

Unir y Ordenar los datos

Curso: Manejo de datos y reportería con R

Néstor Montaña

Sociedad Ecuatoriana de Estadística

Enero-2021



Nota:

Con *Alt + F* o *Option + F* puede hacer que estas diapositivas ocupen todo el navegador (es decir que se ignore el aspecto de diapositiva que tiene por default la presentación)

R - cargar librerías

```
#### LIBRERIAS -----  
library(openxlsx) # para importar desde excel  
library(tidyverse) # manipulacion de datos  
library(ggplot2) # graficos  
library(magrittr) # %>%  
library(lubridate) # Manipulacion de fechas  
library(stringr) # Manipulacion de texto
```

Ejemplo: Transacciones bancarias

El Banco del Pacífico requiere mejorar los tiempos de atención al cliente en ventanilla, para ello ha recolectado esta información anónimamente para cada cajero y transacción realizada.

Le suministran un excel con dos hojas:

1. Tiene los datos de las transacciones, columnas: Sucursal, Cajero, ID_Transaccion, Transaccion, Tiempo_Servicio_seg, Nivel de satisfacción, Monto de la transaccion.
2. Otra hoja que indica si en la sucursal se ha puesto o no el nuevo sistema.



Ejemplo - Importar

Importar las hojas del excel dado

```
# Leer el archivo de excel y asignarlo al objeto data_banco  
data_banco <- read.xlsx(xlsxFile = "Data/Data_Banco.xlsx", sheet = "Data")  
data_sucursal <- read.xlsx(xlsxFile = "Data/Data_Banco.xlsx", sheet = "Data_Sucursal")
```



Ejemplo - Convertir a tibbles (un dataframe mejorado):

```
# Convertir el data_banco a un tibble  
data_banco <- as_tibble( data_banco)  
# Convertir el data_sucursal a un tibble  
data_sucursal <- as_tibble(data_sucursal)
```

Ejemplo - Manipulación de datos

Lo primero que necesitamos es corregir los tipos de datos, nótese que *Monto* tiene una mezcla de "," y ".".

```
data_banco <- data_banco %>%  
  mutate( Monto= str_replace(Monto, pattern = ",", replacement = ".") ) %>%  
  mutate(Sucursal= as.character(Sucursal),  
         Cajero = as.character(Cajero),  
         Satisfaccion = parse_factor(Satisfaccion,  
                                     levels= c('Muy Malo', 'Malo', 'Regular',  
                                               'Bueno', 'Muy Bueno')),  
         Monto= parse_number(Monto, locale = locale(decimal_mark = ".")))
```

Ejemplo - Manipulación de datos

Con todo esto hemos llegado a:

```
# Mostrar estructura del data_banco  
glimpse(data_banco)
```

```
## Rows: 24,299  
## Columns: 7  
## $ Sucursal      <chr> "62", "62", "62", "62", "62", "62", "62", "62", "62", ...  
## $ Cajero        <chr> "4820", "4820", "4820", "4820", "4820", "4820", "4820...  
## $ ID_Transaccion <chr> "2", "2", "2", "2", "2", "2", "2", "2", "2", "2", "2"...  
## $ Transaccion   <chr> "Cobro/Pago (Cta externa)", "Cobro/Pago (Cta externa)...  
## $ Tiempo_Servicio_seg <dbl> 311, 156, 248, 99, 123, 172, 140, 247, 183, 91, 327, ...  
## $ Satisfaccion  <fct> Muy Bueno, Malo, Regular, Regular, Muy Bueno, Bueno, ...  
## $ Monto         <dbl> 2889.30, 1670.69, 3172.49, 1764.92, 1835.69, 2165.42,...
```


Manipulación de datos - Unir Datos

Curso: Manejo de datos y reportería con R

Néstor Montaña

Unir Datos

Se va a crear unos data.frame simples para entender la idea detrás de los comandos a usar

```
# Crear un Data frame
df_1 <- data.frame(
  Nombre= c('Ana', 'Berni', 'Carlos', 'Daniel', 'Ericka'),
  Edad = c(20,19,20,19,18),
  Ciudad= factor(c('Gye', 'Uio', 'Cue', 'Gye', 'Cue')) )
df_1
```

```
##   Nombre Edad Ciudad
## 1   Ana   20    Gye
## 2  Berni   19    Uio
## 3 Carlos   20    Cue
## 4 Daniel   19    Gye
## 5 Ericka   18    Cue
```

Unir Datos

```
# Crear un Data frame
df_2 <- data.frame(
  Nombre= c('Fulton', 'Gilda'),
  Ciudad= factor(c('Mach', 'Gye')) ,
  Edad = c(21,18)
)
df_2
```

```
##   Nombre Ciudad Edad
## 1  Fulton   Mach   21
## 2   Gilda    Gye   18
```

Unir Datos

```
# Crear un Data frame
df_3 <- data.frame(
  Estado_Civil= c('S', 'D', "S", "C", "D")
)
df_3
```

```
##   Estado_Civil
## 1             S
## 2             D
## 3             S
## 4             C
## 5             D
```

Unir filas

Para unir filas se usa `bind_rows`

```
# Aumentar filas  
bind_rows(df_1, df_2)
```

```
##   Nombre Edad Ciudad  
## 1    Ana   20    Gye  
## 2  Berni   19    Uio  
## 3 Carlos   20    Cue  
## 4 Daniel   19    Gye  
## 5 Ericka   18    Cue  
## 6 Fulton   21  Mach  
## 7  Gilda   18    Gye
```

Unir filas

Para unir filas se usa `bind_rows`

```
# Aumentar filas  
bind_rows(df_1, df_3)
```

##	Nombre	Edad	Ciudad	Estado_Civil
## 1	Ana	20	Gye	<NA>
## 2	Berni	19	Uio	<NA>
## 3	Carlos	20	Cue	<NA>
## 4	Daniel	19	Gye	<NA>
## 5	Ericka	18	Cue	<NA>
## 6	<NA>	NA	<NA>	S
## 7	<NA>	NA	<NA>	D
## 8	<NA>	NA	<NA>	S
## 9	<NA>	NA	<NA>	C
## 10	<NA>	NA	<NA>	D

```
# rbind(df_1, df_3) #ERROR
```

Unir filas

Aumentar columna que indica origen de la fila

```
# Aumentar columna que indica origen de la fila  
bind_rows(list(df_1, df_2), .id = "id")
```

##	id	Nombre	Edad	Ciudad
## 1	1	Ana	20	Gye
## 2	1	Berni	19	Uio
## 3	1	Carlos	20	Cue
## 4	1	Daniel	19	Gye
## 5	1	Ericka	18	Cue
## 6	2	Fulton	21	Mach
## 7	2	Gilda	18	Gye

Unir filas

Aumentar columna que indica origen de la fila

```
# Aumentar columna que indica origen de la fila  
bind_rows(list(data_1= df_1, data_2= df_2), .id = "DF_ORIGEN")
```

##	DF_ORIGEN	Nombre	Edad	Ciudad
## 1	data_1	Ana	20	Gye
## 2	data_1	Berni	19	Uio
## 3	data_1	Carlos	20	Cue
## 4	data_1	Daniel	19	Gye
## 5	data_1	Ericka	18	Cue
## 6	data_2	Fulton	21	Mach
## 7	data_2	Gilda	18	Gye

Unir filas

Bind más de dos dataframes

```
# Bind más de dos dataframes
bind_rows(list(df_1= df_1, df_2= df_2, df_3= df_3), .id = "DF_ORIGEN")
```

##	DF_ORIGEN	Nombre	Edad	Ciudad	Estado_Civil
## 1	df_1	Ana	20	Gye	<NA>
## 2	df_1	Berni	19	Uio	<NA>
## 3	df_1	Carlos	20	Cue	<NA>
## 4	df_1	Daniel	19	Gye	<NA>
## 5	df_1	Ericka	18	Cue	<NA>
## 6	df_2	Fulton	21	Mach	<NA>
## 7	df_2	Gilda	18	Gye	<NA>
## 8	df_3	<NA>	NA	<NA>	S
## 9	df_3	<NA>	NA	<NA>	D
## 10	df_3	<NA>	NA	<NA>	S
## 11	df_3	<NA>	NA	<NA>	C
## 12	df_3	<NA>	NA	<NA>	D

Unir columnas

Para unir columnas se usa `bind_cols`

```
# Bind más de dos dataframes  
bind_cols(df_1, df_3)
```

##		Nombre	Edad	Ciudad	Estado_Civil
##	1	Ana	20	Gye	S
##	2	Berni	19	Uio	D
##	3	Carlos	20	Cue	S
##	4	Daniel	19	Gye	C
##	5	Ericka	18	Cue	D



Unir columnas

Para unir columnas se usa `bind_cols`, se requiere que los datos tengan el mismo largo

```
# Bind más de dos dataframes  
bind_cols(df_1, df_2) #ERROR
```

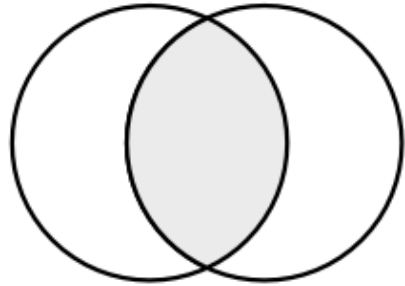


Unir datos - Merge|Join|Buscarv

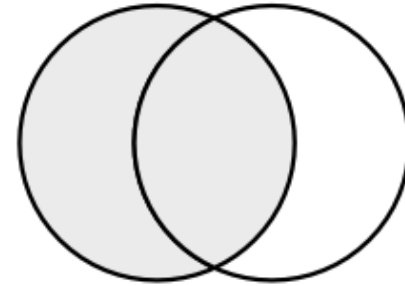
- Se tienen dos data.frames con columnas o variables que hacen las veces de “key” o “id” de los mismos
- Se desea agregar al primer conjunto el contenido del segundo conjunto de datos si y sólo si el “key” o “id” del segundo conjunto corresponde con el “key” o “id” del primer conjunto de datos.
- Parecido al Buscarv y Vlookup de excel
- Equivalente al Join de Bases de datos

Unir datos - Merge|Join|Buscar

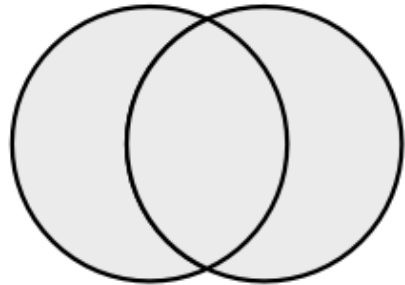
Entendiendo los tipos de Join



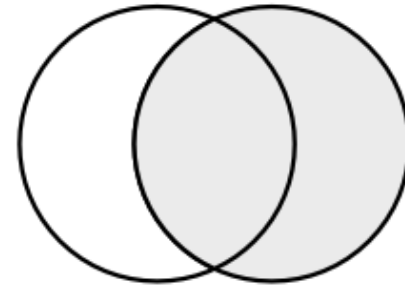
`inner_join(x, y)`



`left_join(x, y)`



`full_join(x, y)`



`right_join(x, y)`

Unir datos - Merge|Join|Buscar

Nuevo data.frame

```
df_6 <- data.frame(A= c('Ana', 'Daniel', 'Jose'), B= c(100,200,300))  
df_6
```

```
##           A    B  
## 1      Ana 100  
## 2 Daniel 200  
## 3   Jose 300
```

Unir datos - Merge|Join|Buscarv

En base R se usaba el comando Merge

```
# Inner Join
merge(x= df_1, y= df_6, by.x= 'Nombre', by.y= 'A')
# Right Join
merge(x= df_1, y= df_6, by.x= 'Nombre', by.y= 'A', all.x= TRUE)
# Left Join
merge(x= df_1, y= df_6, by.x= 'Nombre', by.y= 'A', all.y= TRUE)
# Full Join
merge(x= df_1, y= df_6, by.x= 'Nombre', by.y= 'A', all= TRUE)
```

Unir datos - Inner Join

```
df_1 %>% inner_join(df_6, by = c("Nombre"="A"))
```

```
##   Nombre Edad Ciudad    B  
## 1    Ana   20    Gye 100  
## 2 Daniel   19    Gye 200
```


Unir datos - Left Join

```
df_1 %>% left_join(df_6, by = c("Nombre"="A"))
```

##	Nombre	Edad	Ciudad	B
## 1	Ana	20	Gye	100
## 2	Berni	19	Uio	NA
## 3	Carlos	20	Cue	NA
## 4	Daniel	19	Gye	200
## 5	Ericka	18	Cue	NA

Unir datos - Right Join

```
df_1 %>% right_join(df_6, by = c("Nombre"="A"))
```

```
##   Nombre Edad Ciudad    B
## 1    Ana   20    Gye 100
## 2 Daniel   19    Gye 200
## 3   Jose   NA   <NA> 300
```

Unir datos - Full Join

```
df_1 %>% full_join(df_6, by = c("Nombre"="A"))
```

```
##   Nombre Edad Ciudad    B
## 1    Ana   20    Gye 100
## 2   Berni  19    Uio  NA
## 3  Carlos  20    Cue  NA
## 4 Daniel  19    Gye 200
## 5 Ericka  18    Cue  NA
## 6   Jose  NA   <NA> 300
```

Unir datos - Merge|Join|Buscarv

Vamos a duplicar un Valor en df_6 y a replicar los Joins para revisar qué sucede cuando se tiene "key" no únicos

```
df_6 <- data.frame(A= c('Ana', 'Daniel', 'Jose', 'Ana'), B= c(100,200,300, 110))  
df_6
```

```
##      A    B  
## 1   Ana 100  
## 2 Daniel 200  
## 3   Jose 300  
## 4   Ana 110
```

Unir datos - Inner Join

Nótese la repetición de "Ana"

```
df_1 %>% inner_join(df_6, by = c("Nombre"="A"))
```

##	Nombre	Edad	Ciudad	B
## 1	Ana	20	Gye	100
## 2	Ana	20	Gye	110
## 3	Daniel	19	Gye	200

Unir datos - Left Join

Nótese la repetición de "Ana"

```
df_1 %>% left_join(df_6, by = c("Nombre"="A"))
```

##	Nombre	Edad	Ciudad	B
## 1	Ana	20	Gye	100
## 2	Ana	20	Gye	110
## 3	Berni	19	Uio	NA
## 4	Carlos	20	Cue	NA
## 5	Daniel	19	Gye	200
## 6	Ericka	18	Cue	NA



Unir datos - Right Join

Nótese la repetición de "Ana"

```
df_1 %>% right_join(df_6, by = c("Nombre"="A"))
```

##	Nombre	Edad	Ciudad	B
## 1	Ana	20	Gye	100
## 2	Ana	20	Gye	110
## 3	Daniel	19	Gye	200
## 4	Jose	NA	<NA>	300

Unir datos - Full Join

Nótese la repetición de "Ana"

```
df_1 %>% full_join(df_6, by = c("Nombre"="A"))
```

##	Nombre	Edad	Ciudad	B
## 1	Ana	20	Gye	100
## 2	Ana	20	Gye	110
## 3	Berni	19	Uio	NA
## 4	Carlos	20	Cue	NA
## 5	Daniel	19	Gye	200
## 6	Ericka	18	Cue	NA
## 7	Jose	NA	<NA>	300

Unir datos

Y ¿si repetimos "Ana" en el df_1?

Unir datos

Y ¿si repetimos "Ana" en el df_1?

```
df_1 %>%  
  bind_rows( data.frame(Nombre="Ana", Edad= 42, Ciudad= 'Cue')) %>%  
  full_join(df_6, by = c("Nombre"="A"))
```

##	Nombre	Edad	Ciudad	B
## 1	Ana	20	Gye	100
## 2	Ana	20	Gye	110
## 3	Berni	19	Uio	NA
## 4	Carlos	20	Cue	NA
## 5	Daniel	19	Gye	200
## 6	Ericka	18	Cue	NA
## 7	Ana	42	Cue	100
## 8	Ana	42	Cue	110
## 9	Jose	NA	<NA>	300

Filterin Joins

`semi_join(x, y)` da todas las observaciones en x que tienen al menos una coincidencia en y.

```
df_1 %>% semi_join(df_6, by = c("Nombre"="A"))
```

```
##   Nombre Edad Ciudad
## 1    Ana   20    Gye
## 2 Daniel   19    Gye
```



Filterin Joins

`anti_join(x, y)` da todas las observaciones en `x` que NO tienen coincidencia en `y`.

```
df_1 %>% anti_join(df_6, by = c("Nombre"="A"))
```

```
##   Nombre Edad Ciudad
## 1  Berni   19    Uio
## 2 Carlos   20    Cue
## 3 Ericka   18    Cue
```



Join - Ejemplo transacciones bancarias

¿Qué join necesitamos realizar con nuestros datos?

```
View(data_banco)  
View(data_sucursal)
```



Join - Ejemplo transacciones bancarias

Esto nos dará error, ¿Qué falta?

```
data_banco %>%  
  left_join(data_sucursal, by= c("Sucursal"= "ID_Sucursal"))
```

Join - Ejemplo transacciones bancarias

Esto nos dará error, ¿Qué falta?

R: Corregir tipo de datos

```
data_sucursal <- data_sucursal %>%  
  mutate(ID_Sucursal= as.character(ID_Sucursal))  
data_sucursal
```

```
## # A tibble: 5 x 3  
##   ID_Sucursal Sucursal      Nuevo_Sistema  
##   <chr>        <chr>        <chr>  
## 1 62          Riocentro Sur No  
## 2 85          Centro        Si  
## 3 267         Alborada       Si  
## 4 443         Mall del Sol   Si  
## 5 586         Via Daule      No
```

Join - Ejemplo transacciones bancarias

Nótese el problema con el nombre "Sucursal"

```
data_banco %>%
  left_join(data_sucursal, by= c("Sucursal"= "ID_Sucursal"))
```

```
## # A tibble: 24,299 x 9
##   Sucursal Cajero ID_Transaccion Transaccion Tiempo_Servicio~ Satisfaccion Monto
##   <chr>    <chr>   <chr>          <chr>          <dbl> <fct>      <dbl>
## 1 62      4820     2      Cobro/Pago~      311 Muy Bueno   2889.
## 2 62      4820     2      Cobro/Pago~      156 Malo        1671.
## 3 62      4820     2      Cobro/Pago~      248 Regular     3172.
## 4 62      4820     2      Cobro/Pago~       99 Regular     1765.
## 5 62      4820     2      Cobro/Pago~      123 Muy Bueno   1836.
## 6 62      4820     2      Cobro/Pago~      172 Bueno      2165.
## 7 62      4820     2      Cobro/Pago~      140 Regular     1305.
## 8 62      4820     2      Cobro/Pago~      247 Bueno      4080.
## 9 62      4820     2      Cobro/Pago~      183 Muy Bueno   2541.
## 10 62     4820     2      Cobro/Pago~       91 Muy Bueno   2219.
## # ... with 24,289 more rows, and 2 more variables: Sucursal.y <chr>,
## #   Nuevo_Sistema <chr>
```


Join - Ejemplo transacciones bancarias

Cambiamos el nombre y ya reemplazamos el data_banco con este resultado

```
data_banco <- data_banco %>%
  rename("ID_Sucursal"="Sucursal") %>%
  left_join(data_sucursal, by= c("ID_Sucursal"))
data_banco
```

```
## # A tibble: 24,299 x 9
##   ID_Sucursal Cajero ID_Transaccion Transaccion Tiempo_Servicio~ Satisfaccion Monto
##   <chr>      <chr>   <chr>          <chr>          <dbl> <fct>      <dbl>
## 1 62        4820     2          Cobro/Pago~      311 Muy Bueno   2889.
## 2 62        4820     2          Cobro/Pago~      156 Malo       1671.
## 3 62        4820     2          Cobro/Pago~      248 Regular    3172.
## 4 62        4820     2          Cobro/Pago~       99 Regular    1765.
## 5 62        4820     2          Cobro/Pago~      123 Muy Bueno   1836.
## 6 62        4820     2          Cobro/Pago~      172 Bueno     2165.
## 7 62        4820     2          Cobro/Pago~      140 Regular    1305.
## 8 62        4820     2          Cobro/Pago~      247 Bueno     4080.
## 9 62        4820     2          Cobro/Pago~      183 Muy Bueno   2541.
## 10 62       4820     2          Cobro/Pago~       91 Muy Bueno   2219.
## # ... with 24,289 more rows, and 2 more variables: Sucursal <chr>,
## #   Nuevo_Sistema <chr>
```

Ejemplo transacciones bancarias

El data_banco ahora es

```
glimpse(data_banco)
```

```
## Rows: 24,299
## Columns: 9
## $ ID_Sucursal      <chr> "62", "62", "62", "62", "62", "62", "62", "62", "62",...
## $ Cajero           <chr> "4820", "4820", "4820", "4820", "4820", "4820", "4820", "4820...
## $ ID_Transaccion   <chr> "2", "2", "2", "2", "2", "2", "2", "2", "2", "2", "2"...
## $ Transaccion      <chr> "Cobro/Pago (Cta externa)", "Cobro/Pago (Cta externa)...
## $ Tiempo_Servicio_seg <dbl> 311, 156, 248, 99, 123, 172, 140, 247, 183, 91, 327, ...
## $ Satisfaccion     <fct> Muy Bueno, Malo, Regular, Regular, Muy Bueno, Bueno, ...
## $ Monto            <dbl> 2889.30, 1670.69, 3172.49, 1764.92, 1835.69, 2165.42,...
## $ Sucursal         <chr> "Riocentro Sur", "Riocentro Sur", "Riocentro Sur", "R...
## $ Nuevo_Sistema    <chr> "No", "No", "No", "No", "No", "No", "No", "No", "No", "No",...
```

Tidy: Ordenar los datos

Curso: Manejo de datos y reportería con R

Néstor Montaña

- CUANDO TE DAN UN NUEVO SET DE DATOS Y TE PONES A PENSAR EN LA LIMPIEZA , VALIDACIÓN Y PREPROCESAMIENTO.





Tidy: Ordenar los datos

El objetivo es aprender comandos que nos den más herramientas para realizar el preprocesamiento de nuestra data, sobre todo pensando en que es común que nuestro punto de partida sea un excel que antes ha sido manipulado por otras personas.

Tidy: Ordenar los datos

Suponga que se quiere evaluar los casos de cierta enfermedad en un conjunto de países; nosotros normalmente deseamos tener los datos de esta manera:

```
table1
```

```
## # A tibble: 6 x 4
##   country      year  cases population
##   <chr>      <int>  <int>      <int>
## 1 Afghanistan 1999     745   19987071
## 2 Afghanistan 2000    2666   20595360
## 3 Brazil      1999   37737   172006362
## 4 Brazil      2000   80488   174504898
## 5 China       1999  212258  1272915272
## 6 China       2000  213766  1280428583
```

Tidy: Ordenar los datos

Sin embargo nos lo pueden dar así:

```
table2
```

```
## # A tibble: 12 x 4
##   country      year type      count
##   <chr>      <int> <chr>    <int>
## 1 Afghanistan 1999 cases      745
## 2 Afghanistan 1999 population 19987071
## 3 Afghanistan 2000 cases      2666
## 4 Afghanistan 2000 population 20595360
## 5 Brazil      1999 cases      37737
## 6 Brazil      1999 population 172006362
## 7 Brazil      2000 cases      80488
## 8 Brazil      2000 population 174504898
## 9 China       1999 cases      212258
## 10 China      1999 population 1272915272
## 11 China      2000 cases      213766
## 12 China      2000 population 1280428583
```

Tidy: Ordenar los datos

O así:

```
table3
```

```
## # A tibble: 6 x 3
##   country      year rate
## * <chr>      <int> <chr>
## 1 Afghanistan 1999 745/19987071
## 2 Afghanistan 2000 2666/20595360
## 3 Brazil      1999 37737/172006362
## 4 Brazil      2000 80488/174504898
## 5 China       1999 212258/1272915272
## 6 China       2000 213766/1280428583
```


Tidy: Ordenar los datos

O así:

```
table4a
```

```
## # A tibble: 3 x 3
##   country    `1999` `2000`
## * <chr>      <int>  <int>
## 1 Afghanistan    745    2666
## 2 Brazil       37737   80488
## 3 China        212258  213766
```

Tidy: Ordenar los datos

O así:

```
table4b
```

```
## # A tibble: 3 x 3
##   country      `1999`      `2000`
## * <chr>      <int>      <int>
## 1 Afghanistan 19987071    20595360
## 2 Brazil      172006362   174504898
## 3 China       1272915272  1280428583
```

Tidy: Ordenar los datos

En la mayoría de los casos, antes de realizar nuestros análisis debemos reordenar los datos en algún software como excel para cumplir lo siguiente

- Que cada columna sea una variable
- Que cada fila sea una observación (granularidad)
- Que cada celda sea el valor de la variable para la observación

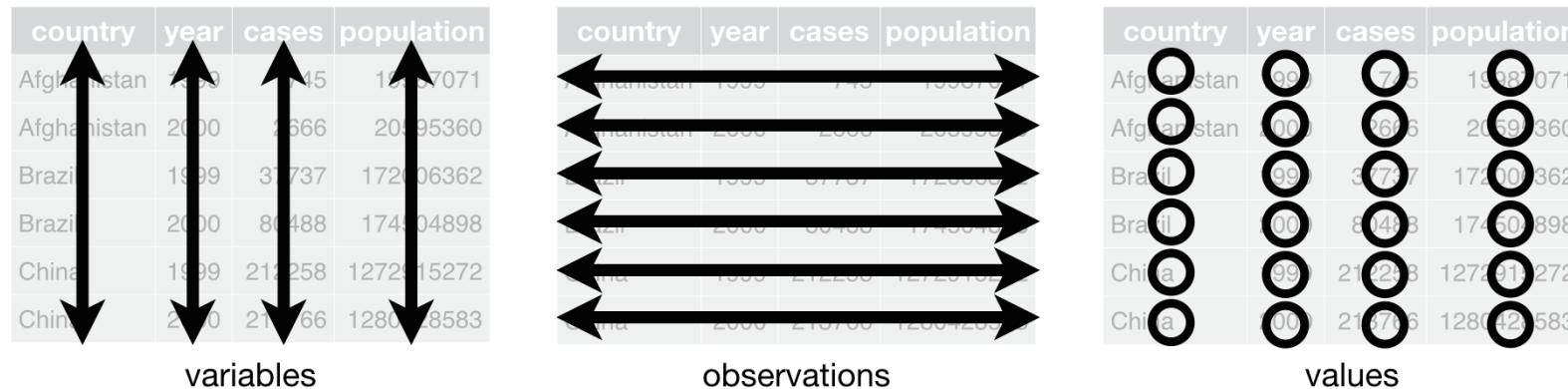


Gráfico tomado del libro R for Data Science, <https://r4ds.had.co.nz/>

Tidy: Ordenar los datos

En el conjunto `table1` se puede ver que la granularidad de la información es País, Año, mientras que las variables son: Casos y Población.

```
table1
```

```
## # A tibble: 6 x 4
##   country      year  cases population
##   <chr>      <int> <int>      <int>
## 1 Afghanistan 1999     745   19987071
## 2 Afghanistan 2000    2666   20595360
## 3 Brazil      1999   37737   172006362
## 4 Brazil      2000   80488   174504898
## 5 China       1999  212258  1272915272
## 6 China       2000  213766  1280428583
```

Tidy: Ordenar los datos

Datos ordenados nos permiten trabajar fácilmente con ellos.

```
# Calcular un ratio por cada 10 mil habitantes
table1 %>%
  mutate(rate = cases / population * 10000)
```

```
## # A tibble: 6 x 5
##   country      year  cases population  rate
##   <chr>      <int>  <int>      <int> <dbl>
## 1 Afghanistan 1999     745   19987071 0.373
## 2 Afghanistan 2000    2666   20595360 1.29
## 3 Brazil      1999   37737   172006362 2.19
## 4 Brazil      2000   80488   174504898 4.61
## 5 China       1999  212258  1272915272 1.67
## 6 China       2000  213766  1280428583 1.67
```

Tidy: Ordenar los datos

Datos ordenados nos permiten trabajar fácilmente con ellos.

```
# Calcular casos por año
table1 %>%
  count(year, wt = cases)
```

```
## # A tibble: 2 x 2
##   year      n
##   <int> <int>
## 1  1999 250740
## 2  2000 296920
```

Tidy: Ordenar los datos

`spread()` - Se lo usa cuando una observación está en diferentes filas

```
table2
```

```
## # A tibble: 12 x 4
##   country      year type      count
##   <chr>      <int> <chr>    <int>
## 1 Afghanistan 1999 cases      745
## 2 Afghanistan 1999 population 19987071
## 3 Afghanistan 2000 cases      2666
## 4 Afghanistan 2000 population 20595360
## 5 Brazil      1999 cases      37737
## 6 Brazil      1999 population 172006362
## 7 Brazil      2000 cases      80488
## 8 Brazil      2000 population 174504898
## 9 China       1999 cases     212258
## 10 China       1999 population 1272915272
## 11 China       2000 cases     213766
## 12 China       2000 population 1280428583
```

Tidy: Ordenar los datos

Comparando table2 con table1

- ¿Qué columna tiene los nombres de las columnas?
- ¿Qué columna contiene los valores observados?

```
table2
```

```
## # A tibble: 12 x 4
##   country      year type      count
##   <chr>      <int> <chr>    <int>
## 1 Afghanistan 1999 cases      745
## 2 Afghanistan 1999 population 19987071
## 3 Afghanistan 2000 cases      2666
## 4 Afghanistan 2000 population 20595360
## 5 Brazil      1999 cases      37737
## 6 Brazil      1999 population 172006362
## 7 Brazil      2000 cases      80488
## 8 Brazil      2000 population 174504898
## 9 China       1999 cases      212258
## 10 China      1999 population 1272915272
## 11 China      2000 cases      213766
## 12 China      2000 population 1280428583
```


Tidy: Ordenar los datos

Comparando table2 con table1

- ¿Qué columna tiene los nombres de las columnas? **type**
- ¿Qué columna contiene los valores observados? **count**

```
table2
```

```
## # A tibble: 12 x 4
##   country      year type      count
##   <chr>      <int> <chr>    <int>
## 1 Afghanistan 1999 cases      745
## 2 Afghanistan 1999 population 19987071
## 3 Afghanistan 2000 cases      2666
## 4 Afghanistan 2000 population 20595360
## 5 Brazil      1999 cases      37737
## 6 Brazil      1999 population 172006362
## 7 Brazil      2000 cases      80488
## 8 Brazil      2000 population 174504898
## 9 China       1999 cases      212258
## 10 China      1999 population 1272915272
## 11 China      2000 cases      213766
## 12 China      2000 population 1280428583
```

Tidy: Ordenar los datos

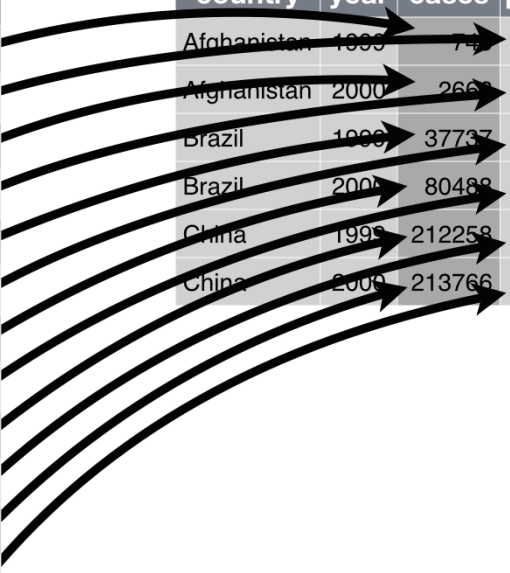
Para convertir `table2` en datos ordenados se usa `spread()`

```
# Aplicar spread a table2  
spread(table2, key = type, value = count)
```

```
## # A tibble: 6 x 4  
##   country      year  cases population  
##   <chr>      <int> <int>      <int>  
## 1 Afghanistan 1999     745   19987071  
## 2 Afghanistan 2000    2666   20595360  
## 3 Brazil      1999   37737   172006362  
## 4 Brazil      2000   80488   174504898  
## 5 China       1999  212258  1272915272  
## 6 China       2000  213766  1280428583
```

Tidy: Ordenar los datos

`spread()` transforma el 'key' en columnas y el 'value' en el valor de la observación



country	year	key	value
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

table2

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

Gráfico tomado del libro R for Data Science, <https://r4ds.had.co.nz/>

Tidy: Ordenar los datos

Para convertir `table2` en datos ordenados también se usa `pivot_wider()`

```
# Aplicar pivot_wider a table2
table2 %>% pivot_wider(
  names_from = type,
  values_from = count,
  values_fill = 0)
```

```
## # A tibble: 6 x 4
##   country      year  cases population
##   <chr>      <int>  <int>      <int>
## 1 Afghanistan 1999     745   19987071
## 2 Afghanistan 2000    2666   20595360
## 3 Brazil      1999   37737   172006362
## 4 Brazil      2000   80488   174504898
## 5 China       1999  212258  1272915272
## 6 China       2000  213766  1280428583
```

Tidy: Ordenar los datos

Además `pivot_wider()` permite combinar niveles de diferentes variables en las columnas y así también definir cómo se juntarán los nombres.

```
# Aplicar pivot_wider a table2
table2 %>% pivot_wider(
  names_from = c(type, year),
  names_glue = "{type}-{year}", # No es necesario
  values_from = count,
  values_fill = 0 )
```

```
## # A tibble: 3 x 5
##   country      `cases-1999` `population-1999` `cases-2000` `population-2000`
##   <chr>          <int>          <int>          <int>          <int>
## 1 Afghanistan      745      19987071         2666      20595360
## 2 Brazil          37737      172006362         80488      174504898
## 3 China           212258      1272915272        213766      1280428583
```

Tidy: Ordenar los datos

Además `pivot_wider()` permite combinar niveles de diferentes variables en las columnas y así también definir cómo se juntarán los nombres.

```
# Aplicar pivot_wider a table2
table2 %>% pivot_wider(
  names_from = c(country, type),
  values_from = c(count),
  values_fill = 0 )
```

```
## # A tibble: 2 x 7
##   year Afghanistan_cas~ Afghanistan_pop~ Brazil_cases Brazil_populati~ China_cases
##   <int>           <int>           <int>           <int>           <int>           <int>
## 1  1999             745       19987071         37737       172006362       212258
## 2  2000            2666       20595360         80488       174504898       213766
## # ... with 1 more variable: China_population <int>
```

Tidy: Ordenar los datos

`pivot_wider()` también permite hacer cálculos, vamos a obtener la suma de casos de ambos años, primero se selecciona las columnas (country, type, count) y miren lo que sucede al hacer el `pivot_wider()`

```
# Aplicar pivot_wider a table2
table2 %>%
  select(country, type, count) %>%
  pivot_wider(
    names_from = type,
    values_from = count)
```

```
## Warning: Values are not uniquely identified; output will contain list-cols.
## * Use `values_fn = list` to suppress this warning.
## * Use `values_fn = length` to identify where the duplicates arise
## * Use `values_fn = {summary_fun}` to summarise duplicates
```

```
## # A tibble: 3 x 3
##   country      cases      population
##   <chr>        <list>    <list>
## 1 Afghanistan <int [2]> <int [2]>
## 2 Brazil      <int [2]> <int [2]>
## 3 China       <int [2]> <int [2]>
```

Tidy: Ordenar los datos

`pivot_wider()` también permite hacer cálculos, vamos a obtener la suma de casos de ambos años, primero se selecciona las columnas (`country`, `type`, `count`) y ahora en `pivot_wider` se define una función a aplicar al vector de 2 que hay en cada celda.

```
# Aplicar pivot_wider a table2
table2 %>%
  select(country, type, count) %>%
  pivot_wider(
    names_from = type,
    values_from = count,
    values_fn = sum)
```

```
## # A tibble: 3 x 3
##   country      cases population
##   <chr>      <dbl>      <dbl>
## 1 Afghanistan    3411    40582431
## 2 Brazil         118225   346511260
## 3 China          426024  2553343855
```


Tidy: Ordenar los datos

Gather- Se lo usa cuando se tiene datos parecidos a tablas cruzadas

```
table4a
```

```
## # A tibble: 3 x 3
##   country    `1999` `2000`
## * <chr>      <int>  <int>
## 1 Afghanistan    745    2666
## 2 Brazil        37737   80488
## 3 China         212258  213766
```

Tidy: Ordenar los datos

¿Qué variables parece que se están cruzando en el table4a? ¿Qué valor ha tomado como dato para la tabla cruzada?

```
table4a
```

```
## # A tibble: 3 x 3
##   country    `1999` `2000`
## * <chr>      <int>  <int>
## 1 Afghanistan    745    2666
## 2 Brazil        37737   80488
## 3 China         212258  213766
```

Tidy: Ordenar los datos

¿Qué variables parece que se están cruzando en el table4a? **País y Año**

¿Qué valor ha tomado como dato para la tabla cruzada? **Casos**

```
table4a
```

```
## # A tibble: 3 x 3
##   country    `1999` `2000`
## * <chr>      <int>  <int>
## 1 Afghanistan    745    2666
## 2 Brazil        37737   80488
## 3 China         212258  213766
```

Tidy: Ordenar los datos

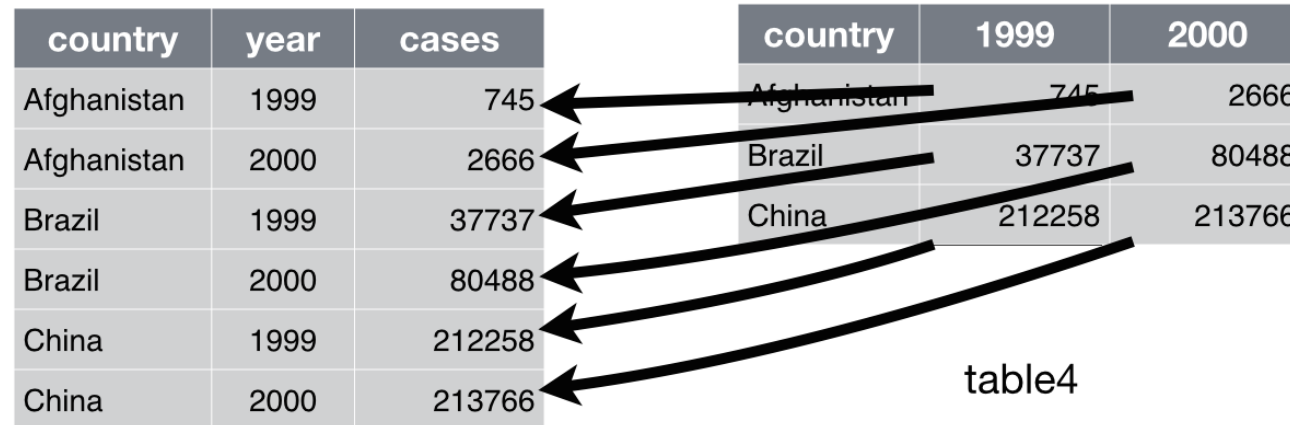
Para convertir `table4a` en datos ordenados se usa `gather()`

```
table4a %>%  
  gather(`1999`, `2000`, key = "year", value = "cases")
```

```
## # A tibble: 6 x 3  
##   country    year  cases  
##   <chr>      <chr> <int>  
## 1 Afghanistan 1999     745  
## 2 Brazil      1999   37737  
## 3 China       1999  212258  
## 4 Afghanistan 2000    2666  
## 5 Brazil      2000   80488  
## 6 China       2000  213766
```

Tidy: Ordenar los datos

`gather()` transforma los nombres de columnas dados como la nueva columna "key" y los valores de las celdas pasan a la columna "value".



country	year	cases
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766

country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

table4

Gráfico tomado del libro R for Data Science, <https://r4ds.had.co.nz/>

Tidy: Ordenar los datos

Para convertir `table4a` en datos ordenados también se puede usar `pivot_longer()`

```
table4a %>%  
  pivot_longer(-country, names_to = "year", values_to = "cases")
```

```
## # A tibble: 6 x 3  
##   country    year  cases  
##   <chr>      <chr> <int>  
## 1 Afghanistan 1999     745  
## 2 Afghanistan 2000    2666  
## 3 Brazil      1999   37737  
## 4 Brazil      2000   80488  
## 5 China       1999  212258  
## 6 China       2000  213766
```



Tidy: Ordenar los datos

Para convertir `table4a` en datos ordenados también se puede usar `pivot_longer()`

```
table4a %>%  
  pivot_longer(cols = `1999`:`2000`, names_to = "year", values_to = "cases")
```

```
## # A tibble: 6 x 3  
##   country    year  cases  
##   <chr>      <chr> <int>  
## 1 Afghanistan 1999     745  
## 2 Afghanistan 2000    2666  
## 3 Brazil      1999   37737  
## 4 Brazil      2000   80488  
## 5 China       1999  212258  
## 6 China       2000  213766
```

Tidy: Ordenar los datos

`Separate()` Permite disociar alguna columna que esté concatenada

```
table3
```

```
## # A tibble: 6 x 3
##   country      year rate
## * <chr>      <int> <chr>
## 1 Afghanistan 1999 745/19987071
## 2 Afghanistan 2000 2666/20595360
## 3 Brazil      1999 37737/172006362
## 4 Brazil      2000 80488/174504898
## 5 China       1999 212258/1272915272
## 6 China       2000 213766/1280428583
```


Tidy: Ordenar los datos

¿Qué columna parece estar concatenada?

```
table3
```

```
## # A tibble: 6 x 3
##   country      year rate
## * <chr>      <int> <chr>
## 1 Afghanistan 1999 745/19987071
## 2 Afghanistan 2000 2666/20595360
## 3 Brazil      1999 37737/172006362
## 4 Brazil      2000 80488/174504898
## 5 China       1999 212258/1272915272
## 6 China       2000 213766/1280428583
```

Tidy: Ordenar los datos

¿Qué columna parece estar concatenada? **rate**

```
table3
```

```
## # A tibble: 6 x 3
##   country      year rate
## * <chr>      <int> <chr>
## 1 Afghanistan 1999 745/19987071
## 2 Afghanistan 2000 2666/20595360
## 3 Brazil      1999 37737/172006362
## 4 Brazil      2000 80488/174504898
## 5 China       1999 212258/1272915272
## 6 China       2000 213766/1280428583
```

Tidy: Ordenar los datos

`separate()` Permite disociar alguna columna que esté concatenada

```
table3 %>%  
  separate(rate, into = c("cases", "population"), sep = "/")
```

```
## # A tibble: 6 x 4  
##   country      year cases population  
##   <chr>      <int> <chr>   <chr>  
## 1 Afghanistan 1999  745    19987071  
## 2 Afghanistan 2000 2666    20595360  
## 3 Brazil      1999 37737   172006362  
## 4 Brazil      2000 80488   174504898  
## 5 China       1999 212258  1272915272  
## 6 China       2000 213766  1280428583
```



Tidy: Ordenar los datos

Con `convert= TRUE` se transforma a número

```
table3 %>%  
  separate(rate, into = c("cases", "population"), sep = "/", convert = TRUE)
```

```
## # A tibble: 6 x 4  
##   country      year  cases population  
##   <chr>      <int> <int>      <int>  
## 1 Afghanistan 1999     745   19987071  
## 2 Afghanistan 2000    2666   20595360  
## 3 Brazil      1999   37737   172006362  
## 4 Brazil      2000   80488   174504898  
## 5 China       1999  212258  1272915272  
## 6 China       2000  213766  1280428583
```

Tidy: Ordenar los datos

Cuando nos pasan información procedentes de tablas dinámicas puede pasar que se nombra el primer valor y lo de abajo se asume es lo mismo

```
treatment <- tribble(  
  ~ person,      ~ treatment, ~response,  
  "Derrick Whitmore", 1,      7,  
  NA,              2,      10,  
  NA,              3,      9,  
  "Katherine Burke", 1,      4  
)
```

Tidy: Ordenar los datos

Con `fill()` se pueden llenar los NA

```
treatment
```

```
## # A tibble: 4 x 3
##   person      treatment response
##   <chr>      <dbl>     <dbl>
## 1 Derrick Whitmore      1         7
## 2 <NA>                2        10
## 3 <NA>                3         9
## 4 Katherine Burke      1         4
```

Tidy: Ordenar los datos

Con `fill()` se pueden llenar los NA

```
treatment %>%  
  fill(person)
```

```
## # A tibble: 4 x 3  
##   person      treatment response  
##   <chr>          <dbl>     <dbl>  
## 1 Derrick Whitmore      1         7  
## 2 Derrick Whitmore      2        10  
## 3 Derrick Whitmore      3         9  
## 4 Katherine Burke       1         4
```

Crear reportes específicos usando lo aprendido con TidyR

Curso: Manejo de datos y reportería con R

Néstor Montaña

Aplicando Tidyr - Construir reportes específicos

Obtener estadísticas del tiempo de servicio por Sucursal, pero mapeando la Sucursal a nivel de columna

```
# Primera parte, calculos
data_banco %>%
  group_by(Sucursal) %>%
  summarise(
    tibble(
      Estadistico = c("Min", "Q1", "Mediana",
                      "Q3", "Max", "Media", "Desv"),
      Valor= c(quantile(Tiempo_Servicio_seg,
                        c(0, 0.25, 0.5, 0.75, 1)),
              mean(Tiempo_Servicio_seg, na.rm= T),
              sd(Tiempo_Servicio_seg, na.rm= T)
            )
    )
  )
```

```
## `summarise()` regrouping output by 'Sucursal' (override with
## # A tibble: 35 x 3
## # Groups:   Sucursal [5]
##   Sucursal Estadistico Valor
##   <chr>      <chr>      <dbl>
## 1 Alborada Min          21.0
## 2 Alborada Q1           93.3
## 3 Alborada Mediana     148.
## 4 Alborada Q3          231.
## 5 Alborada Max        1082.
## 6 Alborada Media       183.
## 7 Alborada Desv       131.
## 8 Centro   Min          18.1
## 9 Centro   Q1           85.2
## 10 Centro  Mediana       136.
## # ... with 25 more rows
```

Aplicando Tidyr - Construir reportes específicos

Obtener estadísticas del tiempo de servicio por Sucursal, pero mapeando la Sucursal a nivel de columna

```
# Primera parte, calculos
data_banco %>%
  group_by(Sucursal) %>%
  summarise(
    tibble(
      Estadistico = c("Min", "Q1", "Mediana",
                      "Q3", "Max", "Media", "Desv"),
      Valor= c(quantile(Tiempo_Servicio_seg,
                        c(0, 0.25, 0.5, 0.75, 1)),
              mean(Tiempo_Servicio_seg, na.rm= T),
              sd(Tiempo_Servicio_seg, na.rm= T)
            )
    )
  ) %>%
  pivot_wider(
    names_from = Sucursal,
    values_from= Valor)
```

```
## `summarise()` regrouping output by 'Sucursal' (override with
```

```
## # A tibble: 7 x 6
##   Estadistico Alborada Centro `Mall del Sol` `Riocentro Sur`
##   <chr>         <dbl> <dbl>         <dbl>         <dbl>
## 1 Min           21.0   18.1           24.4           20
## 2 Q1            93.3   85.2           90.5           49
## 3 Mediana       148.    136.           144.           76
## 4 Q3            231.    208.           228.          111
## 5 Max          1082.   1603.          1213.          522
## 6 Media         183.    166.           181.           89.4
## 7 Desv          131.    121.           133.           57.9
```

Aplicando Tidyr - Construir reportes específicos

Obtener la media del tiempo de servicio y del monto pero mostrar las variables como filas

```
# Obtener la media del tiempo de servicio y del monto
# Pero mostrar las variables como filas
# Se debe tener cuidado con los nombres
data_banco %>%
  rename(TiempoServicioSeg= Tiempo_Servicio_seg) %>%
  summarise_at( vars(TiempoServicioSeg, Monto),
    list(
      Media= ~mean(., na.rm=TRUE),
      MediaAcot= ~mean(., na.rm = TRUE,
        trim = 0.05),
      Cantidad= ~n()
    )
  ) %>%
  gather %>%
  separate(key, c("Var", "Medida"), sep = "_") %>%
  spread(Medida, value)
```

```
## # A tibble: 2 x 4
##   Var          Cantidad Media MediaAcot
##   <chr>          <dbl> <dbl>    <dbl>
## 1 Monto          24299 1996.    1983.
## 2 TiempoServicioSeg 24299  156.    142.
```

FIN

Curso: Manejo de datos y reportería con R

Néstor Montaña