

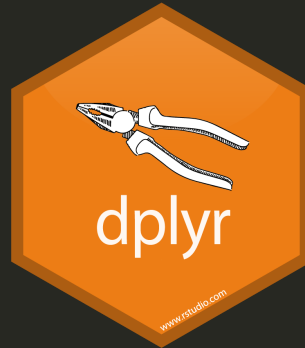
# Manipulación de datos

  
con dplyr

Curso Aprender R desde cero

FaCENA - UNNE

26 de Agosto de 2019



# ¿Qué vamos a ver hoy?

Funciones útiles

Funciones Principales

# Repo del curso



<https://github.com/PatriLoto/AprendeR-desde-cero>

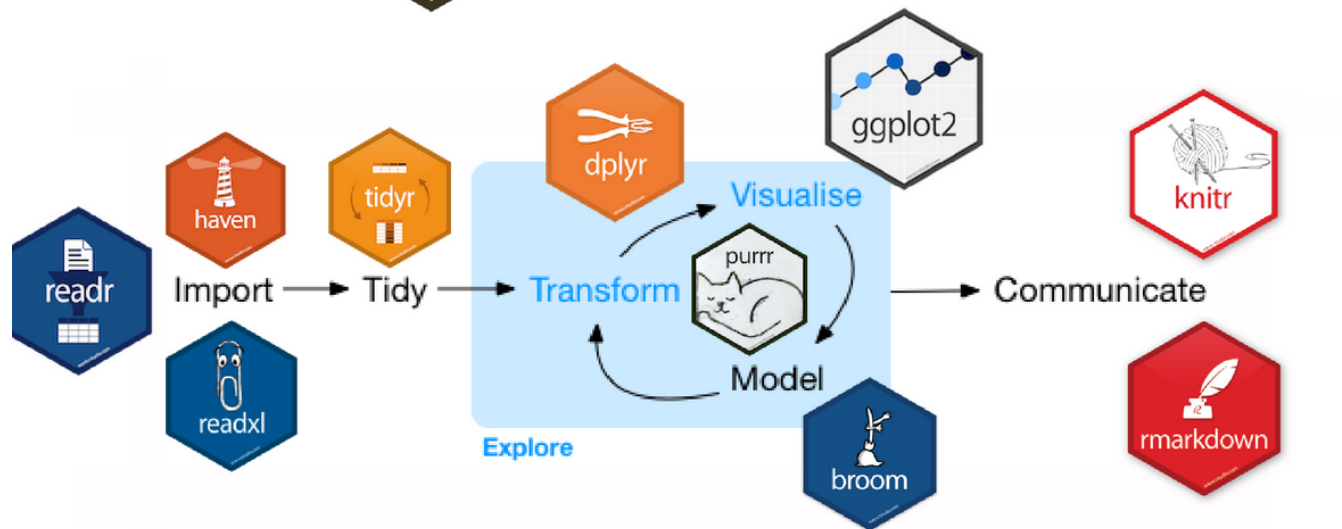
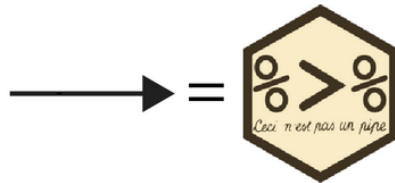
# Repo de la charla

<https://github.com/data-datum/dplyr-manipulation>

# Vignette de dplyr

<https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html>

# Flujo de trabajo para Ciencia de Datos



data.frames

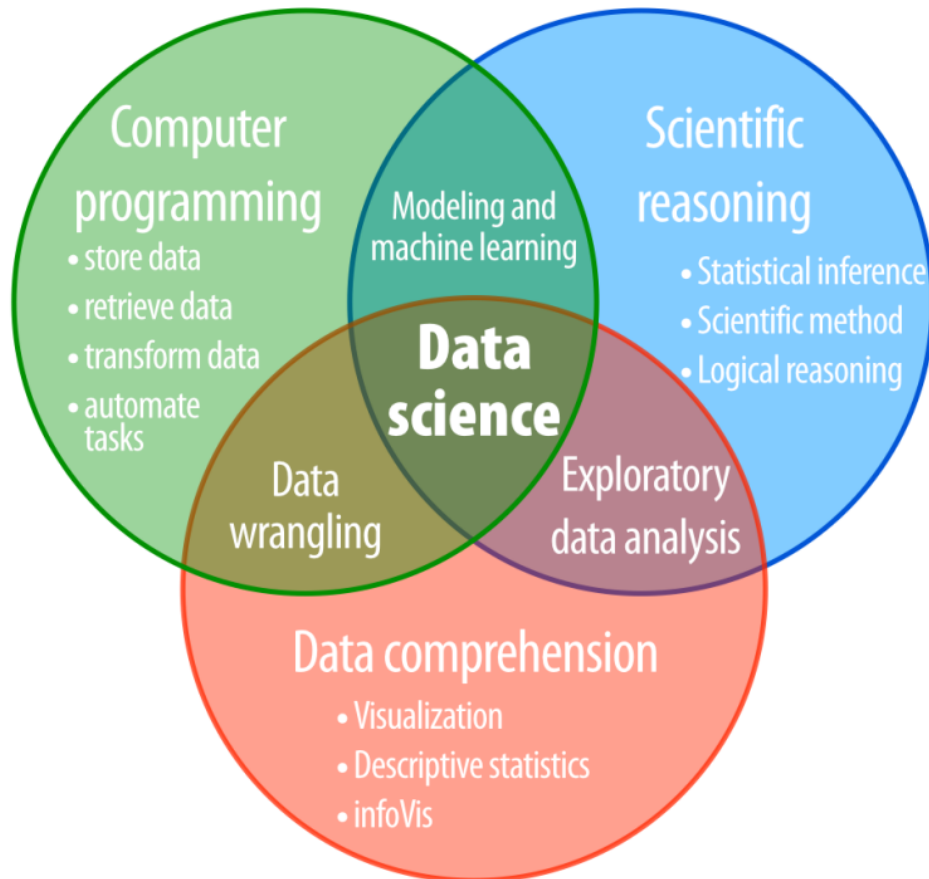


factors



strings





[\*] *Hands on programming with R* by Garret Grolemund

# Dataset: gapminder

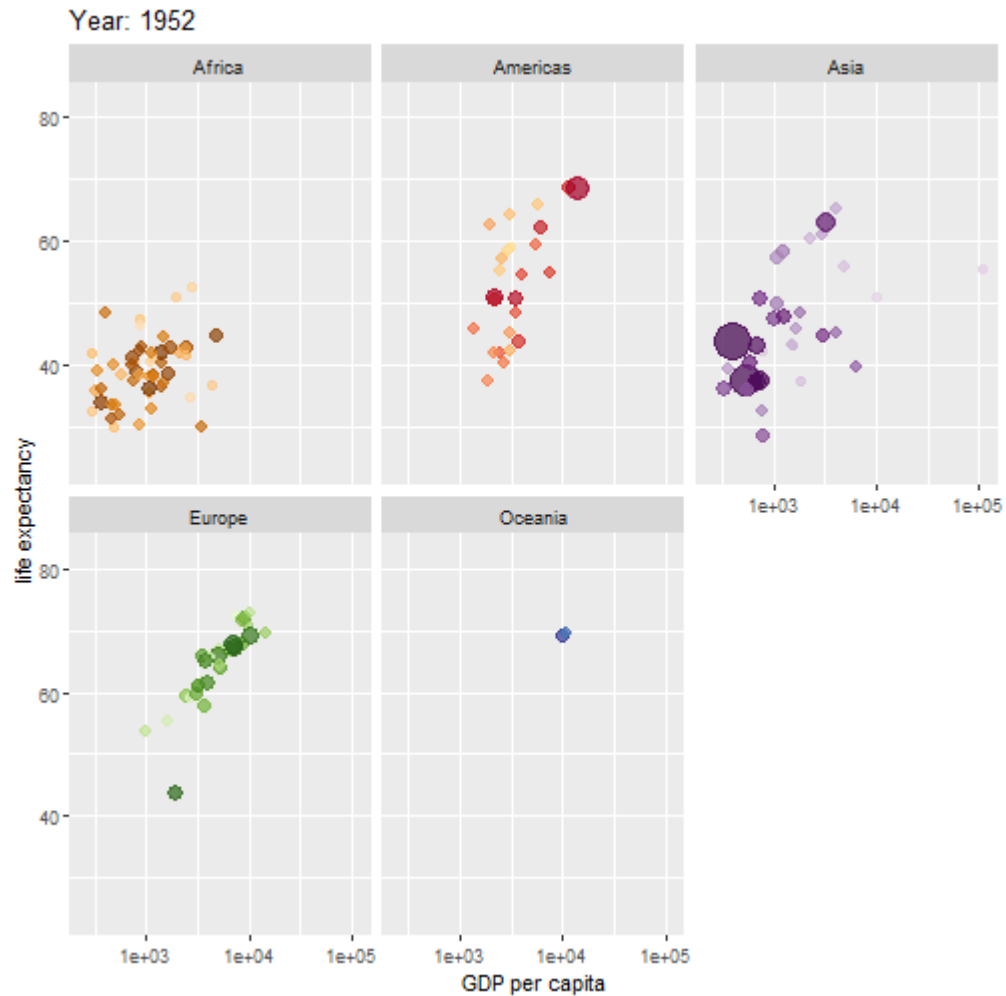
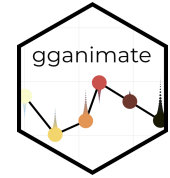


```
install.packages("gapminder")  
install.packages("dplyr")
```

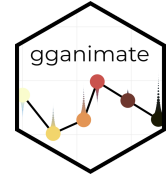
Gapminder es un set de datos que contiene en total 6 columnas correspondientes a:

- country(país)
- continent (continente)
- year (año) desde 1952 a 2007
- lifeExp (esperanza de vida)
- pop (población total)
- gdpPercap (ingreso per Cápita) y 1704 observaciones

# gapminder en el tiempo



# El código



```
library(gganimate)
ggplot(gapminder, aes(gdpPercap, lifeExp, size = pop, colour = country)) +
  geom_point(alpha = 0.7) +
  scale_colour_manual(values = country_colors) +
  scale_size(range = c(2, 12)) +
  scale_x_log10() +
  facet_wrap(~continent) +
  theme(legend.position = 'none') +
  labs(title = 'Year: {frame_time}', x = 'GDP per capita', y = 'life
transition_time(year))
```

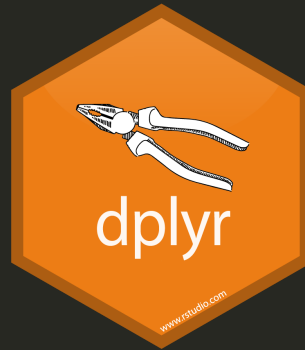


# data.frame vs. tibble



- `data.frame` es una estructura nativas de `rbase` y está disponible sin necesidad de instalar ningún paquete adicional. Es una estructura tabular organizada en filas y columnas. Se puede acceder a las columnas mediante `colnames()` y los nombres de las filas, como `rownames()`.
- `tibble`, sin embargo, es parte de `tidyverse`. También presenta una estructura tabular, en filas y columnas. Los tibbles eliminan los `rownames` por defecto, para tener mejor compatibilidad con bases de datos SQL.
- Ambas formas son intercambiables mediante los comandos

```
rbase::as.data.frame()  
tidyverse::as_tibble()
```



# Funciones Útiles

# Funciones Útiles



## Cargamos las librerías

```
library(dplyr)
library(gapminder)
```

## head()

```
head(gapminder)
```

```
## # A tibble: 6 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.
```

# Funciones Útiles



## tail()

```
tail(gapminder)
```

```
## # A tibble: 6 x 6
##   country continent  year lifeExp      pop gdpPercap
##   <fct>      <fct>    <int>   <dbl>    <int>    <dbl>
## 1 Zimbabwe Africa    1982   60.4  7636524   789.
## 2 Zimbabwe Africa    1987   62.4  9216418   706.
## 3 Zimbabwe Africa    1992   60.4 10704340   693.
## 4 Zimbabwe Africa    1997   46.8 11404948   792.
## 5 Zimbabwe Africa    2002   40.0 11926563   672.
## 6 Zimbabwe Africa    2007   43.5 12311143   470.
```

# Funciones Útiles

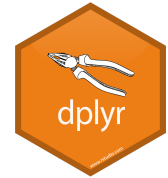


## glimpse()

```
glimpse(gapminder)
```

```
## Observations: 1,704
## Variables: 6
## $ country    <fct> Afghanistan, Afghanistan, Afghanistan, Afghanistan, ...
## $ continent  <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia...
## $ year       <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992...
## $ lifeExp    <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.8...
## $ pop        <int> 8425333, 9240934, 10267083, 11537966, 13079460, 1488...
## $ gdpPercap  <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 78...
```

# Funciones Útiles



## slice()

```
dplyr::slice(gapminder, 5L)
```

```
## # A tibble: 1 x 6
##   country      continent  year lifeExp      pop gdpPercap
##   <fct>        <fct>    <int>   <dbl>   <int>   <dbl>
## 1 Afghanistan Asia      1972   36.1 13079460    740.
```

# Funciones Útiles



## str()

```
str(gapminder)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    1704 obs. of  6 variables:
##  $ country   : Factor w/ 142 levels "Afghanistan",...: 1 1 1 1 1 1 1 1 1 1 ...
##  $ continent: Factor w/ 5 levels "Africa","Americas",...: 3 3 3 3 3 3 3 3 3 3 ...
##  $ year      : int   1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
##  $ lifeExp   : num    28.8 30.3 32 34 36.1 ...
##  $ pop       : int   8425333 9240934 10267083 11537966 13079460 14880372 128...
##  $ gdpPercap: num    779 821 853 836 740 ...
```

# Funciones Útiles



## summary()

```
summary(gapminder)
```

```
##           country      continent      year      lifeExp
## Afghanistan: 12 Africa :624 Min. :1952 Min. :23.60
## Albania : 12 Americas:300 1st Qu.:1966 1st Qu.:48.20
## Algeria : 12 Asia :396 Median :1980 Median :60.71
## Angola : 12 Europe :360 Mean :1980 Mean :59.47
## Argentina : 12 Oceania : 24 3rd Qu.:1993 3rd Qu.:70.85
## Australia : 12 Max. :2007 Max. :82.60
## (Other) :1632
##           pop      gdpPercap
## Min. :6.001e+04 Min. : 241.2
## 1st Qu.:2.794e+06 1st Qu.: 1202.1
## Median :7.024e+06 Median : 3531.8
## Mean :2.960e+07 Mean : 7215.3
## 3rd Qu.:1.959e+07 3rd Qu.: 9325.5
## Max. :1.319e+09 Max. :113523.1
##
```





# Operador pipe

%>%

# Operador pipe %>%



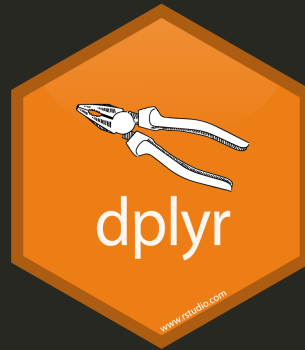
- El operador pipe nos permite concatenar funciones y objetos en R, de forma que el código tenga un aspecto más ordenado, siguiendo la filosofía de tidyverse.

```
library(magrittr)
gapminder %>%
  filter(country=="Argentina")%>%
  select(gdpPercap, year) %>%
  head(2)
```

```
## # A tibble: 2 x 2
##   gdpPercap  year
##   <dbl> <int>
## 1    5911.  1952
## 2    6857.  1957
```

```
head(select(filter(gapminder, country=="Argentina"), gdpPercap, year))
```

```
## # A tibble: 2 x 2
##   gdpPercap  year
```



# Funciones principales de dplyr

`select()`: extraer columnas

`filter()`: extraer filas siguiendo una restricción lógica

`mutate()`: crea nuevas variables

`summarise()`: cambiar la unidad de análisis

`arrange()`: ordenar filas por variables

`group_by()`: agrupar según un determinado criterio

# select()



- Podemos extraer columnas

```
gapminder %>%  
  select(country, lifeExp, gdpPercap)
```

```
## # A tibble: 1,704 x 3  
##   country      lifeExp gdpPercap  
##   <fct>         <dbl>     <dbl>  
## 1 Afghanistan    28.8       779.  
## 2 Afghanistan    30.3       821.  
## 3 Afghanistan    32.0       853.  
## 4 Afghanistan    34.0       836.  
## 5 Afghanistan    36.1       740.  
## 6 Afghanistan    38.4       786.  
## 7 Afghanistan    39.9       978.  
## 8 Afghanistan    40.8       852.  
## 9 Afghanistan    41.7       649.  
## 10 Afghanistan   41.8       635.  
## # ... with 1,694 more rows
```

# Funciones útiles para combinar con `select()`



función	descripción
-	selecciona todas menos
:	selecciona un rango
contains()	selecciona variables cuyo nombre contiene...
start_with()	selecciona variables cuyo nombre empieza con ...
ends_with()	selecciona variables cuyo nombre empieza con...
everything()	selecciona todas las columnas
matches()	selecciona variables cuyo nombre coincide con...
num_range()	selecciona variables por posición

# Si tenemos que seleccionar las mismas columnas varias veces



```
cols<-c("country", "lifeExp", "gdpPercap")

gapminder %>%
  select(!!cols)
```

```
## # A tibble: 1,704 x 3
##   country      lifeExp gdpPercap
##   <fct>         <dbl>     <dbl>
## 1 Afghanistan  28.8       779.
## 2 Afghanistan  30.3       821.
## 3 Afghanistan  32.0       853.
## 4 Afghanistan  34.0       836.
## 5 Afghanistan  36.1       740.
## 6 Afghanistan  38.4       786.
## 7 Afghanistan  39.9       978.
## 8 Afghanistan  40.8       852.
## 9 Afghanistan  41.7       649.
## 10 Afghanistan 41.8       635.
## # ... with 1,694 more rows
```

# Podemos seleccionar según una expresión regular (regex)



```
gapminder %>%  
  select(matches("gdp")) %>%  
  head
```

```
## # A tibble: 6 x 1  
##   gdpPercap  
##   <dbl>  
## 1      779.  
## 2      821.  
## 3      853.  
## 4      836.  
## 5      740.  
## 6      786.
```



# filter()



- Realizamos el filtrado según un valor numérico de una variable. Para ello utilizamos los siguientes operadores: >, >=, <, <=, ==, !=

```
gapminder %>%  
  filter(lifeExp > 80) %>%  
  head(3)
```

```
## # A tibble: 3 x 6  
##   country    continent  year lifeExp      pop gdpPercap  
##   <fct>      <fct>      <int>  <dbl>    <int>    <dbl>  
## 1 Australia Oceania      2002   80.4  19546792  30688.  
## 2 Australia Oceania      2007   81.2  20434176  34435.  
## 3 Canada    Americas      2007   80.7  33390141  36319.
```

# filter()



Podemos filtrar según múltiples condiciones:

- Se cumplen ambas condiciones

```
filter(condicion1, condicion2)
```

- Se cumple la condición 1 solamente

```
filter(condicion1, !condicion2)
```

- Se cumplen la condición 1 y/ o la condición 2

```
filter(condicion1 | condicion2)
```

- Se cumple una u otra condición, pero no ambas (disyunción exclusiva)

```
filter(xor(condicion1, condicion2))
```

# filter()



- Si queremos realizar un filtrado según un rango, lo hacemos de la siguiente manera

```
gapminder %>%  
  select (country, lifeExp, year) %>%  
  filter(lifeExp >= 60, lifeExp < 85) %>%  
  head
```

```
## # A tibble: 6 x 3  
##   country lifeExp  year  
##   <fct>     <dbl> <int>  
## 1 Albania    64.8  1962  
## 2 Albania    66.2  1967  
## 3 Albania    67.7  1972  
## 4 Albania    68.9  1977  
## 5 Albania    70.4  1982  
## 6 Albania    72    1987
```

# mutate()



- `mutate()` es una función que nos permite crear una nueva columna en un tibble, realizando una operación con otras ya existentes.

```
gapminder %>%  
  mutate(gdp = pop * gdpPercap) %>%  
  head
```

```
## # A tibble: 6 x 7  
##   country      continent  year lifeExp      pop gdpPercap      gdp  
##   <fct>        <fct>    <int>  <dbl>    <int>    <dbl>    <dbl>  
## 1 Afghanistan Asia      1952   28.8  8425333    779.  6567086330.  
## 2 Afghanistan Asia      1957   30.3  9240934    821.  7585448670.  
## 3 Afghanistan Asia      1962   32.0 10267083    853.  8758855797.  
## 4 Afghanistan Asia      1967   34.0 11537966    836.  9648014150.  
## 5 Afghanistan Asia      1972   36.1 13079460    740.  9678553274.  
## 6 Afghanistan Asia      1977   38.4 14880372    786. 11697659231.
```

# Si queremos conservar sólo la nueva columna; usamos transmute()



```
gapminder %>%  
  transmute(gdp = pop * gdpPercap) %>%  
  head
```

```
## # A tibble: 6 x 1  
##           gdp  
##       <dbl>  
## 1  6567086330.  
## 2  7585448670.  
## 3  8758855797.  
## 4  9648014150.  
## 5  9678553274.  
## 6 11697659231.
```

# summarise()



```
gdp_bycontinents <- gapminder %>%  
  group_by(continent) %>%  
  summarize(mean_gdpPercap=mean(gdpPercap))  
gdp_bycontinents
```

```
## # A tibble: 5 x 2  
##   continent mean_gdpPercap  
##   <fct>         <dbl>  
## 1 Africa          2194.  
## 2 Americas        7136.  
## 3 Asia            7902.  
## 4 Europe        14469.  
## 5 Oceania       18622.
```

# Funciones útiles para combinar con summarise():



## rbase

funciones	descripción
min(), max()	valores mínimos y máximos
mean()	media
median()	mediana
sum()	suma de los valores
var(), sd()	varianza y desviación típica

## dplyr

dplyr	descripción
first()	primer valor de un vector
last()	último valor de un vector
n()	el numero de valores en un vector
n_distinct()	número de valores distintos en un vector
nth()	extraer el valor que ocupa la posición n en un vector

# arrange()



- Podemos reordenar los datos según otro criterio, por ejemplo, en vez ordenar por países, reordenarlos según año.

```
gapminder %>%  
  arrange(year, country)
```

```
## # 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 Albania      Europe      1952   55.2  1282697   1601.  
## 3 Algeria      Africa      1952   43.1  9279525   2449.  
## 4 Angola       Africa      1952   30.0  4232095   3521.  
## 5 Argentina    Americas    1952   62.5 17876956   5911.  
## 6 Australia    Oceania     1952   69.1  8691212  10040.  
## 7 Austria      Europe      1952   66.8  6927772   6137.  
## 8 Bahrain      Asia        1952   50.9  120447    9867.  
## 9 Bangladesh   Asia        1952   37.5 46886859    684.  
## 10 Belgium     Europe      1952    68  8730405   8343.  
## # ... with 1,694 more rows
```



# arrange()



- Podemos reordenarlos de manera descendente

```
gapminder %>%  
  filter(year == 2007) %>%  
  arrange(desc(lifeExp))
```

```
## # A tibble: 142 x 6  
##   country      continent  year lifeExp      pop gdpPercap  
##   <fct>        <fct>    <int>   <dbl>    <int>    <dbl>  
## 1 Japan        Asia      2007    82.6 127467972  31656.  
## 2 Hong Kong, China Asia      2007    82.2   6980412   39725.  
## 3 Iceland      Europe    2007    81.8   301931    36181.  
## 4 Switzerland Europe    2007    81.7   7554661   37506.  
## 5 Australia    Oceania   2007    81.2  20434176  34435.  
## 6 Spain        Europe    2007    80.9  40448191  28821.  
## 7 Sweden       Europe    2007    80.9   9031088   33860.  
## 8 Israel       Asia      2007    80.7   6426679   25523.  
## 9 France       Europe    2007    80.7  61083916  30470.  
## 10 Canada      Americas  2007    80.7  33390141  36319.  
## # ... with 132 more rows
```



# Si quisiéramos encontrar los top 5 valores más altos o más bajos

```
gapminder %>%  
  top_n(5, lifeExp) %>%  
  head
```

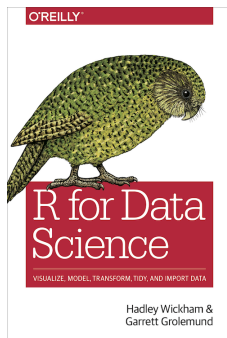
```
## # A tibble: 5 x 6  
##   country      continent  year lifeExp      pop gdpPercap  
##   <fct>         <fct>    <int>   <dbl>    <int>    <dbl>  
## 1 Hong Kong, China Asia      2007   82.2    6980412   39725.  
## 2 Iceland       Europe    2007   81.8     301931   36181.  
## 3 Japan         Asia      2002    82    127065841  28605.  
## 4 Japan         Asia      2007   82.6   127467972  31656.  
## 5 Switzerland  Europe    2007   81.7    7554661   37506.
```

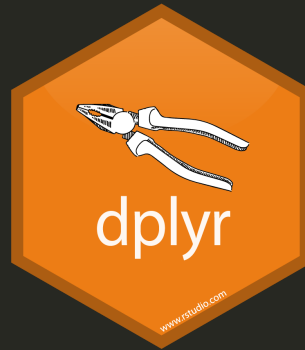
# Si tenemos dudas



Podemos consultar la documentación

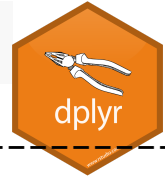
```
?dplyr::select  
?dplyr::filter  
?dplyr::mutate  
?dplyr::arrange  
?dplyr::summarise  
?dplyr::group_by
```





Muchas Gracias!!

```
devtools::session_info()
```



```
## - Session info -----
##   setting  value
##   version  R version 3.6.1 (2019-07-05)
##   os       Windows 10 x64
##   system   x86_64, mingw32
##   ui       RTerm
##   language (EN)
##   collate  Spanish_Argentina.1252
##   ctype    Spanish_Argentina.1252
##   tz       America/Buenos_Aires
##   date     2019-08-24
```

```
## - Packages -----
##   package      * version date      lib source
##   assertthat    0.2.1   2019-03-21 [1] CRAN (R 3.6.1)
##   backports     1.1.4   2019-04-10 [1] CRAN (R 3.6.0)
##   callr         3.3.1   2019-07-18 [1] CRAN (R 3.6.1)
##   cli           1.1.0   2019-03-19 [1] CRAN (R 3.6.1)
##   colorspace    1.4-1   2019-03-18 [1] CRAN (R 3.6.1)
##   crayon        1.3.4   2017-09-16 [1] CRAN (R 3.6.1)
##   desc          1.2.0   2018-05-01 [1] CRAN (R 3.6.1)
##   devtools      2.1.0   2019-07-06 [1] CRAN (R 3.6.1)
##   digest        0.6.20  2019-07-04 [1] CRAN (R 3.6.1)
##   dplyr         * 0.8.3   2019-07-04 [1] CRAN (R 3.6.1)
##   evaluate      0.14    2019-05-28 [1] CRAN (R 3.6.1)
```

