTAL_POB
```

Ahora, si quisieramos los UBIGEOS de región y provincia, podríamos hacer esto:

```
dataMov$UBIGEO_REG = substr(dataMov$UBIGEO,1,2)
```

Agregamos los cero con stringr:

```
dataMov$UBIGEO_REG=stringr::str_pad(dataMov$UBIGEO_REG, 6, side = "right", pad = 0)
```

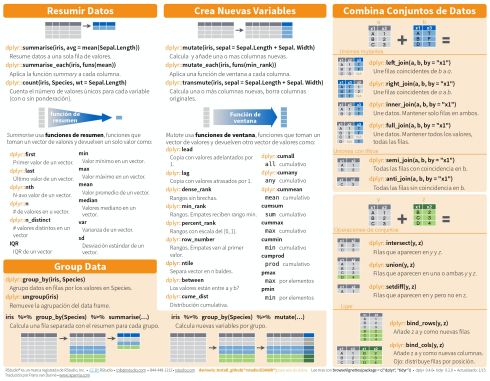
Lo mismo para las provincias:

```
dataMov$UBIGEO_PROV = substr(dataMov$UBIGEO,1,4)
dataMov$UBIGEO_PROV=stringr::str_pad(dataMov$UBIGEO_PROV, 6, side = "right", pad = 0)
```

2. Otras funciones

dplyr()

Uno de los paquetes que más funciona para manipular datos de forma fácil es dplyr. Este paquete tiene, entre otras, cinco funciones para manipular datos: **select() filter() arrange() mutate() summarize()**




```
data="https://github.com/WendyAdrianzenRossi/Statistics/raw/main/Data_sample.xlsx"
base=import(data)
```

Cargamos la librería:

```
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
```

select()

Select nos permite seleccionar columnas. La sintaxis sería: select(dataframe, col1, col2) conde col1, col2, se refiere a los nombres de las columnas que queramos seleccionar.

Por ejemplo supongamos que queremos seleccionar únicamente las columnas de mes de desembolso y el tipo de moneda:

```
select(base, MES_DESEMBOLSO, MONEDA)
```

También podemos seleccionar todas las columnas menos algunas, esto lo hacemos poniendo - antes del nombre de la columna que no queremos seleccionar. Por ejemplo, si queremos todas las columnas menos el tipo de moneda

```
select(base, -MONEDA)
```

Si queremos seleccionar un rango de columnas por ejemplo de mes de desembolso a tasa de desembolso usamos :

```
select(base, MES_DESEMBOLSO:TASA_DESEMBOLSO)
```

Si queremos guardar el resultado de esa función en un nuevo objeto, debemos asignarlo con <-. Por ejemplo guardemos en un objeto el rango de columnas que acabamos de seleccionar

```
datos <- select(base, MES_DESEMBOLSO:TASA_DESEMBOLSO)
head(datos,10)
```

```
##    MES_DESEMBOLSO  MONEDA MONTO_DESEMBOLSO TASA_DESEMBOLSO
## 1          201801   Soles           425000          0.0850
## 2          201801   Soles           250000          0.0891
## 3          201801   Soles           510000          0.0904
## 4          201801   Soles           200000          0.1007
## 5          201801   Soles           600000          0.0826
## 6          201801 Dólares           500000          0.0585
## 7          201801 Dólares           155000          0.0727
## 8          201801 Dólares           140000          0.0685
## 9          201801   Soles           150000          0.1164
## 10         201801   Soles           367000          0.0861
```

filter()

La función filter nos permite filtrar filas:

La sintaxis es simple: **filter(base, condicion)**. Donde condición es la condión lógica por la que queremos filtrar datos. Para ello usamos operadores lógicos:

- > mayor que
- < menor que
- >= mayor o igual que
- <= menor o igual que
- == igual que (se ponen dos signos de igual)
- *!=* diferente
- *&* y
- | o
- *is.na(variable)* filtra los valores en blanco de la variable seleccionada.
- *!is.na(variable)* filtra los valores que no están en blanco de la variable.

Por ejemplo si queremos filtrar solamente los desembolsos superiores a un millón:

```
filter(base, MONTO_DESEMBOLSO>1000000)
```

O solamente los desembolsos superiores a un millón y que el tipo de moneda sea dólares:

```
filter(base, MONTO_DESEMBOLSO>1000000 & MONEDA=="Dólares")
```

```
##       ID MES_DESEMBOLSO  MONEDA MONTO_DESEMBOLSO TASA_DESEMBOLSO
## 1    720         201806 Dólares          1300000          0.0600
## 2    758         201806 Dólares          2500000          0.0550
## 3    945         201808 Dólares          1458713          0.0800
## 4   1191         201809 Dólares          1427000          0.0600
## 5   1985         201903 Dólares          1200000          0.0650
## 6   2214         201905 Dólares          1250000          0.0695
## 7   3309         201912 Dólares          1500000          0.0500
## 8   4681         202009 Dólares          1750000          0.0635
## 9   4682         202009 Dólares          1019955          0.0740
## 10 5133         202101 Dólares          1330137          0.0500
## 11 5161         202101 Dólares          1391711          0.0422
## 12 5514         202104 Dólares          1800000          0.0370
```

Ahora, si queremos filtrar la base por todos los registros que no tengan valores vacíos en el monto de desembolso:

```
base<- filter(base, !is.na(MONTO_DESEMBOLSO))
sum(is.na(base$MONTO_DESEMBOLSO))
```

```
## [1] 0
```

mutate()

Mutate nos permite crear nuevas columnas de forma fácil.

Podemos crear una variable que me diga cuánto es el monto de desembolso por la tasa de desembolso:

```
base <- mutate(base, montoxtasa=MONTO_DESEMBOLSO*TASA_DESEMBOLSO)
```

Ahora podemos crear una nueva variable que me categorice los datos en si son mayores a la media o no. Esto podemos hacerlo con la función if_else() o ifelse (funcionan igual). La sintaxis es: ifelse(condición, valor cierto, valor falso). (Es similar a la función if en Excel).

```
base <- mutate(base, categoria=ifelse(montoxtasa<28473.43,"menor", "mayor"))
table(base$categoria)
```

```
##
## mayor menor
##  1896  4028
```

Con mutate podemos crear multiples variables a la vez, separando cada una por coma, por ejemplo:

```
base <- mutate(base, montoxtasa=MONTO_DESEMBOLSO*TASA_DESEMBOLSO,
                     categoria=ifelse(montoxtasa<28473.43,"menor", "mayor"))
```

arrange()

Arrange nos permite ordenar las base por una o varias columnas Por ejemplo, queremos ordenar la base en orden ascendente por monto de desembolso y por monto por tasa:

```
base <- arrange(base, MONTO_DESEMBOLSO, montoxtasa)
```

```
base <- arrange(base, desc(MONTO_DESEMBOLSO), desc(montoxtasa))
```

1.3 Merge

Este proceso combina data frames con difente información, siempre que tengan un campo común (key), y este no se repita en ninguna otra columna.

```
data="https://github.com/PoliticayGobiernoPUCP/estadistica_anapol2/raw/master/DATA/corruption.csv"
corru=import(data)

data2="https://github.com/PoliticayGobiernoPUCP/estadistica_anapol2/raw/master/DATA/demo.csv"
demo=import(data2)
```

Estos dos data frames deben tener un campo (columa) en común que sirva de "key":

```
names(corru)
```

```
## [1] "Rank"      "Country"   "2016Score" "2015Score" "2014Score" "2013Score"
## [7] "2012Score" "Region"
```

```r
names(demo)
```

```
##  [1] "Rank"                           "Country"
##  [3] "Score"                          "Electoralprocessandpluralism"
##  [5] "Functioningofgovernment"       "Politicalparticipation"
##  [7] "Politicalculture"               "Civilliberties"
##  [9] "Regimetype"                     "Region"
```

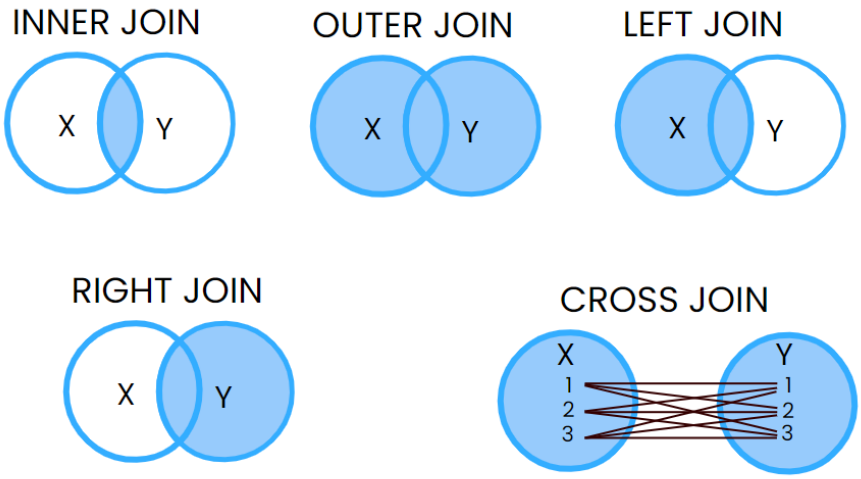
El merge producirá una tabla integrando las columnas que correspondan a cada key. Vea que la columna rank está presente en las tres pero no es la key; como el ranking se puede calcular si se necesitase, eliminemos los de cada data frame:

```r
# eliminando
corru$Rank=NULL
demo$Rank=NULL

#La columna Score de la tabla democ podriamos cambiar para no olvidar qué representa:

colnames(demo)[2]='ScoreDemo'

#Hay una column Region en las dos bases, quedémonos con la de democracia:
corru$Region=NULL
```



Con estas bases de datos, vamos a explorar los escenarios de merge que podemos aplicar. Ten en cuenta que el método principal de la función merge es para data frames. Sin embargo, merge es una función genérica que también se puede usar con otros objetos (como vectores o matrices), pero serán transformados a la clase data.frame

Inner join

Un inner join (en realidad un natural join), es la unión de conjuntos de datos más habitual que se puede realizar. Consiste en fusionar dos data frames en uno que contenga los elementos comunes de ambos. Para fusionar o unir los dos conjuntos de datos de muestra, solo tienes que pasarlos a la función merge, sin la necesidad de cambiar otros argumentos, debido a que, de manera predeterminada, la función combina los conjuntos de datos por los nombres de las columnas comunes.

```
corrgdp=merge(corru,demo)
str(corrgdp)
```

```
## 'data.frame':    157 obs. of  14 variables:
##  $ Country                       : chr  "Afghanistan" "Albania" "Algeria" "Angola" ...
##  $ 2016Score                     : chr  "15" "39" "34" "18" ...
##  $ 2015Score                     : chr  "11" "36" "36" "15" ...
##  $ 2014Score                     : chr  "12" "33" "36" "19" ...
##  $ 2013Score                     : chr  "8" "31" "36" "23" ...
##  $ 2012Score                     : chr  "8" "33" "34" "22" ...
##  $ ScoreDemo                     : num  2.97 5.98 3.5 3.62 7.02 4.79 9.09 8.29 2.65 2.71 ...
##  $ Electoralprocessandpluralism: num  2.92 7 2.58 1.75 9.17 5.67 10 9.58 0.5 0.83 ...
##  $ Functioningofgovernment     : num  1.14 4.71 2.21 2.86 5.36 4.64 8.93 7.86 2.14 3.21 ...
##  $ Politicalparticipation      : num  4.44 5.56 3.89 5.56 6.11 5.56 7.78 8.33 3.33 2.78 ...
##  $ Politicalculture             : num  2.5 5 5 5 6.25 2.5 8.75 6.88 3.75 4.38 ...
##  $ Civilliberties               : num  3.82 7.65 3.82 2.94 8.24 5.59 10 8.82 3.53 2.35 ...
##  $ Regimetype                    : chr  "Authoritarian" "Hybrid regime" "Authoritarian" "Authoritaria
##  $ Region                        : chr  "Asia" "Europe" "Africa" "Africa" ...
```

El resultado es un data frame que contiene las observaciones que están presentes en ambos data frames, las que no, no están presentes en la salida resultante.

Outer join

El outer join, o unión completa, combina todas las columnas de ambos conjuntos de datos en uno para todos los elemento. Para crear el full outer join de dos data frames en R tienes que establecer el argumento all como TRUE.

```
corrgdp2=merge(corru,demo,all=TRUE)
str(corrgdp2)
```

```
## 'data.frame':    187 obs. of  14 variables:
##  $ Country                       : chr  "Afghanistan" "Albania" "Algeria" "Angola" ...
##  $ 2016Score                     : chr  "15" "39" "34" "18" ...
##  $ 2015Score                     : chr  "11" "36" "36" "15" ...
##  $ 2014Score                     : chr  "12" "33" "36" "19" ...
##  $ 2013Score                     : chr  "8" "31" "36" "23" ...
##  $ 2012Score                     : chr  "8" "33" "34" "22" ...
##  $ ScoreDemo                     : num  2.97 5.98 3.5 3.62 7.02 4.79 9.09 8.29 2.65 NA ...
##  $ Electoralprocessandpluralism: num  2.92 7 2.58 1.75 9.17 5.67 10 9.58 0.5 NA ...
##  $ Functioningofgovernment     : num  1.14 4.71 2.21 2.86 5.36 4.64 8.93 7.86 2.14 NA ...
##  $ Politicalparticipation      : num  4.44 5.56 3.89 5.56 6.11 5.56 7.78 8.33 3.33 NA ...
##  $ Politicalculture             : num  2.5 5 5 5 6.25 2.5 8.75 6.88 3.75 NA ...
##  $ Civilliberties               : num  3.82 7.65 3.82 2.94 8.24 5.59 10 8.82 3.53 NA ...
##  $ Regimetype                    : chr  "Authoritarian" "Hybrid regime" "Authoritarian" "Authoritaria
##  $ Region                        : chr  "Asia" "Europe" "Africa" "Africa" ...
```

Ahora tenemos un data frame con todos los casos. Como no todas las filas en el primer data frame coinciden con todas las filas en el segundo, en la salida aparecen valores NA en esos casos.

Left join

El left join en R consiste en unir todas las filas del primer data frame con los valores correspondientes del segundo.Para crear la unión, tienes que establecer all.x = TRUE.

```
corrgdp3=merge(corru,demo,all.x = TRUE)
str(corrgdp3)
```

```
## 'data.frame':    177 obs. of  14 variables:
##  $ Country                      : chr  "Afghanistan" "Albania" "Algeria" "Angola" ...
##  $ 2016Score                    : chr  "15" "39" "34" "18" ...
##  $ 2015Score                    : chr  "11" "36" "36" "15" ...
##  $ 2014Score                    : chr  "12" "33" "36" "19" ...
##  $ 2013Score                    : chr  "8" "31" "36" "23" ...
##  $ 2012Score                    : chr  "8" "33" "34" "22" ...
##  $ ScoreDemo                    : num  2.97 5.98 3.5 3.62 7.02 4.79 9.09 8.29 2.65 NA ...
##  $ Electoralprocessandpluralism: num  2.92 7 2.58 1.75 9.17 5.67 10 9.58 0.5 NA ...
##  $ Functioningofgovernment     : num  1.14 4.71 2.21 2.86 5.36 4.64 8.93 7.86 2.14 NA ...
##  $ Politicalparticipation      : num  4.44 5.56 3.89 5.56 6.11 5.56 7.78 8.33 3.33 NA ...
##  $ Politicalculture             : num  2.5 5 5 5 6.25 2.5 8.75 6.88 3.75 NA ...
##  $ Civilliberties               : num  3.82 7.65 3.82 2.94 8.24 5.59 10 8.82 3.53 NA ...
##  $ Regimetype                    : chr  "Authoritarian" "Hybrid regime" "Authoritarian" "Authoritaria
##  $ Region                        : chr  "Asia" "Europe" "Africa" "Africa" ...
```

Ahora tenemos un data frame con todos los casos de la primera base de datos (177 casos). Si esta contiene observaciones que no están en la segunda base de datos, tendra NA como valores en las variables que correspondan.

Right join

El right join en R es lo opuesto al left outer join. En este caso, la combinación consiste en unir todas las filas del segundo data frame con las correspondientes en el primero.En consecuencia, necesitarás establecer el argumento all.y como TRUE para unir los data frames de esta manera.

```
corrgdp4=merge(corru,demo,all.y = TRUE)
str(corrgdp4)
```

```
## 'data.frame':    167 obs. of  14 variables:
##  $ Country                      : chr  "Afghanistan" "Albania" "Algeria" "Angola" ...
##  $ 2016Score                    : chr  "15" "39" "34" "18" ...
##  $ 2015Score                    : chr  "11" "36" "36" "15" ...
##  $ 2014Score                    : chr  "12" "33" "36" "19" ...
##  $ 2013Score                    : chr  "8" "31" "36" "23" ...
##  $ 2012Score                    : chr  "8" "33" "34" "22" ...
##  $ ScoreDemo                    : num  2.97 5.98 3.5 3.62 7.02 4.79 9.09 8.29 2.65 2.71 ...
##  $ Electoralprocessandpluralism: num  2.92 7 2.58 1.75 9.17 5.67 10 9.58 0.5 0.83 ...
##  $ Functioningofgovernment     : num  1.14 4.71 2.21 2.86 5.36 4.64 8.93 7.86 2.14 3.21 ...
##  $ Politicalparticipation      : num  4.44 5.56 3.89 5.56 6.11 5.56 7.78 8.33 3.33 2.78 ...
##  $ Politicalculture             : num  2.5 5 5 5 6.25 2.5 8.75 6.88 3.75 4.38 ...
##  $ Civilliberties               : num  3.82 7.65 3.82 2.94 8.24 5.59 10 8.82 3.53 2.35 ...
##  $ Regimetype                    : chr  "Authoritarian" "Hybrid regime" "Authoritarian" "Authoritaria
##  $ Region                        : chr  "Asia" "Europe" "Africa" "Africa" ...
```

Ahora tenemos un data frame con todos los casos de la segunda base de datos (228 casos). Si esta contiene observaciones que no están en la segunda base de datos, tendra NA como valores en las variables que correspondan.

Cross join

El cross join o unión cruzada, realiza el producto cartesiano de los conjuntos de datos. Puedes crear un cross join en R estableciendo como NULL el argumento by de la función merge.

```
corrgdp5=merge(corru,demo,by=NULL)
str(corrgdp5)
```

```
## 'data.frame':    29559 obs. of  15 variables:
##  $ Country.x                    : chr  "Denmark" "New Zealand" "Finland" "Sweden" ...
##  $ 2016Score                    : chr  "90" "90" "89" "88" ...
##  $ 2015Score                    : chr  "91" "88" "90" "89" ...
##  $ 2014Score                    : chr  "92" "91" "89" "87" ...
##  $ 2013Score                    : chr  "91" "91" "89" "89" ...
##  $ 2012Score                    : chr  "90" "90" "90" "88" ...
##  $ Country.y                    : chr  "Norway" "Norway" "Norway" "Norway" ...
##  $ ScoreDemo                    : num  9.87 9.87 9.87 9.87 9.87 9.87 9.87 9.87 9.87 9.87 ...
##  $ Electoralprocessandpluralism: num  10 10 10 10 10 10 10 10 10 10 ...
##  $ Functioningofgovernment     : num  9.64 9.64 9.64 9.64 9.64 9.64 9.64 9.64 9.64 9.64 ...
##  $ Politicalparticipation      : num  10 10 10 10 10 10 10 10 10 10 ...
##  $ Politicalculture             : num  10 10 10 10 10 10 10 10 10 10 ...
##  $ Civilliberties               : num  9.71 9.71 9.71 9.71 9.71 9.71 9.71 9.71 9.71 9.71 ...
##  $ Regimetype                    : chr  "Full democracy" "Full democracy" "Full democracy" "Full dem
##  $ Region                        : chr  "Europe" "Europe" "Europe" "Europe" ...
```

Estas operaciones también las podemos hacer con Dplyr, la cual hemos utilizado anteriormente.

```
library(dplyr)
data_resultado1=left_join(corru,demo,by="Country") #left_join Prioriza las filas de la primera base de
head(data_resultado1)
```

```
##       Country 2016Score 2015Score 2014Score 2013Score 2012Score ScoreDemo
## 1     Denmark        90        91        92        91        90      9.22
## 2 New Zealand        90        88        91        91        90      9.26
## 3     Finland        89        90        89        89        90      9.14
## 4      Sweden        88        89        87        89        88      9.39
## 5 Switzerland        86        86        86        85        86      9.03
## 6      Norway        85        87        86        86        85      9.87
##   Electoralprocessandpluralism Functioningofgovernment
## 1                        10.00                     9.29
## 2                        10.00                     9.29
## 3                        10.00                     8.93
## 4                         9.58                     9.64
## 5                         9.58                     9.29
## 6                        10.00                     9.64
##   Politicalparticipation Politicalculture Civilliberties     Regimetype
## 1                   8.33             9.38            9.12 Full democracy
## 2                   8.89             8.13           10.00 Full democracy
```

```
## 3                    8.33             8.75            9.71 Full democracy
## 4                    8.33            10.00            9.41 Full democracy
## 5                    7.78             9.38            9.12 Full democracy
## 6                   10.00            10.00            9.71 Full democracy
##    Region
## 1  Europe
## 2 Oceania
## 3  Europe
## 4  Europe
## 5  Europe
## 6  Europe
```

```
data_resultado2=right_join(corru,demo,by="Country") #right_join Prioriza las filas de la segunda base d
head(data_resultado2)
```

```
##       Country 2016Score 2015Score 2014Score 2013Score 2012Score ScoreDemo
## 1     Denmark        90        91        92        91        90      9.22
## 2 New Zealand        90        88        91        91        90      9.26
## 3     Finland        89        90        89        89        90      9.14
## 4      Sweden        88        89        87        89        88      9.39
## 5 Switzerland        86        86        86        85        86      9.03
## 6      Norway        85        87        86        86        85      9.87
##   Electoralprocessandpluralism Functioningofgovernment
## 1                        10.00                    9.29
## 2                        10.00                    9.29
## 3                        10.00                    8.93
## 4                         9.58                    9.64
## 5                         9.58                    9.29
## 6                        10.00                    9.64
##   Politicalparticipation Politicalculture Civilliberties     Regimetype
## 1                   8.33             9.38            9.12 Full democracy
## 2                   8.89             8.13           10.00 Full democracy
## 3                   8.33             8.75            9.71 Full democracy
## 4                   8.33            10.00            9.41 Full democracy
## 5                   7.78             9.38            9.12 Full democracy
## 6                  10.00            10.00            9.71 Full democracy
##    Region
## 1  Europe
## 2 Oceania
## 3  Europe
## 4  Europe
## 5  Europe
## 6  Europe
```

```
data_resultado3=inner_join(corru,demo,by="Country") #inner_join Prioriza las filas que COINCIDEN en amb
head(data_resultado3)
```

```
##       Country 2016Score 2015Score 2014Score 2013Score 2012Score ScoreDemo
## 1     Denmark        90        91        92        91        90      9.22
## 2 New Zealand        90        88        91        91        90      9.26
## 3     Finland        89        90        89        89        90      9.14
## 4      Sweden        88        89        87        89        88      9.39
## 5 Switzerland        86        86        86        85        86      9.03
```

```
## 6      Norway        85        87        86        86        85      9.87
##   Electoralprocessandpluralism Functioningofgovernment
## 1                        10.00                     9.29
## 2                        10.00                     9.29
## 3                        10.00                     8.93
## 4                         9.58                     9.64
## 5                         9.58                     9.29
## 6                        10.00                     9.64
##   Politicalparticipation Politicalculture Civilliberties     Regimetype
## 1                   8.33             9.38            9.12 Full democracy
## 2                   8.89             8.13           10.00 Full democracy
## 3                   8.33             8.75            9.71 Full democracy
## 4                   8.33            10.00            9.41 Full democracy
## 5                   7.78             9.38            9.12 Full democracy
## 6                  10.00            10.00            9.71 Full democracy
##    Region
## 1  Europe
## 2 Oceania
## 3  Europe
## 4  Europe
## 5  Europe
## 6  Europe
```

```
data_resultado4=full_join(corru,demo,by="Country") #full_join PPrioriza las filas que están en ambas fi
head(data_resultado4)
```

```
##       Country 2016Score 2015Score 2014Score 2013Score 2012Score ScoreDemo
## 1     Denmark        90        91        92        91        90      9.22
## 2 New Zealand        90        88        91        91        90      9.26
## 3     Finland        89        90        89        89        90      9.14
## 4      Sweden        88        89        87        89        88      9.39
## 5 Switzerland        86        86        86        85        86      9.03
## 6      Norway        85        87        86        86        85      9.87
##   Electoralprocessandpluralism Functioningofgovernment
## 1                        10.00                     9.29
## 2                        10.00                     9.29
## 3                        10.00                     8.93
## 4                         9.58                     9.64
## 5                         9.58                     9.29
## 6                        10.00                     9.64
##   Politicalparticipation Politicalculture Civilliberties     Regimetype
## 1                   8.33             9.38            9.12 Full democracy
## 2                   8.89             8.13           10.00 Full democracy
## 3                   8.33             8.75            9.71 Full democracy
## 4                   8.33            10.00            9.41 Full democracy
## 5                   7.78             9.38            9.12 Full democracy
## 6                  10.00            10.00            9.71 Full democracy
##    Region
## 1  Europe
## 2 Oceania
## 3  Europe
## 4  Europe
## 5  Europe
## 6  Europe
```

EJERCICIO 2:

Extraiga la tabla de los volcanes más altos del planeta del siguiente link: https://github.com/Alexanderbenit7/EAP2_2023-2/raw/main/data/data_ex4.xlsx

Le solicitan crear un código de limpieza para ordenar esa base de datos, sobre la cual se irán añadiendo casos en el futuro. El objetivo es utilizar la técnica de limpieza de su preferencia para separar los pies de altura, grados, minutos y segundos de cada una de las coordenadas de cada volcán. Luego de ello, eliminar las variables innecesarias y quedarse con una data limpia. La base debe verse así:

Nombre	País	Pies	GLA	MLA	SLA	GLO	MLO	SLO
Nevado Ojos del Salado	Argentina Chile	22 608	27	06	35	68	32	29
Monte Pissis	Argentina	22 293	27	45	16.6	68	47	55.8
Cerro Bonete Chico	Argentina	22 175	27	52	0	68	49	0
Nevado Tres Cruces Sur	Argentina Chile	22 139	27	05	53	68	46	41
Volcán Llullaillaco	Argentina Chile	22 109	24	43	47	68	32	45
Volcán Walther Penck	Argentina	21 843	27	12	0	68	33	0
Volcán Incahuasi	Argentina Chile	21 778	27	02	0	68	18	0
Nevado Tres Cruces Central	Argentina Chile	21 749	27	04	07	68	47	11
Volcán Tupungato	Argentina Chile	21 555	33	21	21	69	46	10.2
Nevado Sajama	Bolivia	21 463	18	06	14	68	52	53
Volcán Ata	Argentina Chile	21 328	27	10	55	68	34	34