

tidyr

Entornos de Análisis de Datos: R

Alberto Torres Barrán

2020-02-06

Introducción

- El 80% del tiempo de un análisis se emplea limpiando y preparando los datos (Dasu y Johnson, 2003)
- Importados los datos, es importante estructurarlos para que el análisis sea lo más fácil posible
- Las librerías del tidyverse están construidas alrededor del concepto de datos ordenados o *tidy data*:
 - Cada variable forma una columna
 - Cada observación forma una fila
 - Cada tipo de observación forma una tabla
- Datos tabulares/rectangulares no implican datos ordenados!!

Formas de almacenamiento

Distintas formas de almacenar los mismos datos [R for Data Science]:

```
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
```

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

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

table4a

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

table4b

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

Formatos "ancho" y "largo"

wide

id	x	y	z
1	a	c	e
2	b	d	f

long

id	key	val
1	x	a
2	x	b
1	y	c
2	y	d
1	z	e
2	z	f

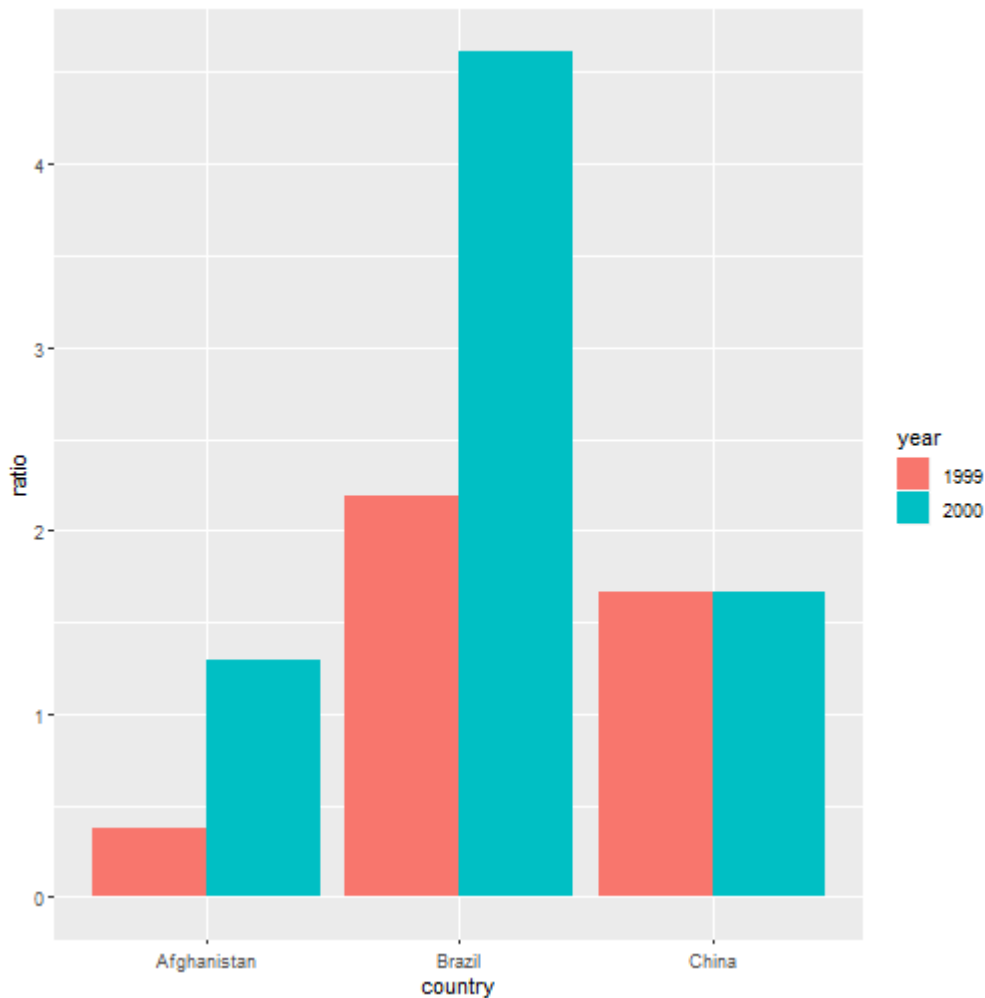
Operaciones con datos ordenados

```
table1 %>%  
  mutate(ratio = cases / population * 10000)  
## # A tibble: 6 x 5  
##   country      year  cases population ratio  
##   <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
```



```
table1 %>%  
  group_by(year) %>%  
  summarize(total = sum(cases))  
## # A tibble: 2 x 2  
##   year total  
##   <int> <int>  
## 1  1999 250740  
## 2  2000 296920
```

```
table1 %>%  
  mutate(ratio = cases / population * 10000,  
         year = as.character(year)) %>%  
  ggplot(mapping = aes(x = country, y = ratio, fill = year)) +  
    geom_bar(stat = "identity", position = "dodge")
```



tidyr 1.0

- Desde la versión 1.0 de tidyr, se han creado versiones más potentes de las antiguas `spread` y `gather`
- En la siguiente tabla podemos ver la equivalencia:

pandas	tidyr <1.0	tidyr 1.0	data.table	reshape2
pivot	spread	pivot_wider	dcast	cast
melt	gather	pivot_longer	melt	melt

pivot_wider

```
head(table2)
## # A tibble: 6 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
```

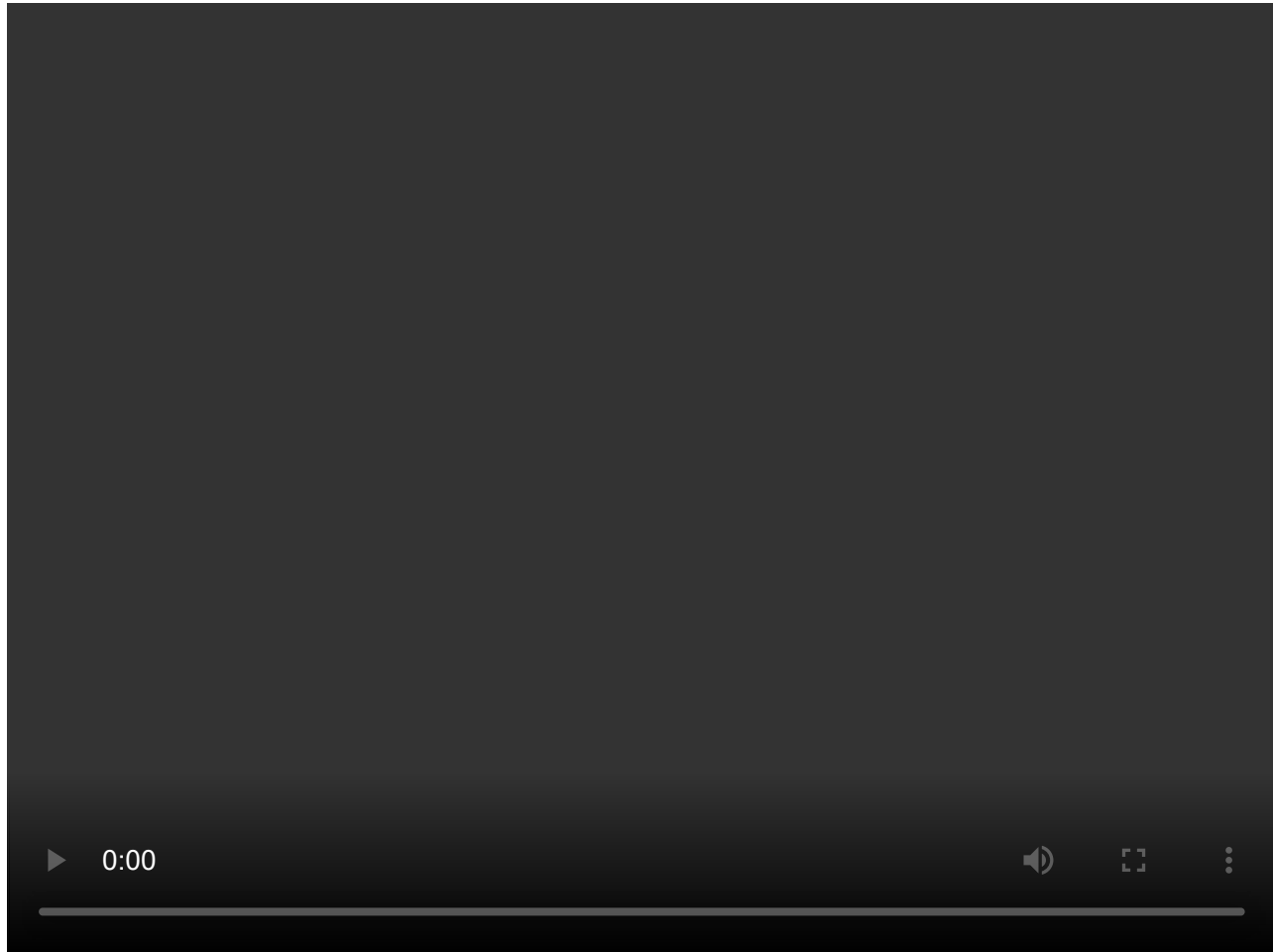
```
pivot_wider(table2, names_from = type, values_from = count)
## # A tibble: 6 x 4
##   country      year cases population
##   <chr>      <int> <int>    <int>
## 1 Afghanistan 1999     745  19987071
## 2 Afghanistan 2000    2666  20595360
## 3 Brazil      1999   37737  172006362
## 4 Brazil      2000   80488  174504898
## 5 China       1999  212258 1272915272
## 6 China       2000  213766 1280428583
```

pivot_longer

```
table4a
## # A tibble: 3 x 3
##   country    `1999`    `2000`
##   <chr>      <int>    <int>
## 1 Afghanistan    745    2666
## 2 Brazil      37737   80488
## 3 China      212258  213766
```

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

Ejemplo



separate

```
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
```

```
separate(table3, 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
```

- Por defecto `separate()` mantiene el tipo de la columna en las nuevas

```
separate(table3, 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
```


unite

```
unite(mpg, make, manufacturer, model, sep = " ")
```

```
## # A tibble: 234 x 10
```

##	make	displ	year	cyl	trans	drv	cty	hwy	fl	class
##	<chr>	<dbl>	<int>	<int>	<chr>	<chr>	<int>	<int>	<chr>	<chr>
##	1 audi a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
##	2 audi a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
##	3 audi a4	2	2008	4	manual(m6)	f	20	31	p	compact
##	4 audi a4	2	2008	4	auto(av)	f	21	30	p	compact
##	5 audi a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
##	6 audi a4	2.8	1999	6	manual(m5)	f	18	26	p	compact
##	7 audi a4	3.1	2008	6	auto(av)	f	18	27	p	compact
##	8 audi a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact
##	9 audi a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact
##	10 audi a4 quattro	2	2008	4	manual(m6)	4	20	28	p	compact
##	# ... with 224 more rows									

Otras funciones

tidyr también tiene otras funciones útiles para trabajar con NA s:

- `drop_na()` , elimina filas que tengan algún NA
- `fill()` , completa NA s con el valor anterior
- `replace_na()` , reemplaza NA s por un valor