

Ciencia de Datos para Políticas Públicas

Clase 04 - Manejo de Datos II

Pablo Aguirre Hormann

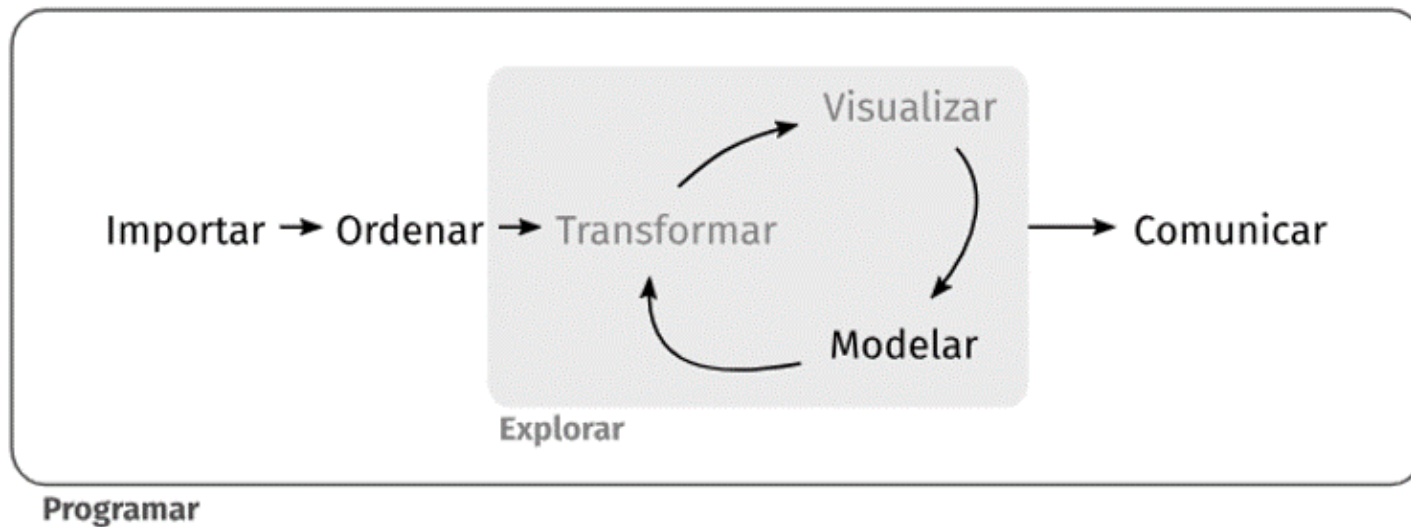
26/08/2020

Antes de empezar

- Dudas
- Preguntas
- Comentarios

¿Qué veremos hoy?

- Un poco más de manejo de datos (`dplyr`)
- Transformación de datos (`tidyr`)
- Funciones e iteraciones



ifelse/case_when

ifelse

```
(x <- 1:10)
```

```
## [1] 1 2 3 4 5 6 7 8 9 10
```

```
ifelse(x >= 4, 1, 0)
```

```
## [1] 0 0 0 1 1 1 1 1 1 1
```

```
ifelse(x == 5, "A", "B")
```

```
## [1] "B" "B" "B" "B" "A" "B" "B" "B" "B" "B"
```

Se puede aplicar en data frames

```
datosONU_tidy <- read_csv("datos/DatosONU_tidy.csv") %>%  
  select(country_name, year) %>%  
  filter(year >= 2004, country_name %in% c("Chile", "Argentina"))  
  
datosONU_tidy %>%  
  mutate(nueva_col = ifelse(year > 2005, 1, 0))
```

```
## # A tibble: 8 x 3  
##   country_name year nueva_col  
##   <chr>      <dbl>     <dbl>  
## 1 Argentina    2004         0  
## 2 Argentina    2005         0  
## 3 Argentina    2006         1  
## 4 Argentina    2007         1  
## 5 Chile        2004         0  
## 6 Chile        2005         0  
## 7 Chile        2006         1  
## 8 Chile        2007         1
```

¿Y si queremos poner más de un valor?

```
datosONU_tidy %>%  
  mutate(nueva_col = ifelse(year > 2005, 1, ifelse(year < 2005, 2, 0)))
```

```
## # A tibble: 8 x 3  
##   country_name  year nueva_col  
##   <chr>        <dbl>    <dbl>  
## 1 Argentina    2004        2  
## 2 Argentina    2005        0  
## 3 Argentina    2006        1  
## 4 Argentina    2007        1  
## 5 Chile        2004        2  
## 6 Chile        2005        0  
## 7 Chile        2006        1  
## 8 Chile        2007        1
```

case_when

```
datosONU_tidy %>%  
  mutate(nueva_col = case_when(  
    year > 2005 ~ 1,  
    year < 2005 ~ 2,  
    TRUE ~ 0  
  ))
```

```
## # A tibble: 8 x 3  
##   country_name  year nueva_col  
##   <chr>        <dbl>    <dbl>  
## 1 Argentina    2004        2  
## 2 Argentina    2005        0  
## 3 Argentina    2006        1  
## 4 Argentina    2007        1  
## 5 Chile        2004        2  
## 6 Chile        2005        0  
## 7 Chile        2006        1  
## 8 Chile        2007        1
```


Transformación de datos

Tidy data - Datos ordenados

- Cada columna es una variable
- Cada fila es una observación
- Cada celda corresponde a un valor

country	year	cases	population
Afghanistan	1999	18145	19987071
Afghanistan	2000	18666	20095360
Brazil	1999	31737	172006362
Brazil	2000	80488	174004898
China	1999	211258	1272015272
China	2000	216766	128042583

variables

country	year	cases	population
Afghanistan	1999	18145	19987071
Afghanistan	2000	18666	20095360
Brazil	1999	31737	172006362
Brazil	2000	80488	174004898
China	1999	211258	1272015272
China	2000	216766	128042583

observations

country	year	cases	population
Afghanistan	1999	18145	19987071
Afghanistan	2000	18666	20095360
Brazil	1999	31737	172006362
Brazil	2000	80488	174004898
China	1999	211258	1272015272
China	2000	216766	128042583

values

tidyr

- `pivot_wider`: transformar “datos largos” a “datos anchos”
- `pivot_longer`: transformar “datos anchos” a “datos largos”
- `separate`: separa columnas
- `unite`: une columnas


Más información en la [web del paquete](#)

tidyr

Principales funciones (i)

- `pivot_wider`

country	year	type	count
A	1999	cases	0.7K
A	1999	pop	19M
A	2000	cases	2K
A	2000	pop	20M
B	1999	cases	37K
B	1999	pop	172M
B	2000	cases	80K
B	2000	pop	174M
C	1999	cases	212K
C	1999	pop	1T
C	2000	cases	213K
C	2000	pop	1T



country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M
C	1999	212K	1T
C	2000	213K	1T

Pivot_wider

```
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
```

```
table2 %>%
```

```
  pivot_wider(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
```

tidyr

Principales funciones (ii)

- `pivot_longer`

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K



country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

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
```

```
table4a %>%
```

```
  pivot_longer(2:3,
               names_to = "year",
               values_to = "value")
```

```
## # A tibble: 6 x 3
##   country    year  value
##   <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
```

Demo - Enfermedades respiratorias

Demo

Script

- Clase04_TransformacionDatosI.R

Enfermedades respiratorias

```
glimpse(who)
```

```
## Rows: 7,240
## Columns: 60
## $ country      <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanis...
## $ iso2          <chr> "AF", "AF", "AF", "AF", "AF", "AF", "AF", "AF", "AF", ...
## $ iso3          <chr> "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "AFG"...
## $ year          <int> 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, ...
## $ new_sp_m014   <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sp_m1524  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sp_m2534  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sp_m3544  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sp_m4554  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sp_m5564  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sp_m65     <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sp_f014   <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sp_f1524  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sp_f2534  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sp_f3544  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sp_f4554  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sp_f5564  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sp_f65     <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sn_m014   <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sn_m1524  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sn_m2534  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sn_m3544  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sn_m4554  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sn_m5564  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sn_m65     <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sn_f014   <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sn_f1524  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sn_f2534  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ new_sn_f3544  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
```

Donde comenzamos

```
(enfermedades <- who %>%  
  select(-iso2, -iso3))
```

```
## # A tibble: 7,240 x 58  
##   country  year new_sp_m014 new_sp_m1524 new_sp_m2534 new_sp_m3544 new_sp_m4554  
##   <chr>   <int>      <int>      <int>      <int>      <int>      <int>  
## 1 Afghan~ 1980         NA         NA         NA         NA         NA  
## 2 Afghan~ 1981         NA         NA         NA         NA         NA  
## 3 Afghan~ 1982         NA         NA         NA         NA         NA  
## 4 Afghan~ 1983         NA         NA         NA         NA         NA  
## 5 Afghan~ 1984         NA         NA         NA         NA         NA  
## 6 Afghan~ 1985         NA         NA         NA         NA         NA  
## 7 Afghan~ 1986         NA         NA         NA         NA         NA  
## 8 Afghan~ 1987         NA         NA         NA         NA         NA  
## 9 Afghan~ 1988         NA         NA         NA         NA         NA  
## 10 Afghan~ 1989         NA         NA         NA         NA         NA  
## # ... with 7,230 more rows, and 51 more variables: new_sp_m5564 <int>,  
## #   new_sp_m65 <int>, new_sp_f014 <int>, new_sp_f1524 <int>,  
## #   new_sp_f2534 <int>, new_sp_f3544 <int>, new_sp_f4554 <int>,  
## #   new_sp_f5564 <int>, new_sp_f65 <int>, new_sn_m014 <int>,  
## #   new_sn_m1524 <int>, new_sn_m2534 <int>, new_sn_m3544 <int>,  
## #   new_sn_m4554 <int>, new_sn_m5564 <int>, new_sn_m65 <int>,  
## #   new_sn_f014 <int>, new_sn_f1524 <int>, new_sn_f2534 <int>,  
## #   new_sn_f3544 <int>, new_sn_f4554 <int>, new_sn_f5564 <int>,  
## #   new_sn_f65 <int>, new_ep_m014 <int>, new_ep_m1524 <int>,  
## #   new_ep_m2534 <int>, new_ep_m3544 <int>, new_ep_m4554 <int>,  
## #   new_ep_m5564 <int>, new_ep_m65 <int>, new_ep_f014 <int>,  
## #   new_ep_f1524 <int>, new_ep_f2534 <int>, new_ep_f3544 <int>,  
## #   new_ep_f4554 <int>, new_ep_f5564 <int>, new_ep_f65 <int>,  
## #   newrel_m014 <int>, newrel_m1524 <int>, newrel_m2534 <int>,  
## #   newrel_m3544 <int>, newrel_m4554 <int>, newrel_m5564 <int>,  
## #   newrel_m65 <int>, newrel_f014 <int>, newrel_f1524 <int>,
```

Donde queremos llegar

tabla_final

```
## # A tibble: 2 x 8
## # Groups:   sexo [2]
##   sexo    `0-14`  `15-24`  `25-34`  `35-44`  `45-54`  `55-64`  `65+`
##   <chr>    <int>    <int>    <int>    <int>    <int>    <int>    <int>
## 1 hombres  97051    406084  495242  478700  417188  325188  288063
## 2 mujeres  99738    320620  347398  260839  184791  136441  129468
```

- ¿Edad?
- ¿Sexo?

¿Qué información tenemos disponible?

names(enfermedades)

```
## [1] "country"      "year"          "new_sp_m014"   "new_sp_m1524"  "new_sp_m2534"
## [6] "new_sp_m3544" "new_sp_m4554"  "new_sp_m5564"  "new_sp_m65"    "new_sp_f014"
## [11] "new_sp_f1524" "new_sp_f2534"  "new_sp_f3544"  "new_sp_f4554"  "new_sp_f5564"
## [16] "new_sp_f65"   "new_sn_m014"   "new_sn_m1524"  "new_sn_m2534"  "new_sn_m3544"
## [21] "new_sn_m4554" "new_sn_m5564"  "new_sn_m65"    "new_sn_f014"   "new_sn_f1524"
## [26] "new_sn_f2534" "new_sn_f3544"  "new_sn_f4554"  "new_sn_f5564"  "new_sn_f65"
## [31] "new_ep_m014"   "new_ep_m1524"  "new_ep_m2534"  "new_ep_m3544"  "new_ep_m4554"
## [36] "new_ep_m5564" "new_ep_m65"    "new_ep_f014"   "new_ep_f1524"  "new_ep_f2534"
## [41] "new_ep_f3544" "new_ep_f4554"  "new_ep_f5564"  "new_ep_f65"    "newrel_m014"
## [46] "newrel_m1524" "newrel_m2534"  "newrel_m3544"  "newrel_m4554"  "newrel_m5564"
## [51] "newrel_m65"   "newrel_f014"   "newrel_f1524"  "newrel_f2534"  "newrel_f3544"
## [56] "newrel_f4554" "newrel_f5564"  "newrel_f65"
```

- `[new o new_]+[enfermedad]+[_]+[sexo]+[RangoEdad]`

¿Pivot_ ... longer o wider?

enfermedades

```
## # A tibble: 7,240 x 58
##   country year new_sp_m014 new_sp_m1524 new_sp_m2534 new_sp_m3544 new_sp_m4554
##   <chr>   <int>      <int>      <int>      <int>      <int>      <int>
## 1 Afghan~ 1980         NA         NA         NA         NA         NA
## 2 Afghan~ 1981         NA         NA         NA         NA         NA
## 3 Afghan~ 1982         NA         NA         NA         NA         NA
## 4 Afghan~ 1983         NA         NA         NA         NA         NA
## 5 Afghan~ 1984         NA         NA         NA         NA         NA
## 6 Afghan~ 1985         NA         NA         NA         NA         NA
## 7 Afghan~ 1986         NA         NA         NA         NA         NA
## 8 Afghan~ 1987         NA         NA         NA         NA         NA
## 9 Afghan~ 1988         NA         NA         NA         NA         NA
## 10 Afghan~ 1989         NA         NA         NA         NA         NA
## # ... with 7,230 more rows, and 51 more variables: new_sp_m5564 <int>,
## #   new_sp_m65 <int>, new_sp_f014 <int>, new_sp_f1524 <int>,
## #   new_sp_f2534 <int>, new_sp_f3544 <int>, new_sp_f4554 <int>,
## #   new_sp_f5564 <int>, new_sp_f65 <int>, new_sn_m014 <int>,
## #   new_sn_m1524 <int>, new_sn_m2534 <int>, new_sn_m3544 <int>,
## #   new_sn_m4554 <int>, new_sn_m5564 <int>, new_sn_m65 <int>,
## #   new_sn_f014 <int>, new_sn_f1524 <int>, new_sn_f2534 <int>,
## #   new_sn_f3544 <int>, new_sn_f4554 <int>, new_sn_f5564 <int>,
## #   new_sn_f65 <int>, new_ep_m014 <int>, new_ep_m1524 <int>,
```

¿Pivot_ ... longer o wider?

Transformamos la forma de nuestros datos

```
(enfermedades2 <- enfermedades %>%  
  pivot_longer(-c(country:year),  
               names_to = "variables",  
               values_to = "valores"))
```

```
## # A tibble: 405,440 x 4  
##   country      year variables    valores  
##   <chr>      <int> <chr>      <int>  
## 1 Afghanistan 1980 new_sp_m014      NA  
## 2 Afghanistan 1980 new_sp_m1524      NA  
## 3 Afghanistan 1980 new_sp_m2534      NA  
## 4 Afghanistan 1980 new_sp_m3544      NA  
## 5 Afghanistan 1980 new_sp_m4554      NA  
## 6 Afghanistan 1980 new_sp_m5564      NA  
## 7 Afghanistan 1980 new_sp_m65      NA  
## 8 Afghanistan 1980 new_sp_f014      NA  
## 9 Afghanistan 1980 new_sp_f1524      NA  
## 10 Afghanistan 1980 new_sp_f2534      NA  
## # ... with 405,430 more rows
```

- Tenemos una columna que contiene tres variables pero también una parte que no nos sirve mucho.

Reference

Pattern matching

<code>str_count()</code>	Count the number of matches in a string
<code>str_detect()</code>	Detect the presence or absence of a pattern in a string
<code>str_extract()</code>	Extract matching patterns from a string
<code>str_extract_all()</code>	
<code>str_locate()</code> <code>str_locate_all()</code>	Locate the position of patterns in a string
<code>str_match()</code> <code>str_match_all()</code>	Extract matched groups from a string

Eliminaremos parte de la columna “variables” que no nos sirve

```
(enfermedades3 <- enfermedades2 %>%  
  mutate(variables = str_remove(variables, "new_"),  
         variables = str_remove(variables, "new")))
```

```
## # A tibble: 405,440 x 4  
##   country      year variables valores  
##   <chr>        <int> <chr>      <int>  
## 1 Afghanistan  1980 sp_m014      NA  
## 2 Afghanistan  1980 sp_m1524     NA  
## 3 Afghanistan  1980 sp_m2534     NA  
## 4 Afghanistan  1980 sp_m3544     NA  
## 5 Afghanistan  1980 sp_m4554     NA  
## 6 Afghanistan  1980 sp_m5564     NA  
## 7 Afghanistan  1980 sp_m65      NA  
## 8 Afghanistan  1980 sp_f014     NA  
## 9 Afghanistan  1980 sp_f1524     NA  
## 10 Afghanistan 1980 sp_f2534     NA  
## # ... with 405,430 more rows
```

- Ahora tenemos una columna que contiene tres variables. ¿Cómo las separamos?

separate (i)

```
enfermedades3
```

```
## # A tibble: 405,440 x 4
##   country      year variables valores
##   <chr>      <int> <chr>      <int>
## 1 Afghanistan 1980 sp_m014      NA
## 2 Afghanistan 1980 sp_m1524     NA
## 3 Afghanistan 1980 sp_m2534     NA
## 4 Afghanistan 1980 sp_m3544     NA
## 5 Afghanistan 1980 sp_m4554     NA
## 6 Afghanistan 1980 sp_m5564     NA
## 7 Afghanistan 1980 sp_m65      NA
## 8 Afghanistan 1980 sp_f014     NA
## 9 Afghanistan 1980 sp_f1524     NA
## 10 Afghanistan 1980 sp_f2534     NA
## # ... with 405,430 more rows
```

```
(enfermedades4 <- enfermedades3 %>%
  separate(variables,
    into = c("enfermedad", "otro"),
    sep = "_"))
```

```
## # A tibble: 405,440 x 5
##   country      year enfermedad otro  valores
##   <chr>      <int> <chr>      <chr>  <int>
## 1 Afghanistan 1980 sp      m014      NA
## 2 Afghanistan 1980 sp      m1524     NA
## 3 Afghanistan 1980 sp      m2534     NA
## 4 Afghanistan 1980 sp      m3544     NA
## 5 Afghanistan 1980 sp      m4554     NA
## 6 Afghanistan 1980 sp      m5564     NA
## 7 Afghanistan 1980 sp      m65      NA
## 8 Afghanistan 1980 sp      f014     NA
## 9 Afghanistan 1980 sp      f1524     NA
## 10 Afghanistan 1980 sp      f2534     NA
## # ... with 405,430 more rows
```

separate (ii)

```
enfermedades4
```

```
## # A tibble: 405,440 x 5
##   country      year enfermedad otro  valores
##   <chr>        <int> <chr>      <chr>  <int>
## 1 Afghanistan  1980 sp          m014    NA
## 2 Afghanistan  1980 sp          m1524   NA
## 3 Afghanistan  1980 sp          m2534   NA
## 4 Afghanistan  1980 sp          m3544   NA
## 5 Afghanistan  1980 sp          m4554   NA
## 6 Afghanistan  1980 sp          m5564   NA
## 7 Afghanistan  1980 sp          m65     NA
## 8 Afghanistan  1980 sp          f014    NA
## 9 Afghanistan  1980 sp          f1524   NA
## 10 Afghanistan 1980 sp          f2534   NA
## # ... with 405,430 more rows
```

```
(enfermedades5 <- enfermedades4 %>%
  separate(otro,
    into = c("sexo", "edad"),
    sep = 1))
```

```
## # A tibble: 405,440 x 6
##   country      year enfermedad sexo  edad  valores
##   <chr>        <int> <chr>      <chr> <chr>  <int>
## 1 Afghanistan  1980 sp          m    014    NA
## 2 Afghanistan  1980 sp          m   1524   NA
## 3 Afghanistan  1980 sp          m   2534   NA
## 4 Afghanistan  1980 sp          m   3544   NA
## 5 Afghanistan  1980 sp          m   4554   NA
## 6 Afghanistan  1980 sp          m   5564   NA
## 7 Afghanistan  1980 sp          m    65    NA
## 8 Afghanistan  1980 sp          f    014   NA
## 9 Afghanistan  1980 sp          f   1524   NA
## 10 Afghanistan 1980 sp          f   2534   NA
## # ... with 405,430 more rows
```

Se podría llegar a lo mismo usando **stringr**

```
enfermedades2 %>%
  transmute(country, year,
    enfermedad = case_when(
      str_detect(variables, "rel") ~
        str_sub(variables, 4, 6),
      TRUE ~ str_sub(variables, 5,6)),
    sexo = case_when(
      str_detect(variables, "m") ~ "m",
      TRUE ~ "f"),
    edad = str_extract(variables, "\\d+"),
    valores
  )
```

```
## # A tibble: 405,440 x 6
##   country      year enfermedad sexo  edad  valores
##   <chr>      <int> <chr>      <chr> <chr>    <int>
## 1 Afghanistan 1980 sp          m    014      NA
## 2 Afghanistan 1980 sp          m   1524      NA
## 3 Afghanistan 1980 sp          m   2534      NA
## 4 Afghanistan 1980 sp          m   3544      NA
## 5 Afghanistan 1980 sp          m   4554      NA
## 6 Afghanistan 1980 sp          m   5564      NA
## 7 Afghanistan 1980 sp          m    65      NA
## 8 Afghanistan 1980 sp          f    014      NA
## 9 Afghanistan 1980 sp          f   1524      NA
## 10 Afghanistan 1980 sp          f   2534      NA
## # ... with 405,430 more rows
```

Modificaremos algunos valores para mayor claridad

```
(enfermedades6 <- enfermedades5 %>%  
  mutate(  
    edad = case_when(  
      edad == "014" ~ "0-14",  
      edad == "1524" ~ "15-24",  
      edad == "2534" ~ "25-34",  
      edad == "3544" ~ "35-44",  
      edad == "4554" ~ "45-54",  
      edad == "5564" ~ "55-64",  
      edad == "65" ~ "65+",  
    ),  
    sexo = case_when(  
      sexo == "m" ~ "hombres",  
      sexo == "f" ~ "mujeres"  
    ))
```

```
## # A tibble: 405,440 x 6  
##   country      year enfermedad sexo    edad  valores  
##   <chr>      <int> <chr>    <chr>  <chr>  <int>  
## 1 Afghanistan 1980 sp      hombres 0-14      NA  
## 2 Afghanistan 1980 sp      hombres 15-24     NA  
## 3 Afghanistan 1980 sp      hombres 25-34     NA  
## 4 Afghanistan 1980 sp      hombres 35-44     NA  
## 5 Afghanistan 1980 sp      hombres 45-54     NA  
## 6 Afghanistan 1980 sp      hombres 55-64     NA  
## 7 Afghanistan 1980 sp      hombres 65+       NA  
## 8 Afghanistan 1980 sp      mujeres 0-14     NA  
## 9 Afghanistan 1980 sp      mujeres 15-24     NA  
## 10 Afghanistan 1980 sp      mujeres 25-34     NA  
## # ... with 405,430 more rows
```

Calculamos el total de enfermedades por sexo y rango de edad

```
(resumen_enfermedades <- enfermedades6 %>%  
  filter(year == 2010) %>%  
  group_by(sexo, edad) %>%  
  summarise(total = sum(valores, na.rm = TRUE)))
```

```
## # A tibble: 14 x 3  
## # Groups:   sexo [2]  
##   sexo    edad  total  
##   <chr>  <chr> <int>  
## 1 hombres 0-14   97051  
## 2 hombres 15-24 406084  
## 3 hombres 25-34 495242  
## 4 hombres 35-44 478700  
## 5 hombres 45-54 417188  
## 6 hombres 55-64 325188  
## 7 hombres 65+   288063  
## 8 mujeres 0-14   99738  
## 9 mujeres 15-24 320620  
## 10 mujeres 25-34 347398  
## 11 mujeres 35-44 260839  
## 12 mujeres 45-54 184791  
## 13 mujeres 55-64 136441  
## 14 mujeres 65+   129468
```

Pivot_wider para dar la forma final a la tabla

```
(tabla_final <- resumen_enfermedades %>%  
  pivot_wider(names_from = edad, values_from = total))
```

```
## # A tibble: 2 x 8
```

```
## # Groups:   sexo [2]
```

```
##   sexo    `0-14`  `15-24`  `25-34`  `35-44`  `45-54`  `55-64`  `65+`  
##   <chr>    <int>    <int>    <int>    <int>    <int>    <int>    <int>  
## 1 hombres  97051    406084    495242    478700    417188    325188    288063  
## 2 mujeres  99738    320620    347398    260839    184791    136441    129468
```


Resumen

```
tabla_final <- who %>%  
  # Eliminar columnas que no usaremos  
  select(-iso2, -iso3) %>%  
  # Ajustar forma de los datos (de ancho a largo)  
  pivot_longer(-c(country:year), names_to = "variables", values_to = "valores", values_drop_na = TRUE) %>%  
  # Extraer información de la columna "variables"  
  mutate(variables = str_remove(variables, "new_"),  
          variables = str_remove(variables, "new")) %>%  
  separate(variables, into = c("enfermedad", "otro"), sep = "_") %>%  
  separate(otro, into = c("sexo", "edad"), sep = " ") %>%  
  # Re-codificar las columnas edad y sexo  
  mutate(edad = case_when(  
    edad == "014" ~ "0-14",  
    edad == "1524" ~ "15-24",  
    edad == "2534" ~ "25-34",  
    edad == "3544" ~ "35-44",  
    edad == "4554" ~ "45-54",  
    edad == "5564" ~ "55-64",  
    edad == "65" ~ "65+"),  
    sexo = case_when(  
      sexo == "m" ~ "hombres",  
      sexo == "f" ~ "mujeres")) %>%  
  # Generar tabla final para el año 2010  
  filter(year == 2010) %>%  
  group_by(sexo, edad) %>%  
  summarise(total = sum(valores, na.rm = TRUE)) %>%  
  pivot_wider(names_from = edad, values_from = total)
```

Ejercicio

Ejercicio

Script

- Clase04_EjercicioI.R

Funciones e iteraciones

Funciones

- Hasta ahora hemos visto funciones que vienen por defecto en R o en paquetes que cargamos
- Pero también podemos generar nuestras propias funciones
 - ¿Para qué?
- Si se encuentran con un script que repite muchas veces (¿más de 3?) probablemente sea necesaria una función

¿Cómo se crea una función?

- Una función tiene tres partes
 - Nombre
 - Argumentos
 - Cuerpo

```
ElevaryDividir <- function(x, y, z){  
  (x^y)/z  
}
```

```
ElevaryDividir(3, 2, 4)
```

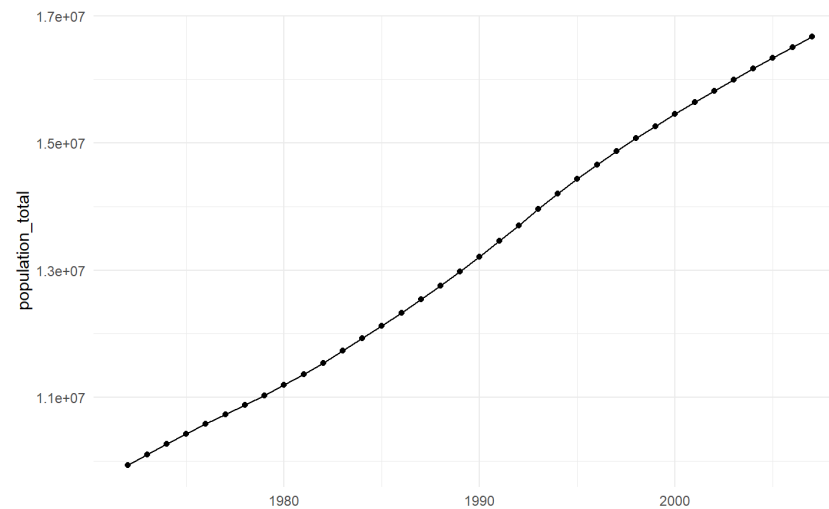
```
## [1] 2.25
```

Ejemplo

Estamos explorando datos y queremos ir graficando distintos indicadores para distintos países

```
datosONU_tidy <- read_csv("datos/DatosONU_tidy.csv")
```

```
datosONU_tidy %>%  
  filter(country_name == "Chile") %>%  
  ggplot(aes(x = year, y = population_total)) +  
    geom_point() +  
    geom_line() +  
    labs(x = NULL) +  
    theme_minimal()
```



Crear una función que haga esto

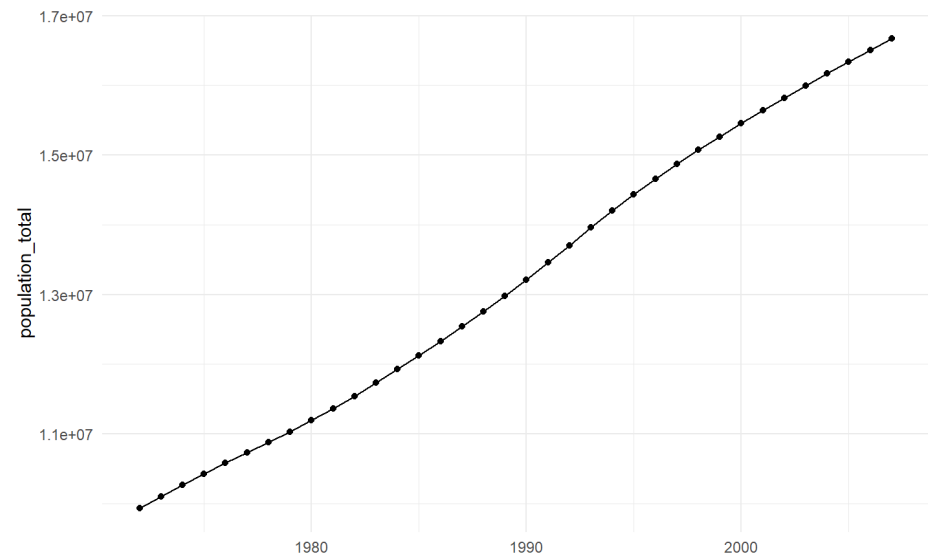
```
graf_indicador_pais <- function(x, y){
```

```
  y <- enquo(y)
```

```
  datosONU_tidy %>%  
    filter(country_name == x) %>%  
    ggplot(aes(x = year, y = !!y)) +  
    geom_point() +  
    geom_line() +  
    labs(x = NULL) +  
    theme_minimal()
```

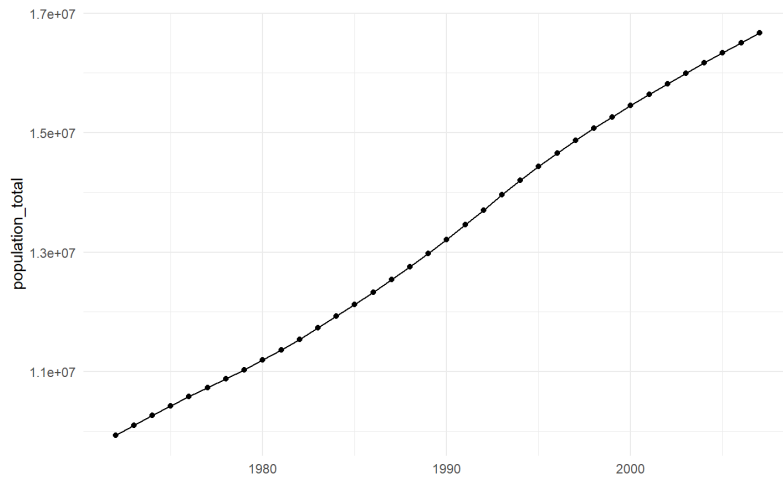
```
}
```

```
graf_indicador_pais("Chile", population_total)
```

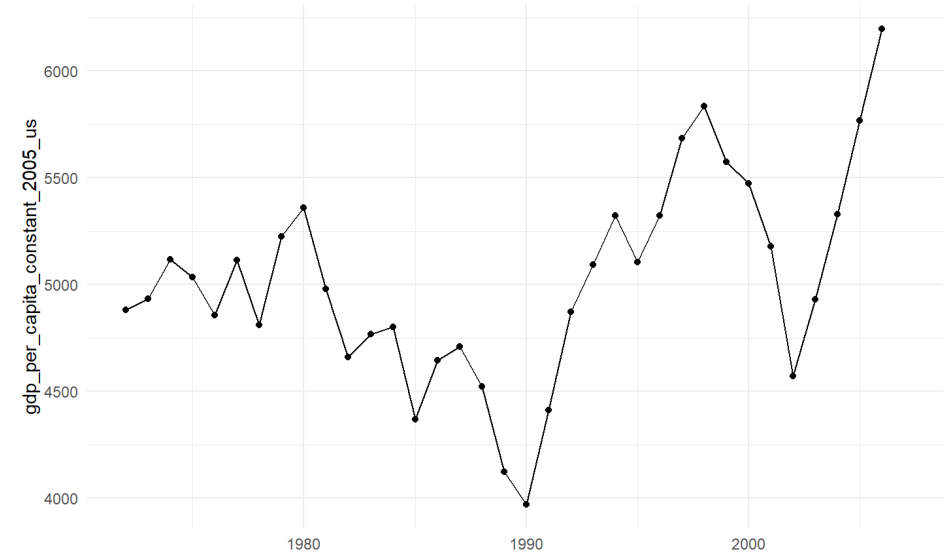


Y ahora podemos hacer muchas cosas más

```
graf_indicador_pais("Chile", population_total)
```



```
graf_indicador_pais("Argentina",  
gdp_per_capita_constant_2005_us)
```



Iteraciones

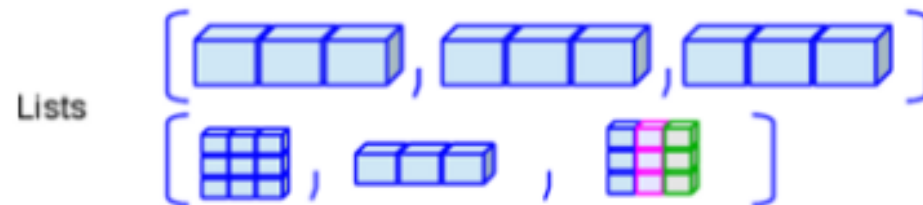
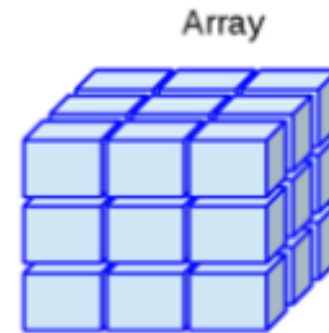
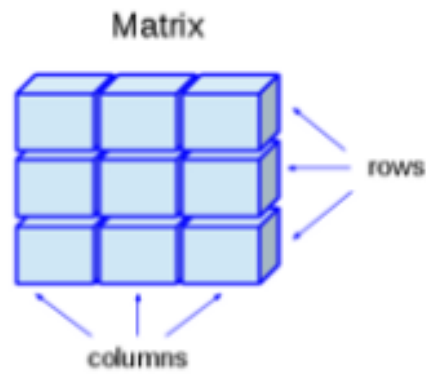
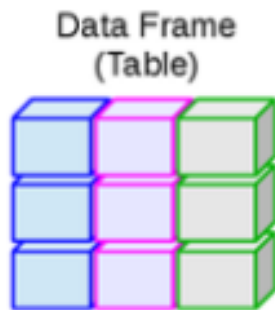
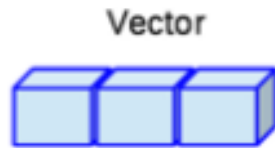
¿Para qué?

Digamos que queremos graficar el mismo indicador para varios países:

```
graf_indicador_pais("Chile", gdp_per_capita_constant_2005_us)  
graf_indicador_pais("Argentina", gdp_per_capita_constant_2005_us)  
graf_indicador_pais("United States", gdp_per_capita_constant_2005_us)
```

Antes de seguir

Tipos de objetos



for loops

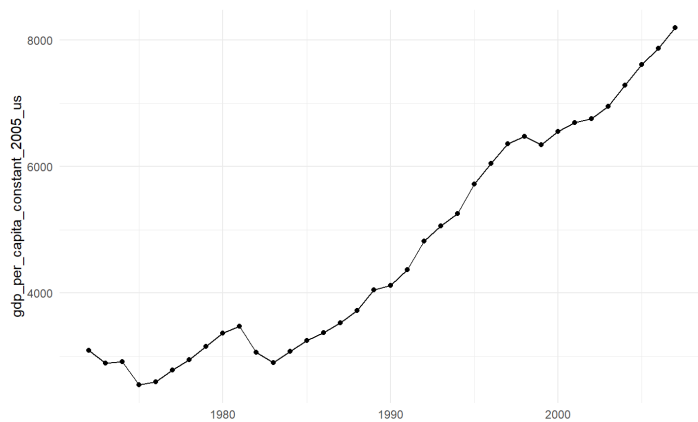
```
for (i in 1:10){  
  print(2^i)  
}
```

```
## [1] 2  
## [1] 4  
## [1] 8  
## [1] 16  
## [1] 32  
## [1] 64  
## [1] 128  
## [1] 256  
## [1] 512  
## [1] 1024
```

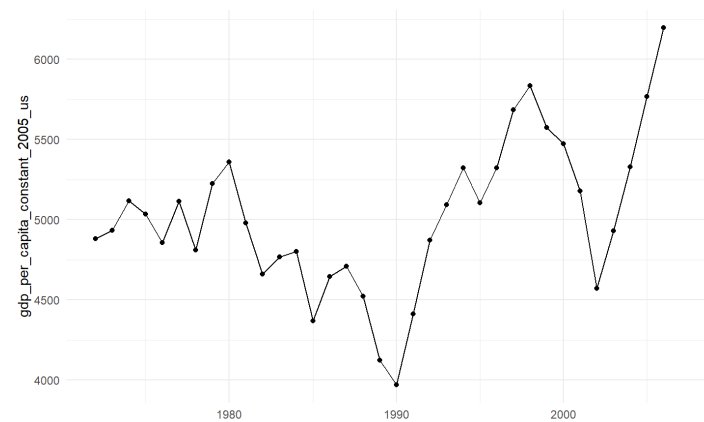
for loop aplicado a nuestro ejemplo

```
graficos <- list()
países <- c("Chile", "Argentina", "United States")
for (i in seq_along(países)){
  graficos[[i]] <- graf_indicador_pais(países[i], gdp_per_capita_constant_2005_us)
}
```

graficos[[1]]



graficos[[2]]

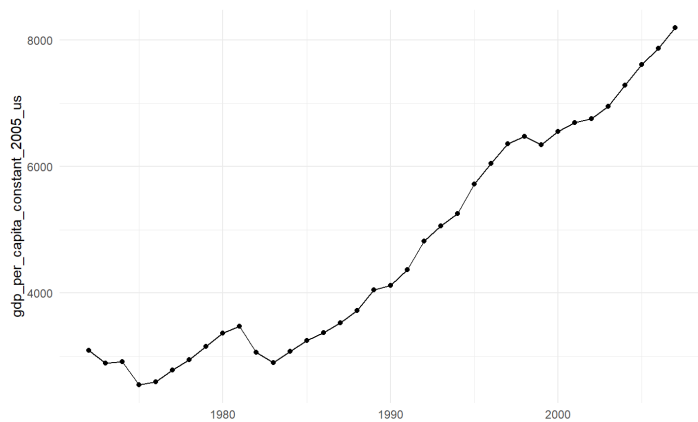


Otra opción

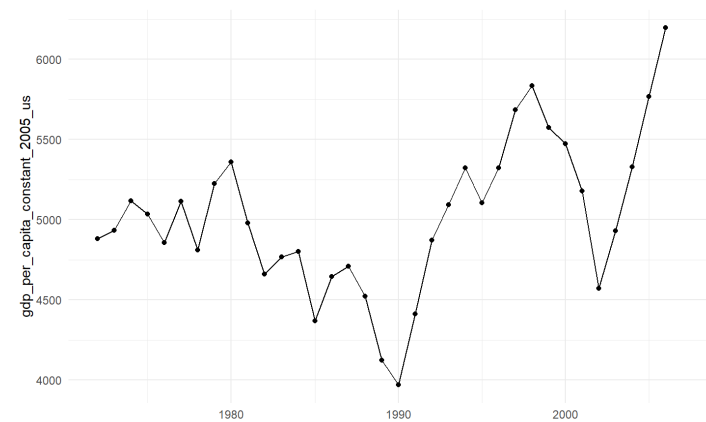
map

```
library(purrr)
países <- c("Chile", "Argentina", "United States")
graficos <- map(países, graf_indicador_pais, gdp_per_capita_constant_2005_us)
```

graficos[[1]]



graficos[[2]]



Funciones **map**

- `map()`: resultado es una lista
- `map_lgl()`: resultado es un vector de valores lógicos
- `map_int()`: resultado es un vector de números integrales
- `map_dbl()`: resultado es un vector de números decimales
- `map_chr()`: resultado es un vector de valores tipo texto

Ejercicio

Ejercicio

Script

- Clase04_EjercicioII.R

¿Qué se viene?

- Próxima clase: regresión vs clasificación
- Tarea 2: sábado 5 de septiembre
- Vayan recopilando sus datos
 - 21 de septiembre: 1ra entrega