Tarea17

Yimmy Eman

2022-07-11

```
library(tidyverse)
```

Pregunta 1

Describe cómo están organizadas las variables y observaciones en cada una de las cinco familias de tablas table1,... table5 detidyverse.

```
# Cada fila es una combinación de country - year con las variables
# cases y population.

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
```

```
# Cada fila es una combinación de country, year, y una variable "type"
# con valores ("cases","population") , y otra variable count con el valor
# numérico de la anterior.
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
```

```
# Cada fila es una combinación de country - year con las variables rate en
# formato string.

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
```

```
# Tabla de los cases:
table4a
```

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

```
# Tabla de los population:
table4b
```

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

```
# Cada fila es una combinación de country - (con la separación de year en
# century y year) con las variables rate en formato string.
table5
```

```
## # A tibble: 6 x 4
##   country     century year  rate
## * <chr>       <chr>   <chr> <chr>
## 1 Afghanistan 19      99    745/19987071
## 2 Afghanistan 20      00    2666/20595360
## 3 Brazil      19      99    37737/172006362
## 4 Brazil      20      00    80488/174504898
## 5 China       19      99    212258/1272915272
## 6 China       20      00    213766/1280428583
```

Pregunta 2

Calcula la columna de rate para table2 y para la combinación de table4a y table4b sin usar las funciones gather o spread.

```
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
```

```
tb2_cases <-filter(table2, type=="cases")[["count"]]
tb2_country <-filter(table2, type=="cases")[["country"]]
tb2_year <-filter(table2, type=="cases")[["year"]]
tb2_population <-filter(table2, type=="population")[["count"]]

table2_clean <- tibble(country = tb2_country,
                       year = tb2_year,
                       rate = tb2_cases/tb2_population)
table2_clean
```

```
## # A tibble: 6 x 3
##   country      year      rate
##   <chr>       <int>     <dbl>
## 1 Afghanistan  1999 0.0000373
## 2 Afghanistan  2000 0.000129
## 3 Brazil       1999 0.000219
## 4 Brazil       2000 0.000461
## 5 China        1999 0.000167
## 6 China        2000 0.000167
```

```
tidy4a <- table4a %>%
  pivot_longer(c(`1999`, `2000`), names_to = "year", values_to = "cases")

tidy4b <- table4b %>%
  pivot_longer(c(`1999`, `2000`), names_to = "year", values_to = "population")

left_join(tidy4a, tidy4b) %>%
  mutate(rate = cases / population)
```

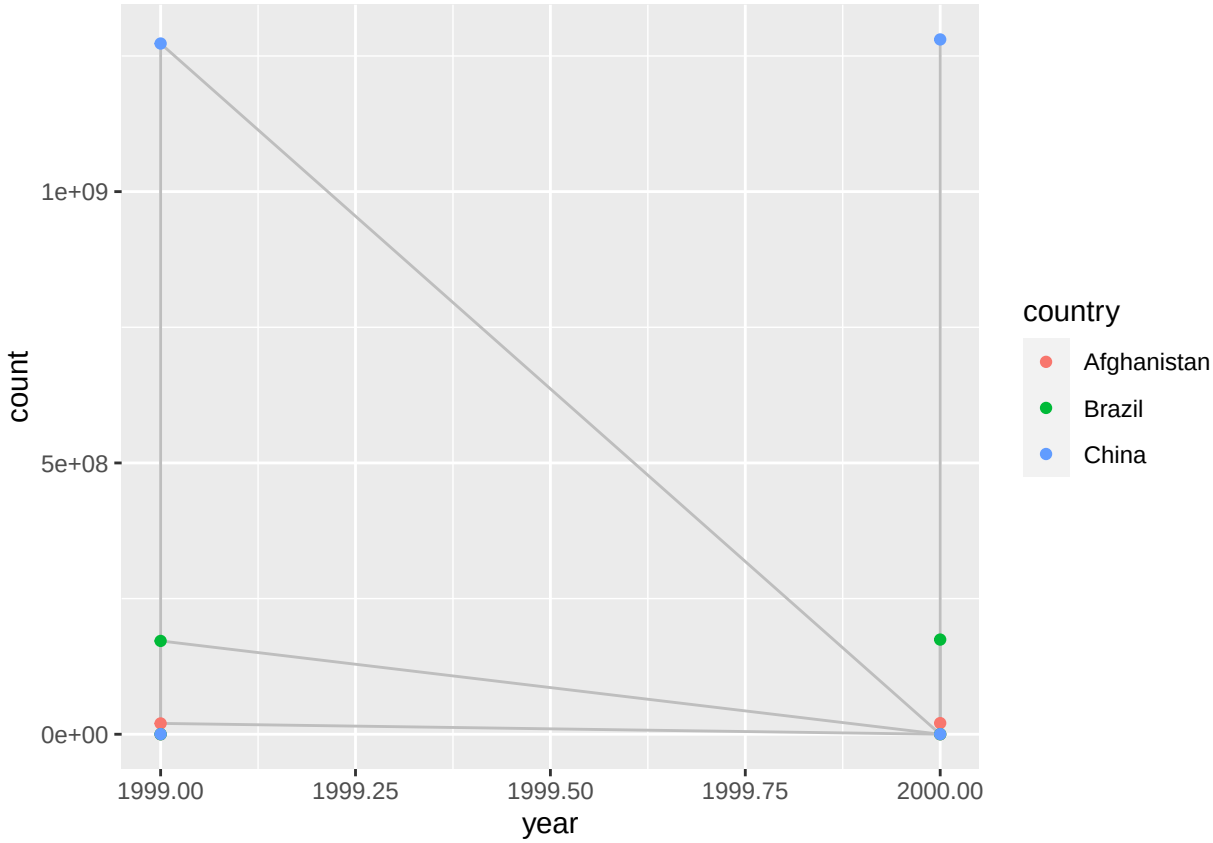
```
## Joining, by = c("country", "year")
```

```
## # A tibble: 6 x 5
##   country     year   cases population      rate
##   <chr>       <chr>  <int>      <int>     <dbl>
## 1 Afghanistan 1999     745   19987071 0.0000373
## 2 Afghanistan 2000    2666   20595360 0.000129
## 3 Brazil      1999   37737  172006362 0.000219
## 4 Brazil      2000   80488  174504898 0.000461
## 5 China       1999  212258 1272915272 0.000167
## 6 China       2000  213766 1280428583 0.000167
```

Pregunta 3

Recrea el plot que hemos hecho en la clase para mostrar los casos de infección a lo largo de los años usando la table2 en lugar de la table1. ¿En qué difiere el proceso?

```
table2 %>%
  ggplot(aes(year, count)) +
  geom_line(aes(group = country), color = "grey") +
  geom_point(aes(color = country))
```



Pregunta 4

Las funciones spread y gather no son absolutamente simétricas. Toma el siguiente ejemplo para explicarlo correctamente:

```
roi <-tibble(year =c(rep(2016,4),rep(2017,4), 2018),
             quarter =c(rep(c(1,2,3,4),2),1),
             return =rnorm(9, mean = 0.5, sd = 1))

roi %>%
  spread(year, return) %>%
  gather("year", "return", '2016':'2018')
```

```
## # A tibble: 12 x 3
##    quarter year  return
##      <dbl> <chr>  <dbl>
##  1       1 2016   0.322
##  2       2 2016   0.449
##  3       3 2016   1.40
##  4       4 2016  -0.221
##  5       1 2017   0.105
##  6       2 2017  -0.152
##  7       3 2017  -2.60
##  8       4 2017   0.164
##  9       1 2018  -0.395
## 10       2 2018  NA
## 11       3 2018  NA
## 12       4 2018  NA
```

The functions spread and gather are not perfectly symmetrical because column type information is not transferred between them. In the original table the column year was numeric, but after running spread() and gather() it is a character vector. This is because variable names are always converted to a character vector by gather().

Pregunta 5

Las funciones de spread y gather comparten un argumento convert. Investiga su uso.

Convert a data object to logical, integer, numeric, complex, character or factor as appropriate.

Pregunta 6

Sin ejecutar, investiga por qué falla el siguiente código table4a %>% gather(1999,2000, key = “year”, value = “cases”)

Porque faltan los backsticks en 1999 y 2000.

Pregunta 7

Explica por qué falla la función spread aplicada a la siguiente tribble:

```
people <- tribble(
  ~name,         ~key,   ~value,
  #-------------/-------/-------
```

```
  "Juan Gabriel", "age",      18,
  "Juan Gabriel", "weight",   58,
  "Juan Gabriel", "age",      30,
  "Juan Gabriel", "weight",   71,
  "Ricardo",      "age",      55,
  "Ricardo",      "age",      75
)
```

Hacer un Spreading del tibble falla porque hay dos filas con el campo age de Juan Gabriel. Se podríasolucionar añadiendo una nueva columna observación que contara si es la primera, la segunda. . . que semide la edad y/o peso del individuo.

Pregunta 8

Limpia la siguiente tibble con la función de spread o gather que creas más útil.

```
pregnancy <- tribble(
  ~pregnant, ~male, ~female,
  #--------/------/---------
  "yes",     NA,    32,
  "no",      85,    43
)
```

```
pregnancy %>%
  gather("male", "female", key = sex, value = count)  %>%
  mutate(pregnant = (pregnant == "yes"),
         female = (sex == "female")) %>%
  select(-sex)
```

```
## # A tibble: 4 x 3
##   pregnant count female
##   <lgl>    <dbl> <lgl>
## 1 TRUE        NA FALSE
## 2 FALSE       85 FALSE
## 3 TRUE        32 TRUE
## 4 FALSE       43 TRUE
```

```
pregnancy %>%
  pivot_longer(c("male", "female"),
               names_to = "sex", values_to = "count")
```

```
## # A tibble: 4 x 3
##   pregnant sex    count
##   <chr>    <chr>  <dbl>
## 1 yes      male      NA
## 2 yes      female    32
## 3 no       male      85
## 4 no       female    43
```

```
pregnancy %>%
  pivot_longer(c("male", "female"),
               names_to = "sex", values_to = "count") %>%
  mutate(pregnant = (pregnant == "yes"),
         female = (sex == "female")) %>%
  select(-sex) # Seleccionar todas - la variable sex
```

```
## # A tibble: 4 x 3
##   pregnant count female
##   <lgl>    <dbl> <lgl>
## 1 TRUE        NA FALSE
## 2 TRUE        32 TRUE
## 3 FALSE       85 FALSE
## 4 FALSE       43 TRUE
```

Pregunta 9

Investiga los parámetros extra y fill de la función separate. Experimenta con varias opciones de las mismas con las dos tibbles siguientes:

```
tibble(x =c("a,b,c", "d,e,f,g","h,i,j")) %>%
  separate(x,c("x", "y", "z"))
```

```
## Warning: Expected 3 pieces. Additional pieces discarded in 1 rows [2].
```

```
## # A tibble: 3 x 3
##   x     y     z
##   <chr> <chr> <chr>
## 1 a     b     c
## 2 d     e     f
## 3 h     i     j
```

```
tibble(x =c("a,b,c", "d,e","f,g,h")) %>%
  separate(x,c("x", "y", "z"))
```

```
## Warning: Expected 3 pieces. Missing pieces filled with `NA` in 1 rows [2].
```

```
## # A tibble: 3 x 3
##   x     y     z
##   <chr> <chr> <chr>
## 1 a     b     c
## 2 d     e     <NA>
## 3 f     g     h
```

Sol.

```
tibble(x =c("a,b,c", "d,e,f,g","h,i,j")) %>%
  separate(x,c("x", "y", "z"), extra = "drop")
```

```
## # A tibble: 3 x 3
##   x     y     z
##   <chr> <chr> <chr>
## 1 a     b     c
## 2 d     e     f
## 3 h     i     j
```

```
tibble(x =c("a,b,c", "d,e","f,g,h")) %>%
  separate(x,c("x", "y", "z"), fill = "left")
```

```
## # A tibble: 3 x 3
##   x     y     z
##   <chr> <chr> <chr>
## 1 a     b     c
## 2 <NA>  d     e
## 3 f     g     h
```

Pregunta 10

unite y separate tienen un argumento llamado remove. ¿Cómo funciona? ¿Se te ocurre cuando lo pondrías a false?

IfTRUE, remove input column from output data frame.

Pregunta 11

Compara las funciones separate y unite. ¿Por qué existen tres variantes de separación (basándonos en posición, separador o por grupos) pero solamente una para unir?

La función de unir simplemente concatena con el separador especificado, el restode parámetros no tienen sentido

Pregunta 12

Compara el argumento fill de spread y el de complete. Investiga también el argumento direction de la función fill.

Fill completa los datos que faltan con NAs donde pertoque. La dirección es para completarlos desde laderecha o desde la izquierda.

```
# Ejemplos

tibble(x =c("a,b,c", "d,e","f,g,h")) %>%
  separate(x,c("x", "y", "z"), fill = "left")
```

```
## # A tibble: 3 x 3
##   x     y     z
##   <chr> <chr> <chr>
## 1 a     b     c
## 2 <NA>  d     e
## 3 f     g     h
```

```r
tibble(x =c("a,b,c", "d,e","f,g,h")) %>%
  separate(x,c("x", "y", "z"), fill = "right")
```

```
## # A tibble: 3 x 3
##   x     y     z
##   <chr> <chr> <chr>
## 1 a     b     c
## 2 d     e     <NA>
## 3 f     g     h
```