OPSO79-1-UCSH2021

Corrección prueba 2, Estadística descriptiva II y transformación avazanzada de data frames

29/10/2021

Transformación avanzada de datos

Un poco más de agrupación, pivotear y combinar data frames

Introducción

La sesión subsiguiente veremos en detalle como elaborar gráficos elegantes en R.

Antes es necesario revisar algunos últimos aspectos sobre tranformación de datos.

La clave para elaborar buenos gráficos en R es tener una data frame coherente con el gráfico que queremos

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)

Introducción

A continuación veremos herramientas que nos permitirán lidiar con estos y otros problemas:

- Funciones de agrupación (`group_by()`, `summarise()`).
- Funciones para pivotear la data (`pivot_longer()`, `pivot_wider()`).
- Funciones para combinar data (`merge()`, `rbind()` y `cbind()`)

Aplicaremos estas funciones a los datos del paquete Gapminder, a datos del Banco Mundial (de donde venían los de Afganistán), entre otros.

Agrupación de datos

`profundización función group_by()`

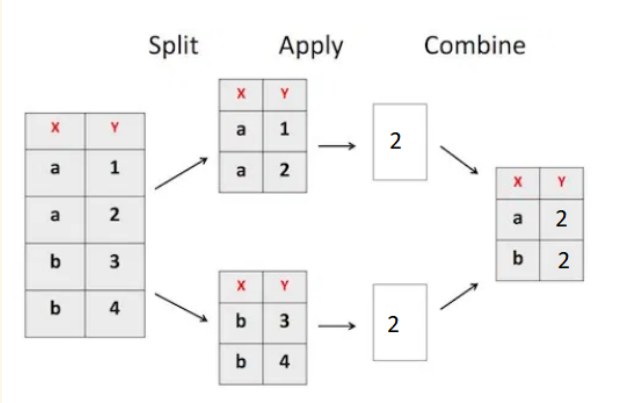
group_by() y summarise()

En conjunto nos permiten resumir información para cada grupo de una variable

Podemos obtener edad promedio por sexo, número de personas en cada región, ingresos por hogar, etcétera.

Estrategia split-apply-combine.

Esta estrategia sucede tras bambalinas (no la vemos). Solo observamos el resultados.



group_by() y summarise()

```
data <- readRDS("data/Latinobarometro_2020_Esp_Rds_v1_0.rds")
```

Conteo de frecuencias

```
data %>% group_by(sexo) %>% summarise(n=n())
```

```
## # A tibble: 2 x 2
##   sexo           n
##   <hvn_lbll> <int>
## 1 1           9667
## 2 2          10537
```

Obtención de estadísitcos para cada grupo

```
data %>% group_by(sexo) %>% summarise(edad=mean(edad))
```

```
## # A tibble: 2 x 2
##   sexo        edad
##   <hvn_lbll> <dbl>
## 1 1           41.6
```

group_by() y mutate()

Con `summarise` "perdemos" la data original. Esta es resumida a una más pequeña.

```
data2 <- data %>% group_by(sexo) %>% summarise(edad=mean(edad))
dim(data2)
```

```
## [1] 2 2
```

```
dim(data)
```

```
## [1] 20204   408
```

Pero en ocasiones queremos una medida de resumen sin perder la data, para poder generar nuevos cálculos.

La alternativa es agrupar sin resumir, sino que mutando la data.

```
data2 <- data %>% group_by(sexo) %>% mutate(edad_promedio=mean(edad))
dim(data2)
```

```
## [1] 20204   409
```

group_by() y mutate()

Veamos un pedazo de la nueva data

```
data2 %>% select(idenpa,sexo,edad,edad_promedio) %>% head()
```

```
## # A tibble: 6 x 4
## # Groups:   sexo [2]
##   idenpa     sexo       edad       edad_promedio
##   <hvn_lbll> <hvn_lbll> <hvn_lbll>         <dbl>
## 1 32         2          63                  40.4
## 2 32         1          24                  41.6
## 3 32         1          20                  41.6
## 4 32         2          54                  40.4
## 5 32         1          38                  41.6
## 6 32         2          62                  40.4
```

Edad promedio aparece en cada observación.

Es el promedio de la edad del grupo (sexo) al que pertenece la observación.

En este caso solo hay valores 41,6 (para los hombres) y 40,4 (para las mujeres)

group_by() y mutate()

¿Cuál es la utilidad?

Sirve para el procesamiento de datos más que para el análisis.

Por ejemplo, identificar casos extraños dentro de un conjunto para luego editarlos.

Países que pertenecen a continentes pobres pero que son **MUY** ricos:

```
library(gapminder)

paises_1972 <- gapminder %>%
  filter(year==1972 ) %>%
  group_by(continent) %>%
  mutate(gdpPercap_continente=quantile(gdpPercap,0.90)) %>%
  ungroup()
```

group_by() y mutate()

```
paises_1972  %>%
  filter(continent %in% c("Africa","Americas") &
         gdpPercap>gdpPercap_continente) %>%
  arrange(-gdpPercap) %>% select(-year,-continent)
```

country	lifeExp	pop	gdpPercap	gdpPercap_continente
United States	71.340	209896000	21806.036	10080.371
Libya	52.773	2183877	21011.497	4139.005
Canada	72.880	22284500	18970.571	10080.371
Gabon	48.690	537977	11401.948	4139.005
Venezuela	65.712	11515649	10505.260	10080.371
South Africa	53.696	23935810	7765.963	4139.005
Angola	37.928	5894858	5473.288	4139.005
Reunion	64.274	461633	5047.659	4139.005
Algeria	54.518	14760787	4182.664	4139.005

Pivotear los datos

`funciones pivot_wider() y pivot_longer()`

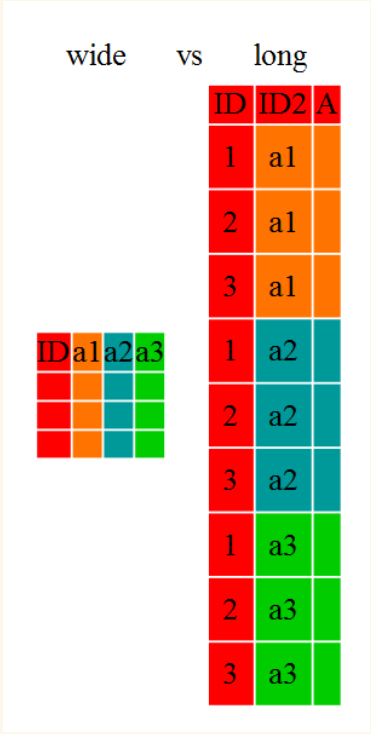
Pivotear los datos

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

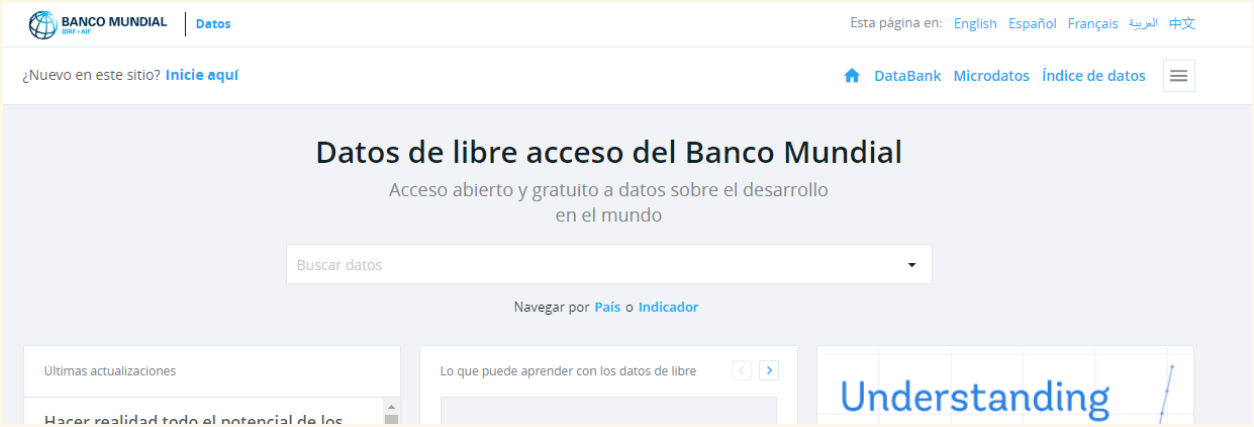
Para esto utilizaremos las funciones `pivot_wider()` y `pivot_longer()` del paquete `tidyr`



Pivotear los datos

Relevante para visualizar (la próxima semana lo entenderemos) 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/afganistan.xlsx")
```

```
## # A tibble: 5 x 3
##   `Indicator Name`                                                `2007`   `
##   <chr>                                                            <dbl>    
## 1 Internally displaced persons, new displacement associated wit~   NA     1
## 2 Mercaderías exportadas hacia economías en desarrollo en Europ~   11.3  NA
## 3 Índice de términos netos de intercambio (2000 = 100)            127.     1
## 4 Mercaderías importadas desde economías de ingreso alto (% del~   15.2    4
## 5 Participación de líneas arancelarias con máximos internaciona~   16.0  NA
```

¡Las variables vienen como filas! (lo contrario a una data tidy u ordenada)

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

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

Pivotear los datos

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

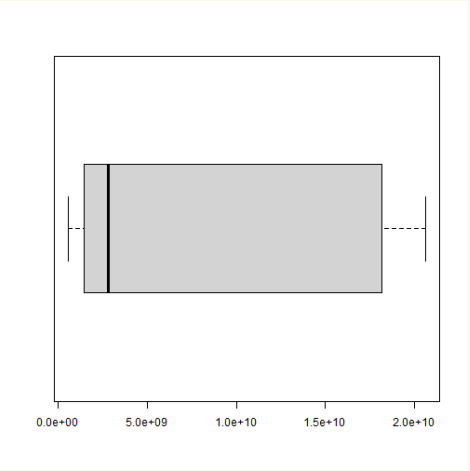
Veamos el código y luego explicamos:

```
# Alargar la data
afganistan <- afganistan %>% tidyr::pivot_longer(3:63) %>%
  select(-`Country Name`)

# 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

```
## # A tibble: 5 x 3
##   anio  ingreso_nacional_bruto_ing_us poblacion_total
##   <chr>                         <dbl>           <dbl>
## 1 2016                  18197299091.        35383028
## 2 2017                  19118263186.        36296111
## 3 2018                  18544615040.        37171922
## 4 2019                  19598008726.        38041757
## 5 2020                  19996141020.        38928341
```

```
boxplot(afganistan$ingreso_nacional_bruto_ing_us, horizontal = TRUE)
```



Pivotear los datos

pivot_longer()

```
data %>% pivot_longer(c(col1, col2, col3))
```

Se especifican las columnas que ahora pasan a ser filas.

pivot_wider()

```
data %>% pivot_wider(names_from = col1,
                     values_from = col2)
```

Se crean varias columnas.

Se debe especificar de que variable se tomarán los nuevos nombres y los nuevos valores.

Volvamos a la transformación de Afganistán. Paso por paso.

Pivotear los datos

También podemos ocupar las funciones para hacer tablas de contingencia.

Agrupar por dos variables, y luego una pasarlas a columnas.

```
casen %>%
  group_by(regiones, sexo) %>%
  summarise(n=n()) %>%
  pivot_wider(names_from = region, values_from = n)
```

Combinación de data frames

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

Recursos web utilizados

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

Ilustraciones de Allison Horst

Para reforzar y seguir aprendiendo