Taller R - Rifa Valentina

1. Transformación avazanzada de data frames

26/01/2022

Contenidos

- Manipulación avanzada de data frames (funciones `pivot` y combinación de data frames)
- Manipulación de fechas: paquete `lubridate`
- Visualización de datos con `ggplot2`
- Estimar desde diseños muestrales complejos (paquete `survey` y `srvyr`)
- Hacer funciones, procesos iterativos e introducción a paquetes en R

Contenidos

- **Manipulación avanzada de data frames (funciones `pivot` y combinación de data frames)**
- Manipulación de fechas: paquete `lubridate`
- Visualización de datos con `ggplot2`
- Estimar desde diseños muestrales complejos (paquete `survey` y `srvyr`)
- Hacer funciones, procesos iterativos e introducción a paquetes en R
- Hoy veremos un extra de procesamiento: paquete `janitor`

1. Transformación avanzada de datos

Pivotear data frames

Introducción

Para elaborar gráficos *elegantes* en R es necesario saber transformar la data previamente

Por ejemplo,

- si queremos graficar N de hogares por región, no nos servirá una base de datos de personas.
- si queremos graficar mediante barras el porcentaje de personas que reciben mas y menos del sueldo mínimo, la variable numérica salario debe ser categorizada
- Si queremos graficar 2 variables, distinguiendo la relación por una tercera, necesitamos tener una base en formato *longer* (hacia abajo), no *wider* (hacia el lado)
- En palabras simples: para visualizar más de dos **series de tiempo**, es ideal pasar la data a formato **longer**

Para esto último son necesarias las funciones `pivot_wider()` y `pivot_longer`

Aplicaremos las funciones a datos del paquete `gapminder` y a datos del Banco Mundial.

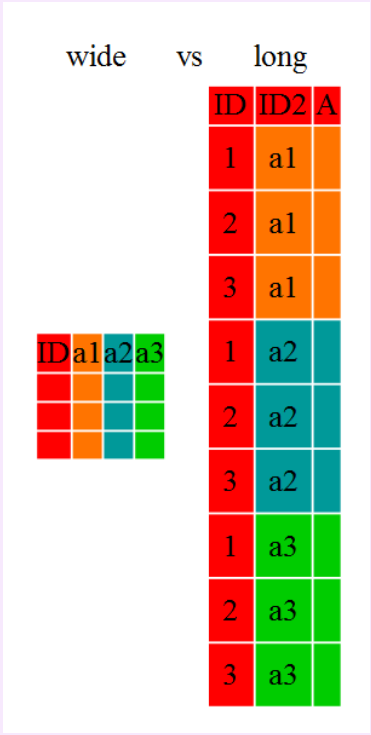
Pivotear los datos

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`



Función pivot_wider()

Esta función se utiliza para ordenar un dataframe de forma tal de mostrar categorías de una variable como columnas de un dataframe.

Incrementa el número de las columnas y disminuye el número de las filas.

Es útil para la presentación de cuadros de resumen con doble entrada.

sexo	posicion_politica	n
1	centro	3740
1	der	2161
1	izq	2245
1	ninguna	979
2	centro	3582
2	der	2187
2	izq	2327
2	ninguna	1415

Función pivot_wider()

Ahora vemos las categorías de sexo hacia la derecha

posicion_politica	1	2
centro	3740	3582
der	2161	2187
izq	2245	2327
ninguna	979	1415

Pasamos de un formato largo a uno ancho

```
library(tidyr)
```

```
data %>%
  filter(!is.na(posicion_politica)) %>%
  group_by(sexo,posicion_politica) %>%
  summarise(n=n()) %>%
  pivot_wider(names_from = sexo,
              values_from = n)
```

Función pivot_wider()

Básicamente dos argumentos:

- *names_from:* categorías que se quiere convertir en columnas
- *values_from:* columna desde la cual extraer los valores

Además, podemos usar el argumento `names_prefix` cuando tenemos números

```
data %>% filter(!is.na(posicion_politica)) %>%
  group_by(sexo,posicion_politica) %>%
  summarise(n=n()) %>%
  pivot_wider(names_from = sexo,
              values_from = n,
              names_prefix = "sexo_")
```

```
## # A tibble: 4 x 3
##   posicion_politica sexo_1 sexo_2
##   <chr>              <int>  <int>
## 1 centro              3740   3582
## 2 der                 2161   2187
## 3 izq                 2245   2327
## 4 ninguna              979   1415
```

Función pivot_longer()

Esta función se puede considerar como la opuesta a pivot_wider().

Esta función incrementa el número de filas y disminuye el número de columnas.

Los dataframes obtenidos por esta función son más fáciles de manipular, pero son poco presentables

```
df1 <- data.frame(region = c(1, 2),
                  hombres = c(100, 200),
                  mujeres = c(50, 300))
df1
```

```
##   region hombres mujeres
## 1      1     100      50
## 2      2     200     300
```

Función pivot_longer()

```
df1 %>%
  pivot_longer(cols = c(hombres,mujeres) )
```

```
## # A tibble: 4 x 3
##   region name    value
##    <dbl> <chr>   <dbl>
## 1      1 hombres   100
## 2      1 mujeres    50
## 3      2 hombres   200
## 4      2 mujeres   300
```

El argumento principal es cols:

- *cols:* columnas a las que se le aplicará la operación (que se convertirán en categorías de una nueva variable)

Función pivot_longer()

Además, se pueden especificar los nombres de las columnas "name" y "value"

- *names_to:* indica el nombre de la variable que será creada para "guardar" los nombres de las categorías.
- *values_to:* indica el nombre de la variable que será creada para "guardar" los valores asociados a las categorías.

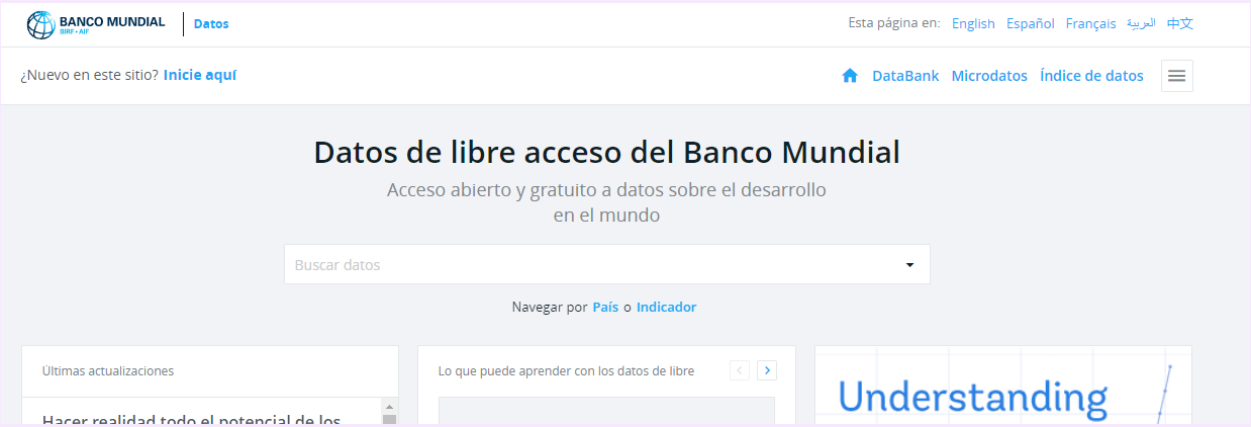
```
df1 %>%
  pivot_longer(cols = c(hombres,mujeres) ,
               names_to = "sexo", values_to = "total_sexo")
```

```
## # A tibble: 4 x 3
##   region sexo    total_sexo
##    <dbl> <chr>        <dbl>
## 1      1 hombres        100
## 2      1 mujeres         50
## 3      2 hombres        200
## 4      2 mujeres        300
```

Pivotear los datos

Relevante para visualizar (ggplot2) y para trabajar datos importados

Por ejemplo, descarguemos datos de Afganistán



Pivotear los datos

¿Cómo vienen los datos?

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

Country Code	Indicator Name	2005	2012
AFG	Exportaciones de productos de alta tecnología (US$ a precios actuales)	NA	NA
AFG	Exportaciones de mercadería hacia economías en el mundo árabe (% del total de exportaciones de mercadería)	3.910455	0.5579121
AFG	Índice del valor de las importaciones (2000 = 100)	210.115520	771.2416744
AFG	Seguro y servicios financieros (% de las importaciones de servicios comerciales)	NA	1.9269141

Pivotear los datos

La data no es un dato ordenado (tidy data)

¿Como graficamos el PIB de Afganistan si no es una variable? Solo podemos tabular años, lo que no tiene sentido:

```
table(afganistan$`1962`)
```

```
##
##       -1099999400 -24444431.1111111             -20000              -2400
##                 1                  1                  1                  1
## -11.1977715877437 -4.47154218553776                  0 0.039774951722225
##                 1                  1                  2                  1
```

Pivotear los datos

La solución es pivotear los datos. Hacer que las filas pasen a ser variables.

```
# Alargar la data
afganistan <- afganistan %>% pivot_longer(5:ncol(afganistan)) %>%
  select(-`Country Name`,`Country Code`,`Indicator Code`)
```

```
# Quitar filas repetidas para evitar errores
afganistan <- afganistan %>%
  distinct(`Indicator Name`,value,name)
```

```
# Ensanchar la data
afganistan <- afganistan %>%
  tidyr::pivot_wider(names_from = `Indicator Name`,
                                  values_from = value,
                                  values_fn = {sum})
```

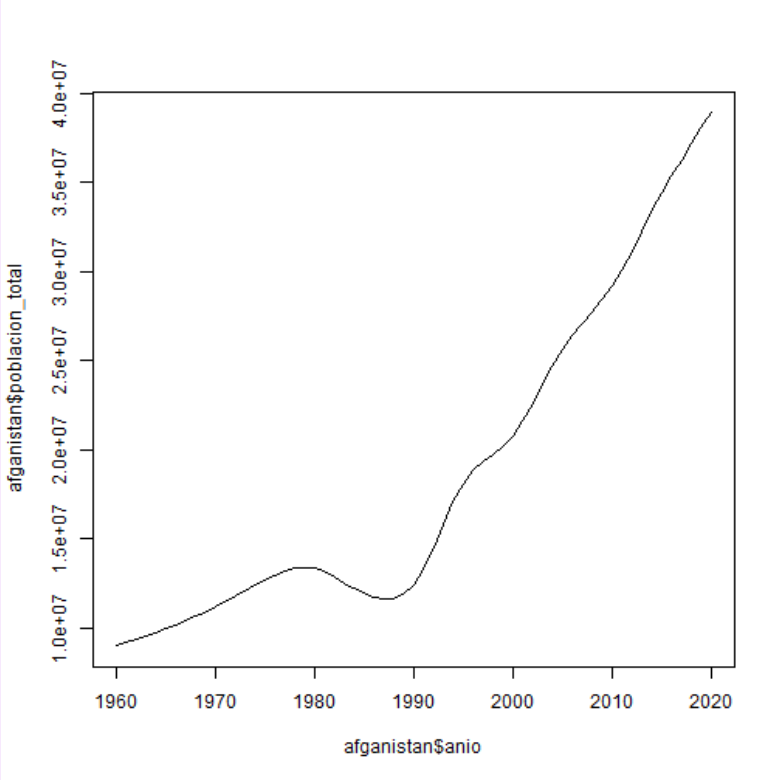
```
# Limpiar los nombres
afganistan <- afganistan %>%
  janitor::clean_names() %>% rename(anio=name)
```

Pivotear los datos

anio	ingreso_nacional_bruto_ing_us	poblacion_total
2012	20033093818	31161378
2013	20632806188	32269592
2014	20482514566	33370804
2015	20087077459	34413603
2016	18197299091	35383028
2017	19118263186	36296111
2018	18544615040	37171922
2019	19598008726	38041757
2020	19996141020	38928341

Visualizar una variable

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



¿Que diablos hicimos?

1. Alargar la data

```
afganistan <- afganistan %>% pivot_longer(5:ncol(afganistan))  %>%
  select(-`Country Name`,-`Country Code`,-`Indicator Code`) # y quitar l
```

```
## # A tibble: 6 x 3
##   `Indicator Name`                                      name  value
##   <chr>                                                 <chr> <dbl>
## 1 Índice del valor de las importaciones (2000 = 100) 1987  181.
## 2 Índice del valor de las importaciones (2000 = 100) 1988  164.
## 3 Índice del valor de las importaciones (2000 = 100) 1989  149.
## 4 Índice del valor de las importaciones (2000 = 100) 1990  170.
## 5 Índice del valor de las importaciones (2000 = 100) 1991  112
## 6 Índice del valor de las importaciones (2000 = 100) 1992   74.7
```

2. Quitar filas repetidas

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

```
## # A tibble: 6 x 3
##   `Indicator Name`                                      name  value
##   <chr>                                                 <chr> <dbl>
## 1 Índice del valor de las importaciones (2000 = 100) 1987  181.
## 2 Índice del valor de las importaciones (2000 = 100) 1988  164.
## 3 Índice del valor de las importaciones (2000 = 100) 1989  149.
## 4 Índice del valor de las importaciones (2000 = 100) 1990  170.
## 5 Índice del valor de las importaciones (2000 = 100) 1991  112
## 6 Índice del valor de las importaciones (2000 = 100) 1992   74.7
```

3. Ensanchar la data

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

```
## # A tibble: 6 x 3
##   name  `Índice del valor de las importaciones (2000 = 100)` `Población rur
##   <chr>                                                 <dbl>             <d
## 1 2009                                                   284.          21714
## 2 2010                                                   438.          22257
## 3 2011                                                   554.          22904
## 4 2012                                                   771.          23632
## 5 2013                                                   727.          24404
## 6 2014                                                   657.          25165
```

4. Limpiar los nombres

```
names(afganistan)[1:3]
```

```
## [1] "name"
## [2] "Exportaciones de productos de alta tecnología (US$ a precios actuales)
## [3] "Exportaciones de mercadería hacia economías en el mundo árabe (% del t
```

```
afganistan <- afganistan %>% janitor::clean_names() %>%
  rename(anio=name)
```

```
names(afganistan)[1:3]
```

```
## [1] "anio"
## [2] "exportaciones_de_productos_de_alta_tecnologia_us_a_precios_actuales"
## [3] "exportaciones_de_mercaderia_hacia_economias_en_el_mundo_arabe_percent_
```

Visualizar muchas variables

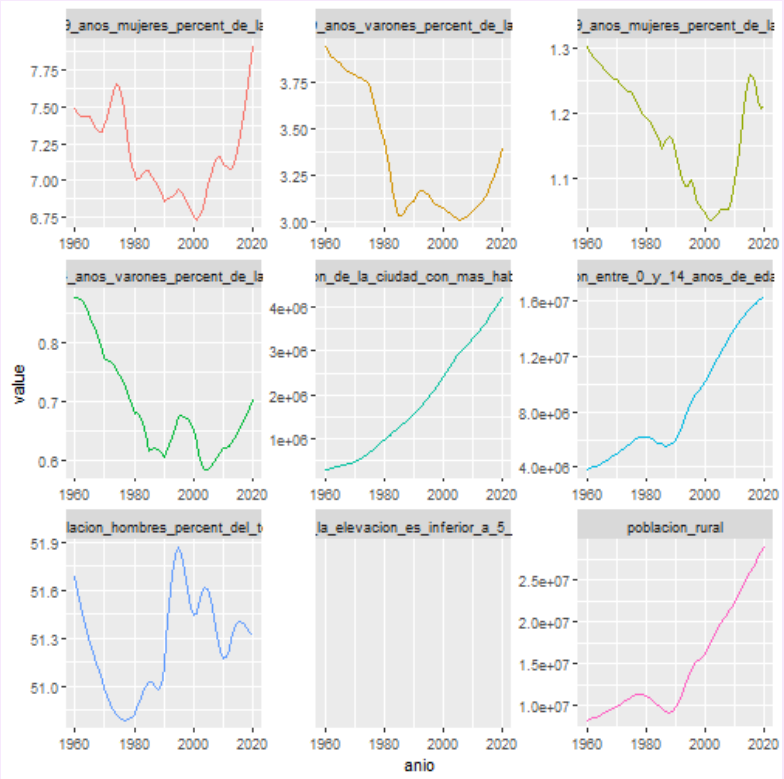
Primero alargar...

```
data <- afganistan %>%
  select(anio,starts_with("poblacion"))

data <- data[,c(1:10)] %>% pivot_longer(c(2:ncol(.)))
```

```
## # A tibble: 6 x 3
##   anio  name                                                                 
##   <chr> <chr>                                                                
## 1 1960  poblacion_rural                                                      8
## 2 1960  poblacion_de_65_a_69_anos_mujeres_percent_de_la_poblacion_fem~       1
## 3 1960  poblacion_de_25_a_29_anos_mujeres_percent_de_la_poblacion_fem~       7
## 4 1960  poblacion_entre_0_y_14_anos_de_edad_total                            3
## 5 1960  poblacion_de_la_ciudad_con_mas_habitantes                            2
## 6 1960  poblacion_que_vive_en_zonas_donde_la_elevacion_es_inferior_a_~      NA
```

…Luego visualizar



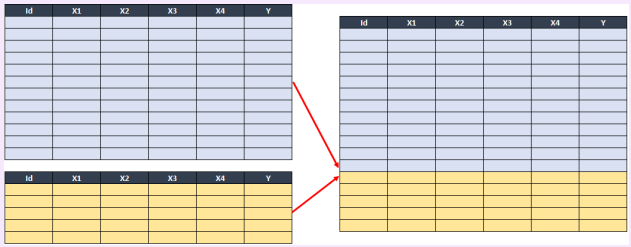
2. Combinar data frame

Combinación de data frames usando llaves en merge

Repaso: sin llaves

Para pegar columnas: `cbind()`

Para pegar filas, una bajo la otra `bind()`



Limitación: `rbind()` solo cuando las variables se llaman igual.

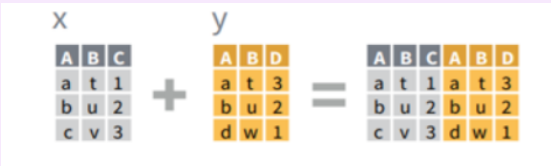
Por otro lado, `cbind()` combina cuando existe el mismo número de filas.

`bind_rows()` y `col_rows()` son las versiones `dplyr` más flexibles para la combinación.

Sin embargo, todas estas opciones pegan filas y columnas, sin considerar identificadores de las unidades (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 utilizados

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

Ilustraciones de Allison Horst

Capacitación INE tidy data

Funciones merge() en R