OPSO79-1-UCSH2021

Visualización de datos con ggplot2 (y combinación de data frames)

05/11/2021

Transformación avanzada de datos

pendiente pivotear datos y combinar data frames

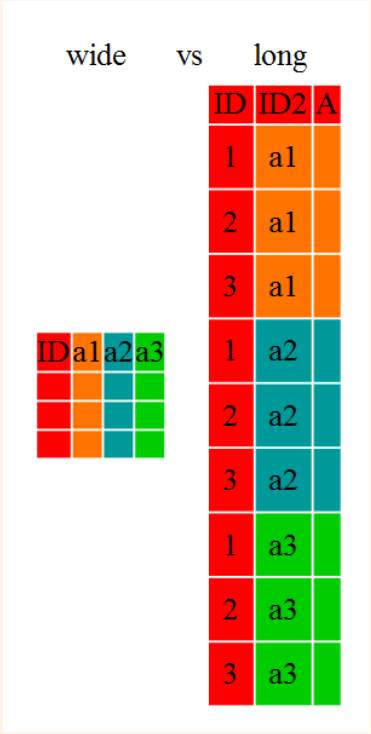
Pivotear los datos

Alargamiento o ensanchamiento de una data frame.

Alargamiento: incremento en el número de filas y decrecimiento del número de columnas

Ensanchamiento: incremento en el número de columnas y decrecimiento del número de filas

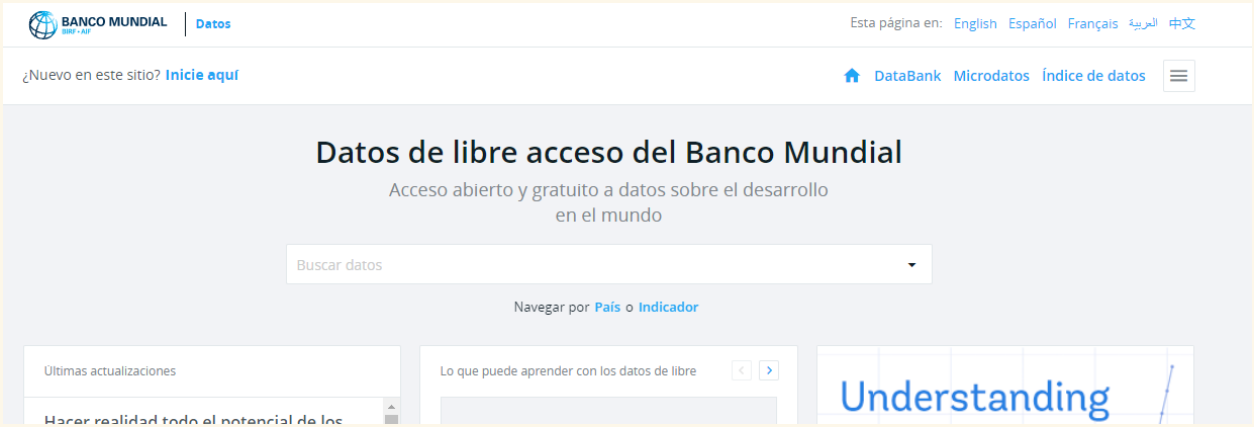
Para esto utilizaremos las funciones `pivot_wider()` y `pivot_longer()` del paquete `tidyr`



Pivotear los datos

Relevante para visualizar (ggplot2) y para trabajar datos importados

Por ejemplo, descarguemos los datos de Afganistán que usamos clases atrás. Esta vez sin trampa.



Pivotear los datos

¿Cómo vienen los datos?

```
afganistan <- readxl::read_excel("data/API_AFG_DS2_es_excel_v2_3162018.x
                                 skip = 3)
```

Country Code	2005	2012
AFG	NA	NA
AFG	3.910455	0.5579121
AFG	210.115520	771.2416744
AFG	NA	1.9269141
AFG	NA	0.0000000

¡Las variables vienen como filas!

Pivotear los datos

La solución es trasponer los datos.

Hacer que las filas pasen a ser variables (*to wider*), y columnas variables (*to longer*).

- Quitamos variables innecesarias

```
afganistan <- afganistan %>%
  select(-`Country Name`,-`Country Code`,-`Indicator Code`)
```

- Las variables que indicaban los años pasan a ser categorías de variable "anio"

```
afganistan <- afganistan %>% pivot_longer(2:ncol(afganistan),
                                          names_to = "anio")
```

Indicator Name	anio	value
Flujos oficiales netos procedentes de organismos de las Naciones Unidas, UNICEF (US$ a precios actuales)	1970	440000
Flujos oficiales netos procedentes de organismos de las Naciones Unidas, UNICEF (US$ a precios actuales)	1971	760000

Pivotear los datos

- Quitar filas repetidas para evitar errores

```
afganistan <- afganistan %>%
  distinct(`Indicator Name`,value,anio)
```

- Ensanchar la data

```
afganistan <- afganistan %>%
  tidyr::pivot_wider(names_from = `Indicator Name`,
                     values_from = value,
                     values_fn = {mean})
```

anio	Población rural	Población entre 0 y 14 años de edad, total
1960	8241132	3791398
1961	8373135	3892774
1962	8512057	3987207

Pivotear los datos

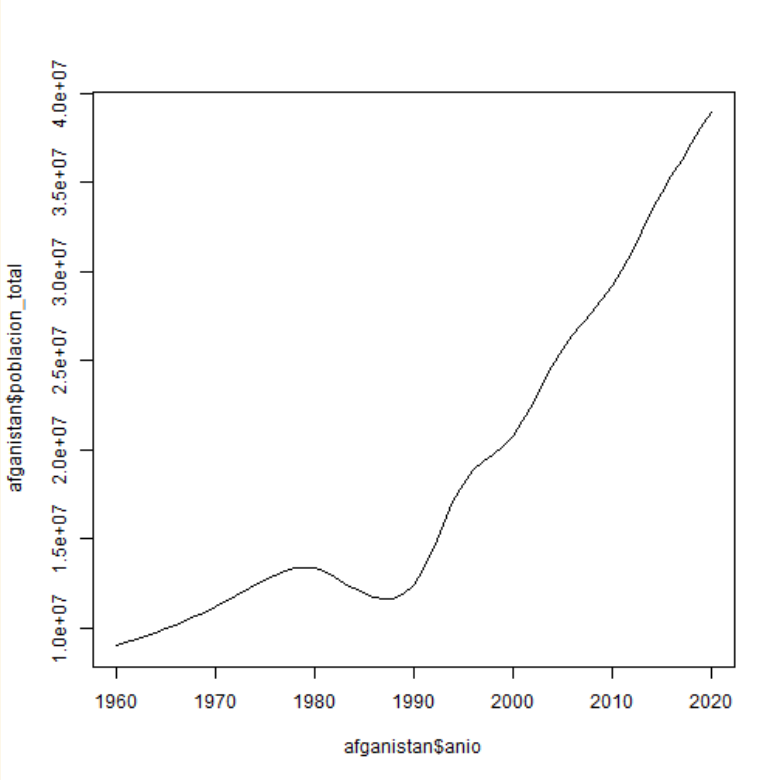
- Limpiar los nombres

```
# Limpiar los nombres
afganistan <- afganistan %>%
  janitor::clean_names()
```

anio	poblacion_rural	poblacion_entre_0_y_14_anos_de_edad_total
1960	8241132	3791398
1961	8373135	3892774
1962	8512057	3987207

Pivotear los datos

```
plot(afganistan$anio,afganistan$poblacion_total,type = "l")
```



Combinación de data frames

`funciones cbind(), rbind() y merge()`

Combinación de data frames

Descarguemos la siguiente base de datos con datos de algunos países de América Latina desde 1960 a 2020: DATA BANCO MUNDIAL

El archivo excel tiene 5 hojas, una para cada país (Argentina, Bolivia, Chile, Haití y México)

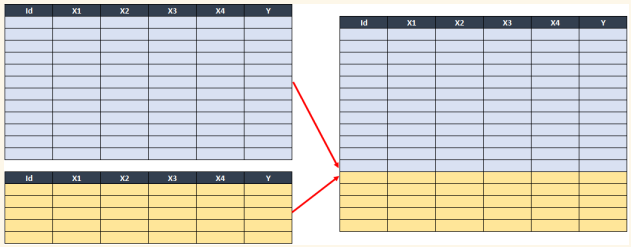
```
library(readxl)
arg <- read_excel("data/datos_bm/paises_banco_mundial.xlsx",sheet = 1)
bol <- read_excel("data/datos_bm/paises_banco_mundial.xlsx",sheet = 2)
chl <- read_excel("data/datos_bm/paises_banco_mundial.xlsx",sheet = 3)
hti <- read_excel("data/datos_bm/paises_banco_mundial.xlsx",sheet = 4)
mex <- read_excel("data/datos_bm/paises_banco_mundial.xlsx",sheet = 5)
```

```
head(arg,3)
```

```
## # A tibble: 3 x 5
##   year       pop lifeExp gdpPercap country
##   <chr>    <dbl>   <dbl>     <dbl> <chr>
## 1 1960  20481781    65.1       NA  Argentina
## 2 1961  20817270    65.2       NA  Argentina
## 3 1962  21153042    65.3     1156. Argentina
```

rbind()

Cuando tenemos las mismas variables, lo más sencillo es combinar pegando los datos uno bajo el otro.




```
data <- rbind(arg,chl,bol,hti,mex)
dim(data)
```

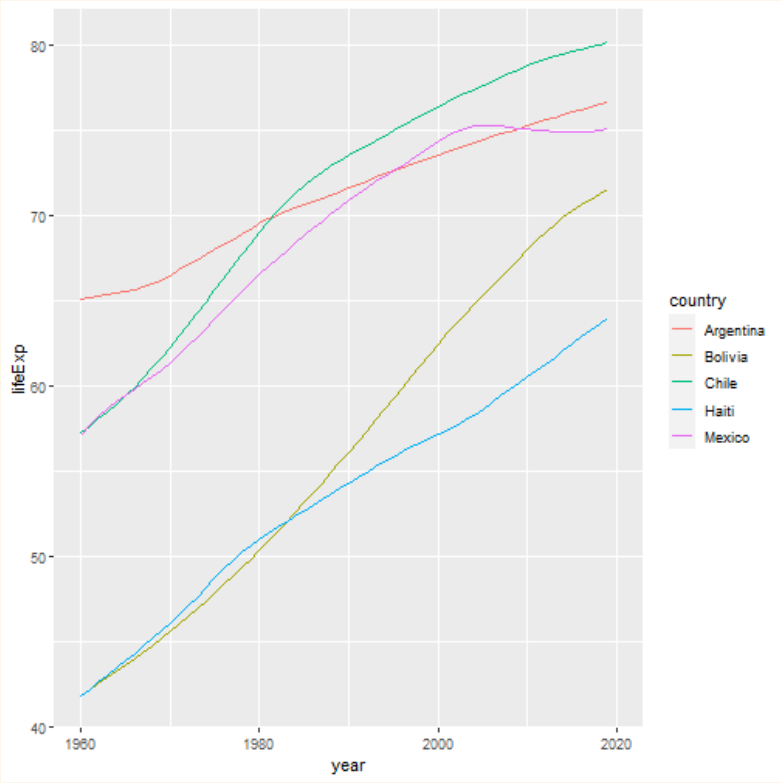
```
## [1] 305   5
```

```
table(data$country)
```

```
##
## Argentina   Bolivia     Chile     Haiti    Mexico
##        61        61        61        61        61
```

rbind()

Con más de un país en la data podemos distinguir distintas series temporales



cbind()

Tenemos los datos para el 2018 de la data recién cargada

```
data %>% filter(year==2018 & country!="Haiti") %>% arrange(country)
```

year	pop	lifeExp	gdpPercap	country
2018	44494502	76.520	11633.498	Argentina
2018	11353140	71.239	3548.591	Bolivia
2018	18729166	80.042	15888.144	Chile
2018	126190782	74.992	9686.514	Mexico

Y de latinobarómetro

idenpa	evangelicos
32	3.750000
68	15.666667
152	9.583333
484	2.500000

cbind()

```
cbind(bm2018,evangelicos)
```

```
##   year       pop lifeExp gdpPercap   country idenpa evangelicos
## 1 2018  44494502  76.520 11633.498 Argentina     32    3.750000
## 2 2018  11353140  71.239  3548.591   Bolivia     68   15.666667
## 3 2018  18729166  80.042 15888.144     Chile    152    9.583333
## 4 2018 126190782  74.992  9686.514    Mexico    484    2.500000
```

¿Cuál es el gran problema?

Los países estaban desordenados.

Ordenar países

```
evangelicos<- evangelicos %>%
                mutate(idenpa=case_when(idenpa==32 ~ "Argentina",
                                        idenpa==68 ~ "Bolivia",
                                        idenpa==152 ~ "Chile",
                                        idenpa==484 ~ "Mexico"))
```

cbind()

Ahora sí combinar:

```
cbind(bm2018,evangelicos) %>% select(-idenpa)
```

```
##   year       pop lifeExp gdpPercap   country evangelicos
## 1 2018  44494502  76.520 11633.498 Argentina    3.750000
## 2 2018  11353140  71.239  3548.591   Bolivia   15.666667
## 3 2018  18729166  80.042 15888.144     Chile    9.583333
## 4 2018 126190782  74.992  9686.514    Mexico    2.500000
```

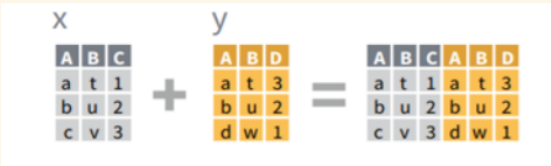
`cbind()` y `bind()` tienen limitaciones, solo combinan cuando existe el mismo número de filas y las variables se llaman igual.

`bind_rows()` y `col_rows()`, las versiones `dplyr` de las funciones vistas, son un poco más flexibles.

De todas formas `merge()` nos permitirá hacer más cosas, siendo fundamental el uso de variables llaves

merge()

La lógica de bind_rows()



La lógica de merge()

Para especificar el tipo de merge()

```
all.x=TRUE

all.y=TRUE

all=FALSE

all=TRUE
```

merge()

La función tiene 4 argumentos fundamentales

```
merge(x, y, by="key", all.x=FALSE)
```

- `x` es la data 1
- `y` es la data 2
- "key" es la variable llave usada para combinar
- con `all` indicamos si queremos mantener los valores de x, de y, de todas o de ninguna

merge()

Crear dos bases de datos

```
songs <- tibble(song = c("Come Together", "Dream On",
                         "Hello,Goodbye", "It's Not Unusual"),
                album  = c("Abbey Road", "Aerosmith",
                           "Magical Mystery Tour", "Along Came Jones"),
                first = c("John", "Steven", "Paul", "Tom"),
                last = c("Lennon", "Tyler", "McCartney", "Jones"))

albums <- tibble( album = c("A Hard Day's Night",
                            "Magical Mystery Tour", "Beggar's Banquet",
                           "Abbey Road", "Led Zeppelin IV",
                           "The Dark Side of the Moon",
                           "Aerosmith", "Rumours", "Hotel California"),
                  band = c("The Beatles", "The Beatles",
                           "The Rolling Stones",
                           "The Beatles", "Led Zeppelin",
                           "Pink Floyd", "Aerosmith",
                           "Fleetwood Mac", "Eagles"),
                year = c(1964,1967,1968,1969,1971,1973,1973,1977,1982))
```

merge()

¿Que variable tienen en común songs y albums?

album, por lo que será la llave.

```
merge(albums,songs, by="album", all = TRUE)
```

album	band	year	song	first	last
A Hard Day's Night	The Beatles	1964	NA	NA	NA
Abbey Road	The Beatles	1969	Come Together	John	Lennon
Aerosmith	Aerosmith	1973	Dream On	Steven	Tyler
Along Came Jones	NA	NA	It's Not Unusual	Tom	Jones
Beggar's Banquet	The Rolling Stones	1968	NA	NA	NA
Hotel California	Eagles	1982	NA	NA	NA

merge()

Probemos con `all=FALSE`

```
merge(albums,songs, by="album", all = FALSE)
```

album	band	year	song	first	last
Abbey Road	The Beatles	1969	Come Together	John	Lennon
Aerosmith	Aerosmith	1973	Dream On	Steven	Tyler
Magical Mystery Tour	The Beatles	1967	Hello,Goodbye	Paul	McCartney

¿Que sucedió?

Solo se mantienen las observaciones que simultáneamente están en X y en Y

merge()

Quedarse con todos los valores de x (`all.x=TRUE`)

```
merge(albums,songs, by="album", all.x = TRUE)
```

album	band	year	song	first	last
A Hard Day's Night	The Beatles	1964	NA	NA	NA
Abbey Road	The Beatles	1969	Come Together	John	Lennon
Aerosmith	Aerosmith	1973	Dream On	Steven	Tyler
Beggar's Banquet	The Rolling Stones	1968	NA	NA	NA
Hotel California	Eagles	1982	NA	NA	NA
Led Zeppelin IV	Led Zeppelin	1971	NA	NA	NA
Magical Mystery Tour	The Beatles	1967	Hello,Goodbye	Paul	McCartney
Rumours	Fleetwood Mac	1977	NA	NA	NA

merge()

Quedarse con todos los valores de y (`all.y=TRUE`)

```
merge(albums,songs, by="album", all.y = TRUE)
```

album	band	year	song	first	last
Abbey Road	The Beatles	1969	Come Together	John	Lennon
Aerosmith	Aerosmith	1973	Dream On	Steven	Tyler
Along Came Jones	NA	NA	It's Not Unusual	Tom	Jones
Magical Mystery Tour	The Beatles	1967	Hello,Goodbye	Paul	McCartney

merge()

Para cerrar

¿Podemos usar más de una llave?

Sí, con `by=c("var1","var2")`

¿Se puede combinar más de una data frame al mismo tiempo?

Sí, teóricamente infinitas hasta que colapse la memoria del pc:

```
Reduce(function(x, y) merge(x, y), list(x, y, z, etc))
```

Deben escribirse dentro de `list()`

Con comandos más avanzados se pueden leer y combinar todas las bases de datos de una carpeta del computador o del ambiente.

Recursos web utilizados

Xaringan: Presentation Ninja, de Yihui Xie. Para generar esta presentación.

Ilustraciones de Allison Horst

Para reforzar y seguir aprendiendo

Funciones merge() en R