

tidyr

Entornos de Análisis de Datos: R

Alberto Torres Barrán

2021-02-01

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*:
 1. Cada variable forma una columna
 2. Cada observación forma una fila
 3. Cada celda contiene un único valor
- Todas las librerías del tidyverse están construidas alrededor de este concepto
- Datos tabulares/rectangulares no implican datos ordenados!!!

Formas de almacenamiento

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

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

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

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

- Tabla 1

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

- Tabla 2

country	cases_1999	cases_2000	population_1999	population_2000
Afghanistan	745	2666	19987071	20595360
Brazil	37737	80488	172006362	174504898
China	212258	213766	1272915272	1280428583

Crear una nueva variable

- Crear una nueva variable es fácil con la tabla 1

```
table_1 %>%  
  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
```

- ¿Como podríamos hacer lo mismo con la tabla 2?

Operaciones agrupadas

- Tabla 1: calcular el total de casos por año

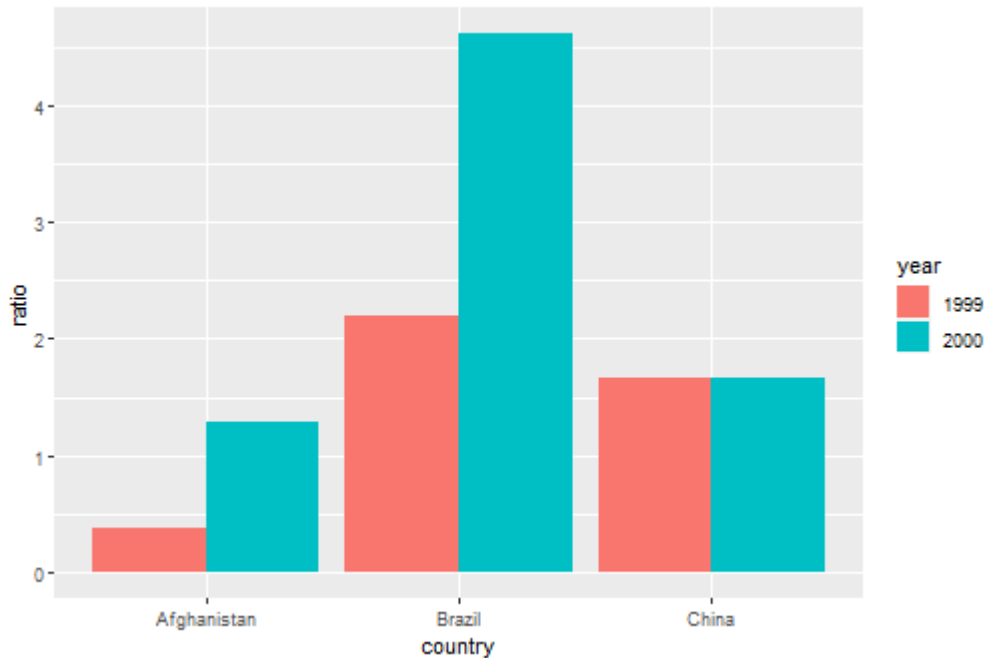
```
table_1 %>%  
  group_by(year) %>%  
  summarize(total = sum(cases))  
## # A tibble: 2 x 2  
##   year  total  
##   <int> <int>  
## 1  1999 250740  
## 2  2000 296920
```

- Tabla 2: el resultado final está en un formato que es más difícil de procesar

```
table_2 %>%  
  summarize(across(starts_with("cases"), sum))  
## # A tibble: 1 x 2  
##   cases_1999 cases_2000  
##   <int>      <int>  
## 1    250740    296920
```

Gráficos

```
table_1 %>%  
  mutate(ratio = cases / population * 10000,  
         year = as.character(year)) %>%  
  ggplot(mapping = aes(x = country, y = ratio, fill = year)) +  
    geom_col(position = "dodge")
```



Librería tidy

- Implementa funciones para transformar entre los formatos anteriores
- Desde la versión 1.0 de tidy, 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_longer

- Pivota una dataframe para que cada variable tenga su propia columna (**documentación**)

```
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
```

Animación

resp_id	age	airplane	anchorman	bridesmaids
1	48	TRUE	TRUE	TRUE
2	31	FALSE	TRUE	TRUE
3	30	FALSE	FALSE	FALSE



Ejemplo pivot_longer

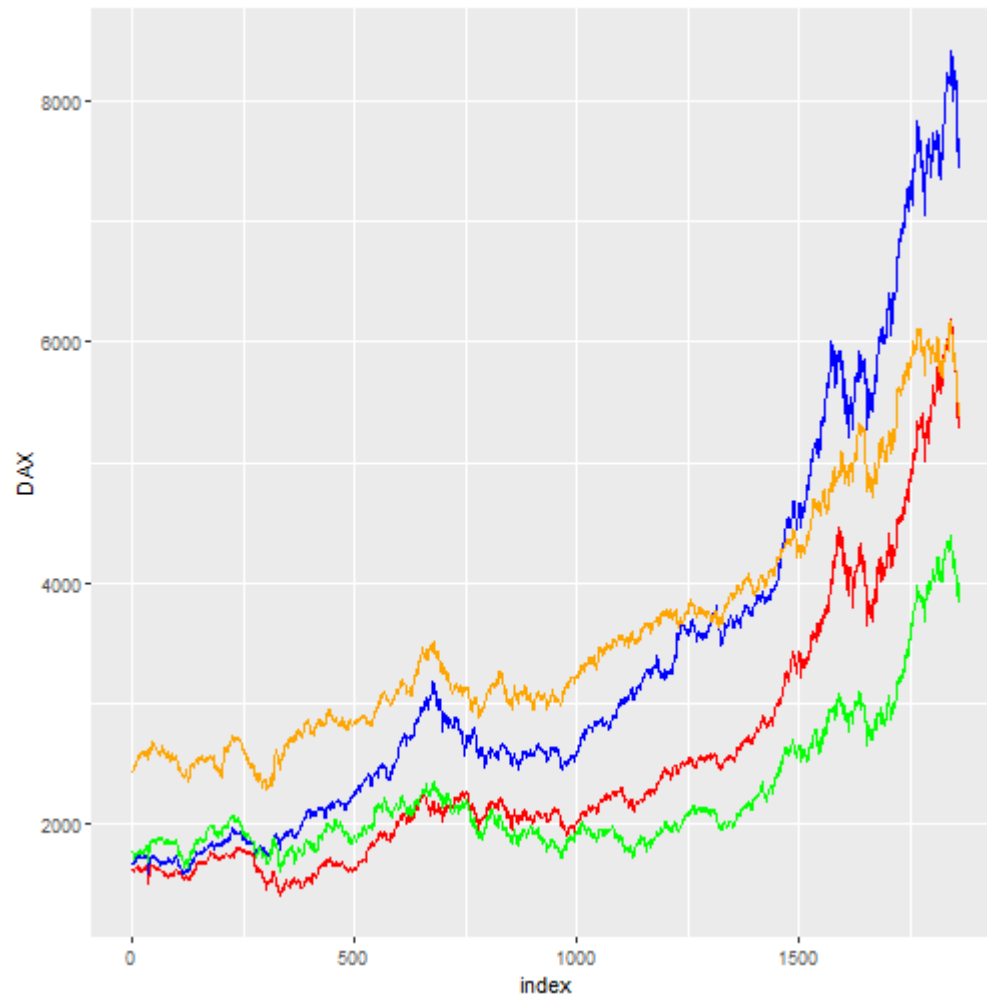
- Queremos representar las siguientes series temporales, que representan 4 indicadores económicos:

```
eu <- EuStockMarkets %>%  
  data.frame() %>%  
  mutate(index = 1:n())  
head(eu)
```

##		DAX	SMI	CAC	FTSE	index
##	1	1628.75	1678.1	1772.8	2443.6	1
##	2	1613.63	1688.5	1750.5	2460.2	2
##	3	1606.51	1678.6	1718.0	2448.2	3
##	4	1621.04	1684.1	1708.1	2470.4	4
##	5	1618.16	1686.6	1723.1	2484.7	5
##	6	1610.61	1671.6	1714.3	2466.8	6

- Las 4 variables tienen las mismas unidades, por lo que podemos representar 4 líneas en un único gráfico:

```
ggplot(eu, aes(x = index)) +  
  geom_line(aes(y = DAX), color = "red") +  
  geom_line(aes(y = SMI), color = "blue") +  
  geom_line(aes(y = CAC), color = "green") +  
  geom_line(aes(y = FTSE), color = "orange")
```



- Pivotamos a formato "largo"

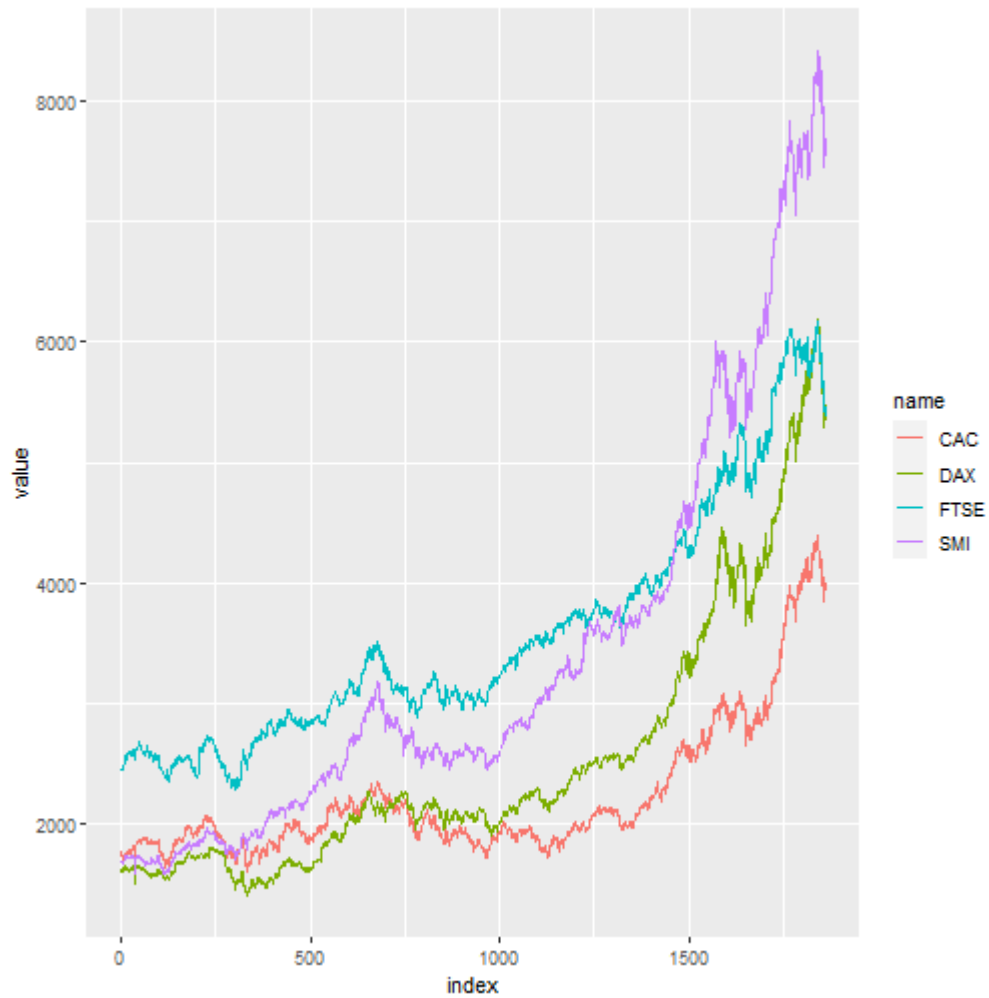
```
eu_long <-  
  eu %>%  
    pivot_longer(-index, names_to = "name", values_to = "value")  
  
head(eu_long, 8)  
## # A tibble: 8 x 3  
##   index name    value  
##   <int> <chr>  <dbl>  
## 1     1 DAX    1629.  
## 2     1 SMI    1678.  
## 3     1 CAC    1773.  
## 4     1 FTSE   2444.  
## 5     2 DAX    1614.  
## 6     2 SMI    1688.  
## 7     2 CAC    1750.  
## 8     2 FTSE   2460.
```

- Repetimos el gráfico anterior, asignando el nombre del indicador a la propiedad `color`

```
ggplot(eu_long, aes(x = index, y = value, color = name)) +  
  geom_line()
```

- Ventajas:

1. código más conciso
2. colores elegidos de forma automática a partir de una paleta
3. leyenda automática



Ejemplo avanzado

```
stats <-  
  mpg %>%  
    summarize(across(is.numeric, list(mean = mean, sd = sd)))
```

```
stats  
## # A tibble: 1 x 10  
##   displ_mean displ_sd year_mean year_sd cyl_mean cyl_sd cty_mean cty_sd hwy_mean  
##   <dbl>      <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>  
## 1      3.47      1.29    2004.      4.51      5.89      1.61     16.9      4.26     23.  
## # ... with 1 more variable: hwy_sd <dbl>
```

```
pivot_longer(stats, everything(), names_sep = "_", names_to = c("var", ".value"))  
## # A tibble: 5 x 3  
##   var      mean      sd  
##   <chr>    <dbl> <dbl>  
## 1 displ      3.47  1.29  
## 2 year    2004.   4.51  
## 3 cyl       5.89  1.61  
## 4 cty      16.9   4.26  
## 5 hwy      23.4   5.95
```


pivot_wider

- Realiza la operación inversa a `pivot_longer` ([documentación](#))

```
library(tidyr)
table2
## # A tibble: 12 x 4
##   country    year type
##   <chr>    <int> <chr>
## 1 Afghanistan 1999 cases
## 2 Afghanistan 1999 population 1
## 3 Afghanistan 2000 cases
## 4 Afghanistan 2000 population 2
## 5 Brazil      1999 cases
## 6 Brazil      1999 population 17
## 7 Brazil      2000 cases
## 8 Brazil      2000 population 17
## 9 China       1999 cases
## 10 China      1999 population 127
## 11 China      2000 cases
## 12 China      2000 population 128
```

```
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    199870
## 2 Afghanistan 2000    2666    205953
## 3 Brazil      1999   37737   1720063
## 4 Brazil      2000   80488   1745048
## 5 China       1999  212258  12729152
## 6 China       2000  213766  12804285
```

Ejemplos pivot_wider

- Dependiendo del uso, a veces es útil transformar nuestros datos en un formato no ordenado
- Es raro tener que usar `pivot_wider` para transformar una dataframe en datos ordenados (cada columna representa una variable)
- Se usa bastante para dos tareas:
 1. crear tablas resumen
 2. transformar datos para modelización
- Más ejemplos: *vignette pivoting*

Ejemplo tabla resumen

```
cty <-  
  mpg %>%  
    group_by(year, class) %>%  
    summarize(avg_cty = mean(cty))
```

```
head(cty)  
## # A tibble: 6 x 3  
## # Groups:   year [1]  
##   year class      avg_cty  
##   <int> <chr>      <dbl>  
## 1  1999 2seater      15.5  
## 2  1999 compact      19.8  
## 3  1999 midsize      18.2  
## 4  1999 minivan      16.2  
## 5  1999 pickup       13  
## 6  1999 subcompact    21.6
```

```
pivot_wider(cty, names_from = class, values_from = avg_cty)  
## # A tibble: 2 x 8  
## # Groups:   year [2]  
##   year 2seater compact midsize minivan pickup subcompact suv  
##   <int>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl> <dbl>  
## 1  1999    15.5    19.8    18.2    16.2     13    21.6  13.4  
## 2  2008    15.3    20.5    19.3    15.4     13    18.9  13.6
```

Ejemplo tabla resumen múltiple

```
cty <-  
  mpg %>%  
    group_by(year, class) %>%  
    summarize(avg_cty = mean(c  
      sd_cty = sd(cty))  
  
head(cty)  
## # A tibble: 6 x 4  
## # Groups:   year [1]  
##   year class      avg_cty  
##   <int> <chr>      <dbl>  
## 1  1999 2seater      15.5  
## 2  1999 compact      19.8  
## 3  1999 midsize      18.2  
## 4  1999 minivan      16.2  
## 5  1999 pickup       13  
## 6  1999 subcompact    21.6
```

```
pivot_wider(cty,  
            names_from = year,  
            values_from = c(avg_cty, sd_cty))  
## # A tibble: 7 x 5  
##   class      avg_cty_1999 avg_cty_2008 sd_c  
##   <chr>      <dbl>      <dbl>  
## 1 2seater      15.5      15.3  
## 2 compact      19.8      20.5  
## 3 midsize      18.2      19.3  
## 4 minivan      16.2      15.4  
## 5 pickup       13       13  
## 6 subcompact    21.6      18.9  
## 7 suv          13.4      13.6
```

Ejemplo modelización

- En este ejemplo tenemos 6 variables: país, continente, año, esperanza de vida, población, PIB per cápita

```
gapminder
## # A tibble: 1,704 x 6
##   country      continent year lifeExp      pop gdpPercap
##   <fct>        <fct>    <int>   <dbl>    <int>    <dbl>
## 1 Afghanistan Asia      1952    28.8  8425333    779.
## 2 Afghanistan Asia      1957    30.3  9240934    821.
## 3 Afghanistan Asia      1962    32.0 10267083    853.
## 4 Afghanistan Asia      1967    34.0 11537966    836.
## 5 Afghanistan Asia      1972    36.1 13079460    740.
## 6 Afghanistan Asia      1977    38.4 14880372    786.
## 7 Afghanistan Asia      1982    39.9 12881816    978.
## 8 Afghanistan Asia      1987    40.8 13867957    852.
## 9 Afghanistan Asia      1992    41.7 16317921    649.
## 10 Afghanistan Asia      1997    41.8 22227415    635.
## # ... with 1,694 more rows
```

- Diríamos que estos datos están ordenados, ya que cada variable se corresponde con una columna
- Este format es útil, entre otras cosas, para realizar gráficos

- Si quisiéramos, por ejemplo, realizar un modelo para predecir la esperanza de vida de China basándonos en el resto de países de su continente, necesitamos una columna para cada país

```
gapminder %>%
  filter(continent == "Asia") %>%
  pivot_wider(id_cols = year, names_from = country, values_from = lifeExp)
## # A tibble: 12 x 34
##   year Afghanistan Bahrain Bangladesh Cambodia China `Hong Kong, Chi~ India
##   <int>          <dbl>    <dbl>         <dbl>      <dbl> <dbl>          <dbl> <dbl>
## 1  1952          28.8      50.9          37.5        39.4  44            61.0  37.4
## 2  1957          30.3      53.8          39.3        41.4  50.5          64.8  40.2
## 3  1962          32.0      56.9          41.2        43.4  44.5          67.6  43.6
## 4  1967          34.0      59.9          43.5        45.4  58.4          70    47.2
## 5  1972          36.1      63.3          45.3        40.3  63.1          72    50.7
## 6  1977          38.4      65.6          46.9        31.2  64.0          73.6  54.2
## 7  1982          39.9      69.1          50.0        51.0  65.5          75.4  56.6
## 8  1987          40.8      70.8          52.8        53.9  67.3          76.2  58.6
## 9  1992          41.7      72.6          56.0        55.8  68.7          77.6  60.2
## 10 1997          41.8      73.9          59.4        56.5  70.4          80    61.8
## 11 2002          42.1      74.8          62.0        56.8  72.0          81.5  62.9
## 12 2007          43.8      75.6          64.1        59.7  73.0          82.2  64.7
## # ... with 26 more variables: Indonesia <dbl>, Iran <dbl>, Iraq <dbl>, Israel <dbl>,
## # Japan <dbl>, Jordan <dbl>, `Korea, Dem. Rep.` <dbl>, `Korea, Rep.` <dbl>,
## # Kuwait <dbl>, Lebanon <dbl>, Malaysia <dbl>, Mongolia <dbl>, Myanmar <dbl>,
## # Nepal <dbl>, Oman <dbl>, Pakistan <dbl>, Philippines <dbl>, `Saudi
## # Arabia` <dbl>, Singapore <dbl>, `Sri Lanka` <dbl>, Syria <dbl>, Taiwan <dbl>,
## # Thailand <dbl>, Vietnam <dbl>, `West Bank and Gaza` <dbl>, `Yemen, Rep.` <dbl>
```

separate

- Múltiples variables codificadas en una única columna

```
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
```


Ejemplo

```
separate(economics, date, into = c("year", "month", "day"), sep = "-", convert = T)
## # A tibble: 574 x 8
##   year month   day   pce    pop psavert uempmed unemploy
##   <int> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  1967     7     1  507. 198712  12.6   4.5   2944
## 2  1967     8     1  510. 198911  12.6   4.7   2945
## 3  1967     9     1  516. 199113  11.9   4.6   2958
## 4  1967    10     1  512. 199311  12.9   4.9   3143
## 5  1967    11     1  517. 199498  12.8   4.7   3066
## 6  1967    12     1  525. 199657  11.8   4.8   3018
## 7  1968     1     1  531. 199808  11.7   5.1   2878
## 8  1968     2     1  534. 199920  12.3   4.5   3001
## 9  1968     3     1  544. 200056  11.7   4.1   2877
## 10 1968     4     1  544. 200208  12.3   4.6   2709
## # ... with 564 more rows
```

unite

- Una única variable que está codificada en varias columnas

```
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
```

Ejemplo

```
storms %>%
  unite(date, year, month, day, sep = "-", remove = FALSE, na.rm = TRUE) %>%
  select(date, year, month, day)
## # A tibble: 10,010 x 4
##   date      year month  day
##   <chr>    <dbl> <dbl> <int>
## 1 1975-6-27  1975     6     27
## 2 1975-6-27  1975     6     27
## 3 1975-6-27  1975     6     27
## 4 1975-6-27  1975     6     27
## 5 1975-6-28  1975     6     28
## 6 1975-6-28  1975     6     28
## 7 1975-6-28  1975     6     28
## 8 1975-6-28  1975     6     28
## 9 1975-6-29  1975     6     29
## 10 1975-6-29  1975     6     29
## # ... with 10,000 more rows
```

Missing values en R

- `NA` es una constante que representa valores que faltan (*missing values*)
- Puede estar contenida dentro de vectores (columnas) de cualquier tipo
- `is.na()` devuelve `TRUE` si el valor es `NA` y `FALSE` en caso contrario
- Muchas funciones de R tienen un parámetro opcional `na.rm` que ignora `NA`s

```

airquality %>%
  mutate(Ozone_NA = is.na(Ozone)) %>%
  select(Ozone, Ozone_NA) %>%
  slice(1:15)

```

```

##      Ozone Ozone_NA
## 1      41    FALSE
## 2      36    FALSE
## 3      12    FALSE
## 4      18    FALSE
## 5      NA     TRUE
## 6      28    FALSE
## 7      23    FALSE
## 8      19    FALSE
## 9       8    FALSE
## 10     NA     TRUE
## 11       7    FALSE
## 12      16    FALSE
## 13      11    FALSE
## 14      14    FALSE
## 15      18    FALSE

```

```

dia <-
  diamonds %>%
    mutate(y_new = ifelse(!between(y, 3, 20), NA, y))

```

```

dia %>%
  summarize(y_na = sum(is.na(y_new)))
## # A tibble: 1 x 1
##   y_na
##   <int>
## 1     9

```

```

dia %>%
  filter(is.na(y_new)) %>%
  select(y, y_new)
## # A tibble: 9 x 2
##       y y_new
##   <dbl> <dbl>
## 1     0    NA
## 2     0    NA
## 3  58.9    NA
## 4     0    NA
## 5     0    NA
## 6     0    NA
## 7  31.8    NA
## 8     0    NA
## 9     0    NA

```

```

dia %>%
  summarize(avg = mean(y_new))
## # A tibble: 1 x 1
##       avg
##   <dbl>
## 1    NA

```

```

dia %>%
  summarize(avg = mean(y_new,
                        na.rm = TRUE))
## # A tibble: 1 x 1
##       avg
##   <dbl>
## 1  5.73

```

Funciones para gestionar NA

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

drop_na

```
str(airquality)
```

```
## 'data.frame': 153 obs. of 6 variables:
## $ Ozone : int 41 36 12 18 NA 28 23 19 8 NA ...
## $ Solar.R: int 190 118 149 313 NA NA 299 99 19 194 ...
## $ Wind : num 7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
## $ Temp : int 67 72 74 62 56 66 65 59 61 69 ...
## $ Month : int 5 5 5 5 5 5 5 5 5 5 ...
## $ Day : int 1 2 3 4 5 6 7 8 9 10 ...
```

```
summary(airquality)
```

```
##      Ozone      Solar.R      wind      Temp      Month
## Min.   : 1.00   Min.   : 7.0   Min.   : 1.700   Min.   :56.00   Min.   :5.00
## 1st Qu.:18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:72.00   1st Qu.:6.00
## Median :31.50   Median :205.0   Median : 9.700   Median :79.00   Median :7.00
## Mean   :42.13   Mean   :185.9   Mean   : 9.958   Mean   :77.88   Mean   :6.99
## 3rd Qu.:63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00   3rd Qu.:8.00
## Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00   Max.   :9.00
## NA's   :37      NA's   :7
##      Day
## Min.   : 1.0
## 1st Qu.: 8.0
## Median :16.0
## Mean   :15.8
## 3rd Qu.:23.0
## Max.   :31.0
##
```



```

airquality %>%
  drop_na() %>%
  str()
## 'data.frame': 111 obs. of 6 variables:
## $ Ozone : int 41 36 12 18 23 19 8 16 11 14 ...
## $ Solar.R: int 190 118 149 313 299 99 19 256 290 274 ...
## $ Wind : num 7.4 8 12.6 11.5 8.6 13.8 20.1 9.7 9.2 10.9 ...
## $ Temp : int 67 72 74 62 65 59 61 69 66 68 ...
## $ Month : int 5 5 5 5 5 5 5 5 5 5 ...
## $ Day : int 1 2 3 4 7 8 9 12 13 14 ...

```

```

filter(airquality, is.na(Ozone), is.na(Solar.R))
##   Ozone Solar.R Wind Temp Month Day
## 1   NA      NA 14.3   56     5    5
## 2   NA      NA  8.0   57     5   27

```

```

airquality %>%
  drop_na(Ozone) %>%
  str()
## 'data.frame': 116 obs. of 6 variables:
## $ Ozone : int 41 36 12 18 28 23 19 8 7 16 ...
## $ Solar.R: int 190 118 149 313 NA 299 99 19 NA 256 ...
## $ Wind : num 7.4 8 12.6 11.5 14.9 8.6 13.8 20.1 6.9 9.7 ...
## $ Temp : int 67 72 74 62 66 65 59 61 74 69 ...
## $ Month : int 5 5 5 5 5 5 5 5 5 5 ...
## $ Day : int 1 2 3 4 6 7 8 9 11 12 ...

```

fill

```
airquality %>%  
  # no fill  
  slice(c(1:10, 50:59)) %>%  
  select(-c(Temp, wind))
```

##	Ozone	Solar.R	Month	Day
## 1	41	190	5	1
## 2	36	118	5	2
## 3	12	149	5	3
## 4	18	313	5	4
## 5	NA	NA	5	5
## 6	28	NA	5	6
## 7	23	299	5	7
## 8	19	99	5	8
## 9	8	19	5	9
## 10	NA	194	5	10
## 11	12	120	6	19
## 12	13	137	6	20
## 13	NA	150	6	21
## 14	NA	59	6	22
## 15	NA	91	6	23
## 16	NA	250	6	24
## 17	NA	135	6	25
## 18	NA	127	6	26
## 19	NA	47	6	27
## 20	NA	98	6	28

```
airquality %>%  
  fill(Ozone, .direction = "down") %>%  
  slice(c(1:10, 50:59)) %>%  
  select(-c(Temp, wind))
```

##	Ozone	Solar.R	Month	Day
## 1	41	190	5	1
## 2	36	118	5	2
## 3	12	149	5	3
## 4	18	313	5	4
## 5	18	NA	5	5
## 6	28	NA	5	6
## 7	23	299	5	7
## 8	19	99	5	8
## 9	8	19	5	9
## 10	8	194	5	10
## 11	12	120	6	19
## 12	13	137	6	20
## 13	13	150	6	21
## 14	13	59	6	22
## 15	13	91	6	23
## 16	13	250	6	24
## 17	13	135	6	25
## 18	13	127	6	26
## 19	13	47	6	27
## 20	13	98	6	28

replace_na

```
airquality %>%  
  mutate(O3_noNA = replace_na(Ozone, mean(Ozone, na.rm = T)), .after = Ozone) %>%  
  slice(c(1:10, 50:55)) %>%  
  select(-c(Wind, Temp, Solar.R))
```

##	Ozone	O3_noNA	Month	Day
## 1	41	41.00000	5	1
## 2	36	36.00000	5	2
## 3	12	12.00000	5	3
## 4	18	18.00000	5	4
## 5	NA	42.12931	5	5
## 6	28	28.00000	5	6
## 7	23	23.00000	5	7
## 8	19	19.00000	5	8
## 9	8	8.00000	5	9
## 10	NA	42.12931	5	10
## 11	12	12.00000	6	19
## 12	13	13.00000	6	20
## 13	NA	42.12931	6	21
## 14	NA	42.12931	6	22
## 15	NA	42.12931	6	23
## 16	NA	42.12931	6	24