Rladies

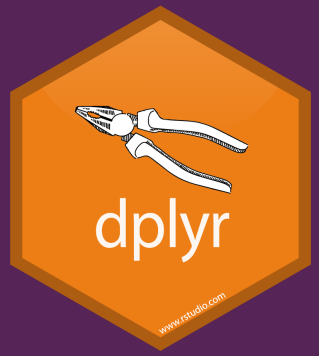
Resistencia Corrientes

Tutorial de dplyr

22 de Agosto de 2019

Realizado con Rmarkdown, xaringan y knitr

Bienvenidas!!



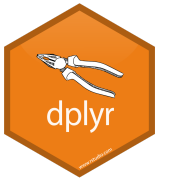
¿Qué vamos a ver hoy?

Funciones útiles

Funciones Principales + Tips

Split - Apply - Combine + Joins

Nuestro material

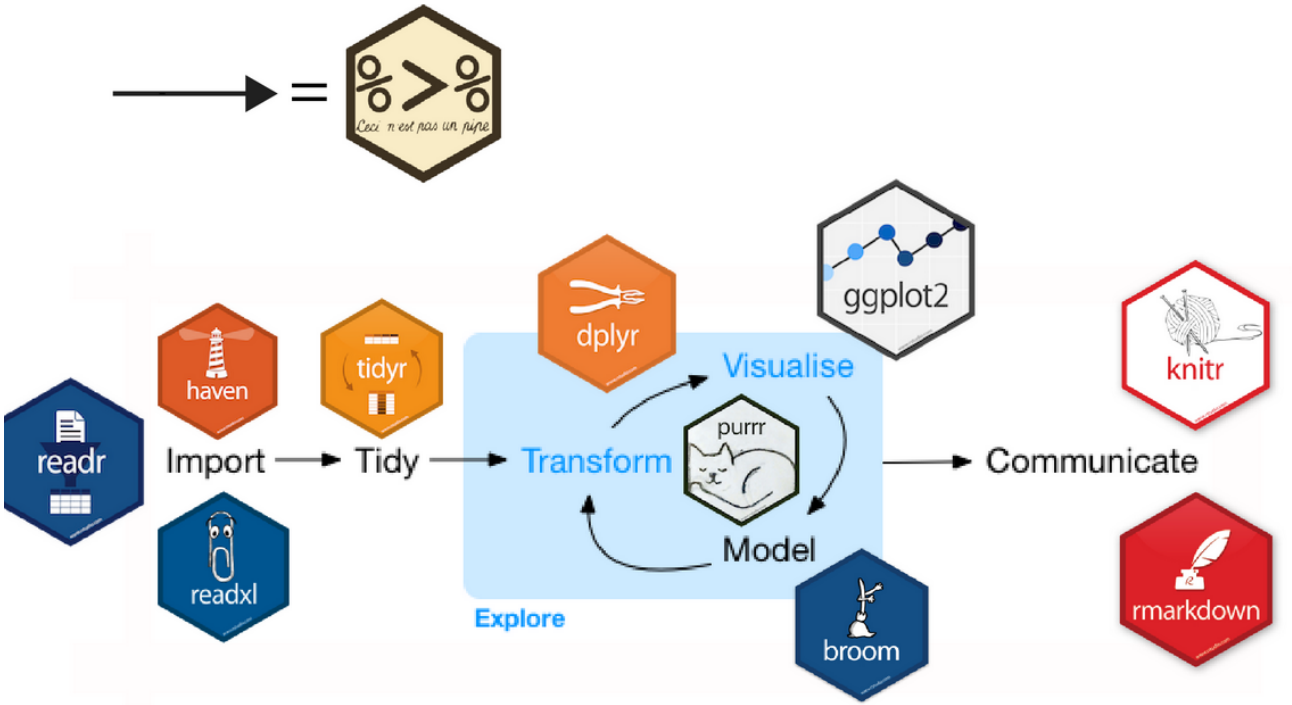


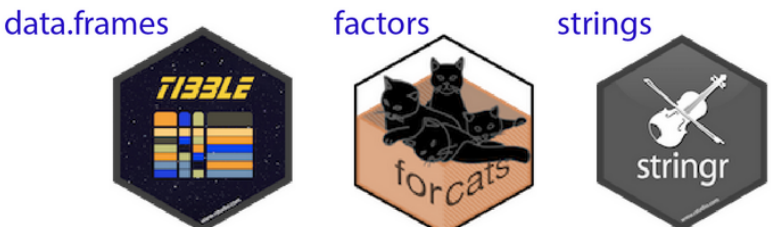
https://github.com/RLadiesResistenciaCorrientes/2019-dplyr-tips-tricks

Vignette de dplyr

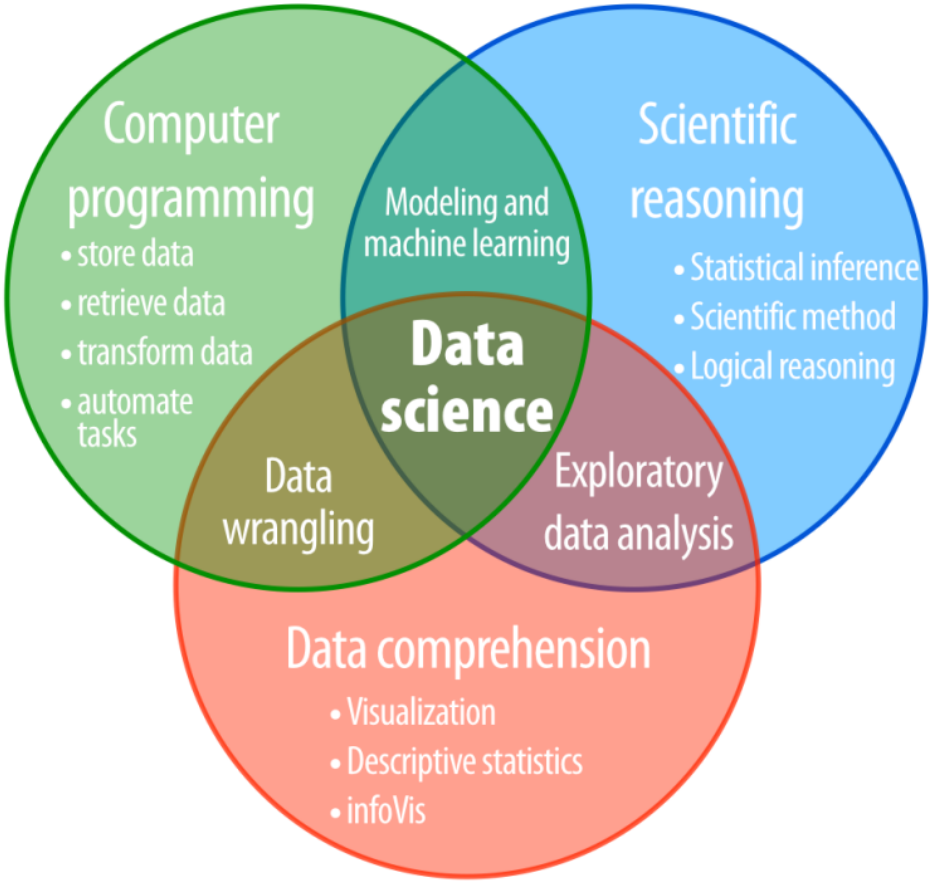
https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html

Flujo de trabajo para Ciencia de Datos



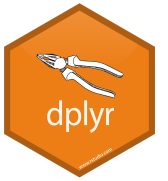


Ciencia de datos



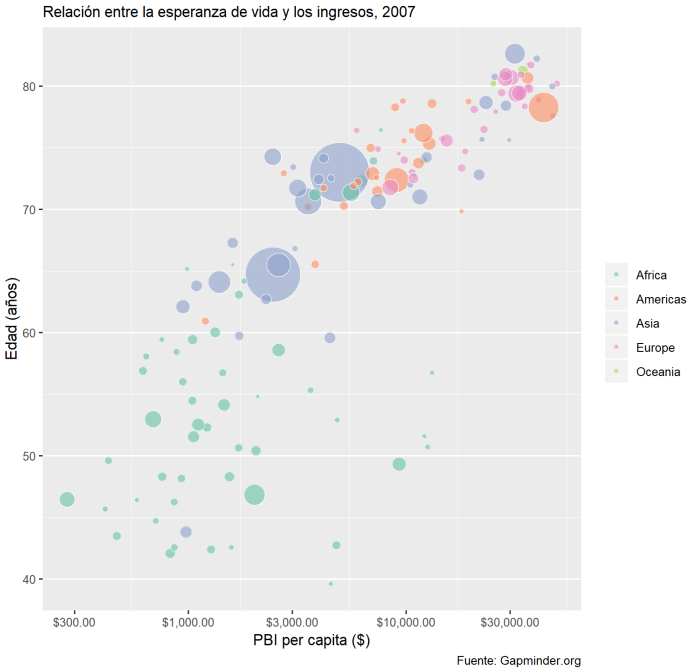
[*] *Hands on programming with R* by Garret Grolemund

Dataset: gapminder

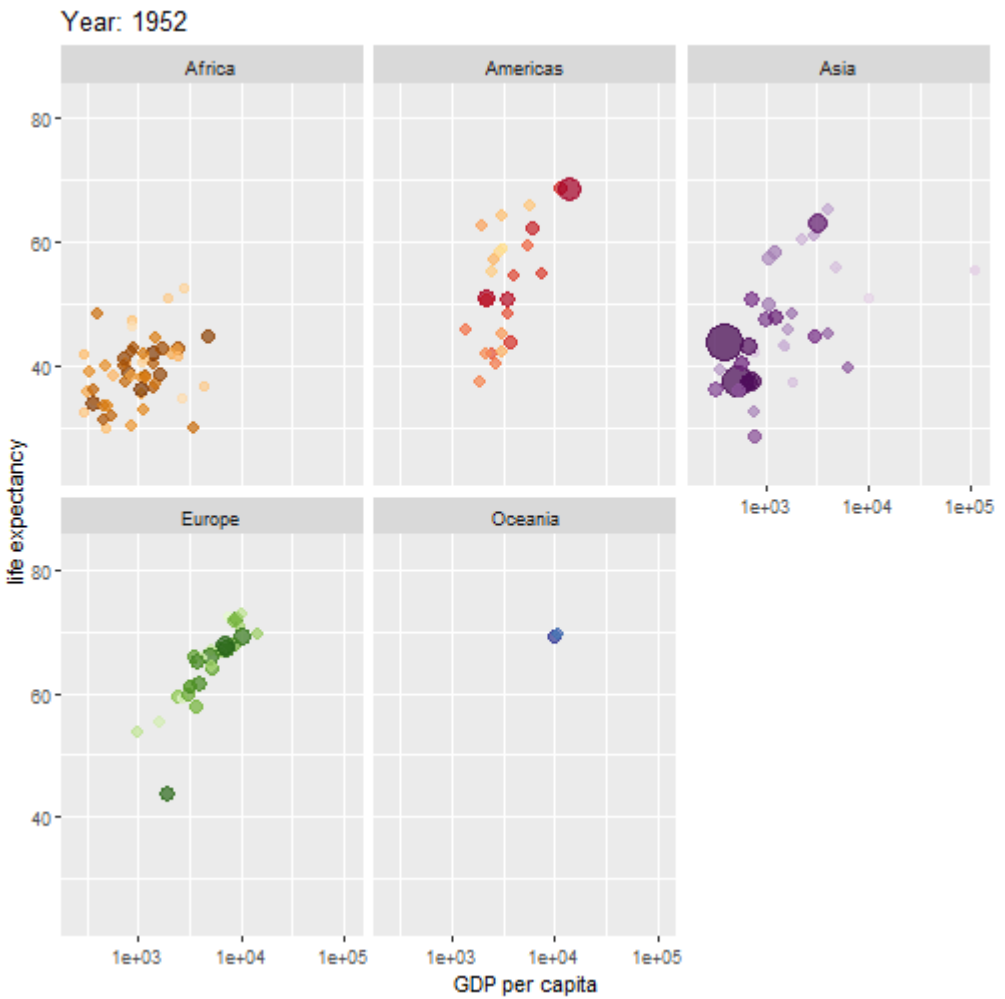



```
install.packages("gapminder")
install.packages("dplyr")
```

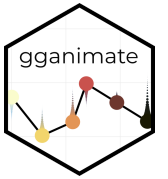
```
library(gapminder)
library(dplyr)
library(ggplot2)
ggplot2::ggplot(filter(gapminder, year
  scale_x_log10(labels = scales::dolla
  geom_point(aes(size = pop, fill = co
  scale_fill_brewer(palette = "Set2") +
  scale_size_continuous(range = c(1, 2
  labs(title = "",
       subtitle = "Relación entre la e
       caption = "Fuente: Gapminder.or
       x = "PBI per capita ($)",
       y = "Edad (años)") +
  guides(size = FALSE) +
  theme(panel.grid.major.x = element_b
        legend.position = "right",
        legend.title = element_blank()
```



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 expectancy') +
  transition_time(year)
```

data.frame vs. tibble

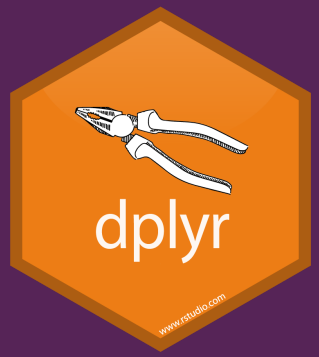
- data.frame es una estructura nativas de rbase y está disponible sin necesidad de instalar ningún paquete adicional. Es una estructura tabular organizada en filas y columnas. Se puede acceder a las columnas mediantes colnames() y los nombres de las filas, como rownames().
- tibble, sin embargo, es parte 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()
```

- Slicing

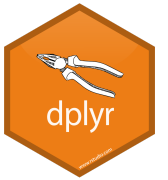
```
iris$Sepal.Length
iris[["Sepal.Length"]]
```

```
iris %>% .$Sepal.Length
iris %>% .[["Sepal.Length"]]
```



Funciones Útiles

Funciones Útiles



Cargamos el paquete dplyr

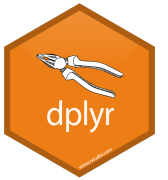
```
library(dplyr)
library(gapminder)
```

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

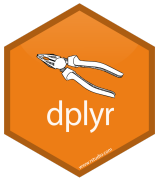


tail()

```
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

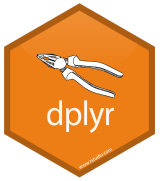


glimpse()

```
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.
```

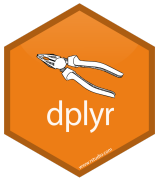
Funciones Útiles

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 12881816 13867957
##  $ 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.

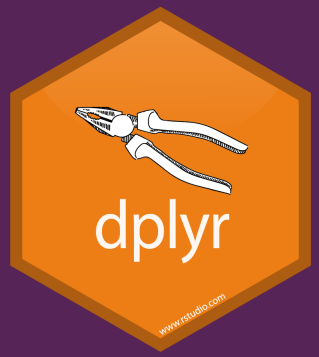
```
library(magrittr)
gapminder %>%
  filter(country=="Argentina")%>%
  select(gdpPercap, year) %>%
  head(2)
```

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

- Sin operador pipe

```
head(select(filter(gapminder, country=="Argentina"), gdpPercap, year), 2)
```

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



Funciones principales de dplyr

select(): extraer columnas

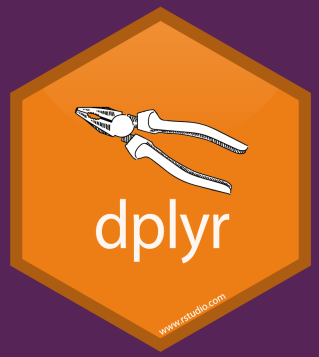
filter(): extraer filas siguiendo una restricción lógica

mutate(): crea nuevas variables

summarise(): cambiar la unidad de análisis

arrange(): ordenar filas por variables

group_by(): agrupar según un determinado criterio



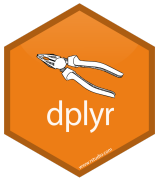
Otras funcionalidades de dplyr

dplyr_all: aplica una operación a todas las variables

dplyr_if: aplica una operación a un set de columnas o filas que fueron seleccionadas según sus características

dplyr_at: aplica una operación a un set de columnas o filas basados en los nombres de las variables

select()



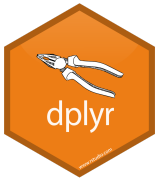
- Podemos extraer columnas

```
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
```

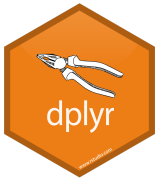
[*] Mas información en https://dplyr.tidyverse.org/reference/select.html

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

Tip 1: si tenemos que seleccionar las mismas columnas varias veces

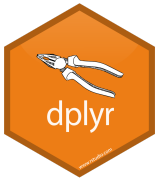



```
cols<-c("country", "lifeExp", "gdpPercap")

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
```

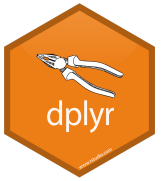
Tip 2: seleccionar según una expresión regular (regex)




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

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

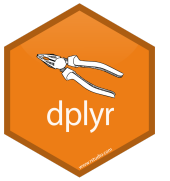
Tip 3: para reordenar columnas




```
gapminder %>%
  select("lifeExp", "gdpPercap", everything())%>%
  head
```

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

select_all()



- Permite seleccionar todas las columnas y aplicar una operación a todas las columnas

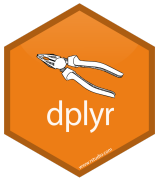
```
gapminder %>%
  select_all(toupper) %>%
  head
```

```
## # 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.
```

- Para deshacer el cambio anterior

```
gapminder %>%
    select_all(tolower)
```

Tip 4: renombrar las variables de una sola vez



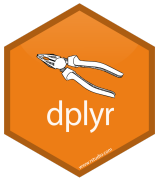

```
iris %>%
  head(1)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
```

```
library(stringr)
iris %>%
  select_all(tolower) %>%
  rename_all(~str_replace_all(., "\\.", "_")) %>%
  head(2)
```

```
##   sepal_length sepal_width petal_length petal_width species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
```

filter()



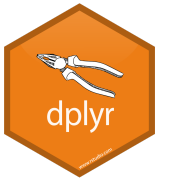
- Realizamos el filtrado según un valor numérico de una variable. Para ello utilizamos los siguientes operadores: >, >=, <, =<, ==, !=

```
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.
```

[*] Mas información en https://dplyr.tidyverse.org/reference/filter.html

filter()



Podemos filtrar según múltiples condiciones:

- Se cumplen ambas condiciones

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

- Se cumple la condición 1 solamente

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

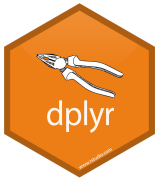
- Se cumplen la condición 1 y/ o la condición 2

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

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

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

filter()

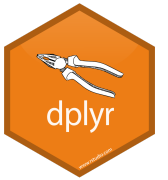


- Si queremos realizar un filtrado según un rango, lo hacemos de la siguiente manera

```
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
```

Tip 5: usar between() para especificar rangos

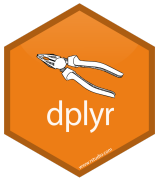


- También podemos hacerlo combinando con between():

```
gapminder %>%
  select (country, lifeExp, year) %>%
  filter(between(lifeExp, 60, 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()



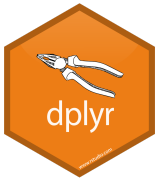
- mutate() es una función que nos permite crear una nueva columna en un tibble, realizando una operación con otras ya existentes.

```
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.
```

[*] Más información en https://dplyr.tidyverse.org/reference/mutate.html

Tip 6: 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.
```

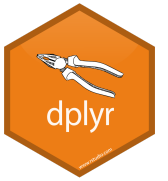
summarise()


```
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.
```

[*] Más información en https://dplyr.tidyverse.org/reference/summarise.html

Funciones útiles para combinar con summarise():



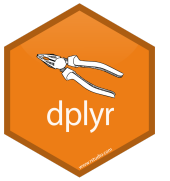
rbase

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

summarise_all()

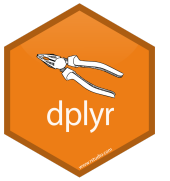


- Requiere una función que se aplicará a todas las columnas

```
iris %>%
  group_by(Species) %>%
  summarise_all(mean)%>%
  head
```

```
## # A tibble: 3 x 5
##   Species    Sepal.Length Sepal.Width Petal.Length Petal.Width
##   <fct>             <dbl>       <dbl>        <dbl>       <dbl>
## 1 setosa             5.01        3.43         1.46       0.246
## 2 versicolor         5.94        2.77         4.26       1.33
## 3 virginica          6.59        2.97         5.55       2.03
```

summarise_at()

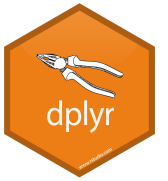


- Requiere dos argumentos, uno indicando las columnas que se tendrán en cuenta, y luego la operación con la que se resumirán los datos.

```
iris %>%
  group_by(Species) %>%
  summarise_at(vars(contains("Sepal")), mean)
```

```
## # A tibble: 3 x 3
##   Species    Sepal.Length Sepal.Width
##   <fct>             <dbl>       <dbl>
## 1 setosa             5.01        3.43
## 2 versicolor         5.94        2.77
## 3 virginica          6.59        2.97
```

summarise_if():

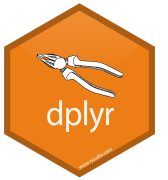


- Requiere dos argumentos

```
gapminder %>%
  group_by(continent) %>%
  summarise_if(is.numeric, mean, na.rm=TRUE)
```

```
## # A tibble: 5 x 5
##   continent  year lifeExp       pop gdpPercap
##   <fct>     <dbl>   <dbl>     <dbl>     <dbl>
## 1 Africa    1980.    48.9  9916003.     2194.
## 2 Americas  1980.    64.7 24504795.     7136.
## 3 Asia      1980.    60.1 77038722.     7902.
## 4 Europe    1980.    71.9 17169765.    14469.
## 5 Oceania   1980.    74.3  8874672.    18622.
```

arrange()



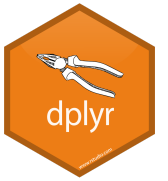
- Podemos reordenar los datos según otro criterio, por ejemplo, en vez ordenar por países, reordernarlos según año.

```
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
```

[*] Más información en https://dplyr.tidyverse.org/reference/arrange.html

arrange()

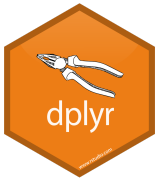


- Podemos reordenarlos de manera descendente

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

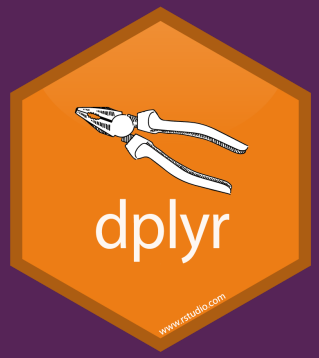
```
## # A tibble: 142 x 6
##    country          continent  year lifeExp       pop gdpPercap
##    <fct>            <fct>     <int>   <dbl>     <int>     <dbl>
##  1 Japan            Asia       2007    82.6 127467972    31656.
##  2 Hong Kong, China Asia       2007    82.2   6980412    39725.
##  3 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
```

Tip 7: Encontrar los top 5 valores más altos o más bajos




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

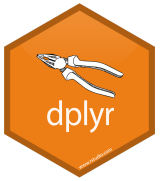
```
## # A tibble: 5 x 6
##   country          continent  year lifeExp       pop gdpPercap
##   <fct>            <fct>     <int>   <dbl>     <int>     <dbl>
## 1 Hong Kong, China Asia       2007    82.2   6980412    39725.
## 2 Iceland          Europe     2007    81.8    301931    36181.
## 3 Japan 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



Split - Apply - Combine

con dplyr

Split - Apply - Combine con dplyr



- Definimos la función

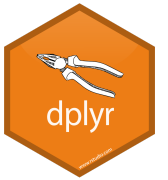
```
fun <- function(slice, keys) {
broom::tidy(lm(Petal.Length ~ Sepal.Length, data = slice))
}
```

- Aplicamos split - apply - combine

```
iris %>%
  group_by(Species) %>%
  group_modify(fun)
```

```
## # A tibble: 6 x 6
## # Groups:   Species [3]
##   Species    term         estimate std.error statistic  p.value
##   <fct>      <chr>           <dbl>     <dbl>     <dbl>    <dbl>
## 1 setosa     (Intercept)     0.803    0.344      2.34  2.38e- 2
## 2 setosa     Sepal.Length    0.132    0.0685     1.92  6.07e- 2
## 3 versicolor (Intercept)     0.185    0.514      0.360 7.20e- 1
## 4 versicolor Sepal.Length    0.686    0.0863     7.95  2.59e-10
## 5 virginica  (Intercept)     0.610    0.417      1.46  1.50e- 1
## 6 virginica  Sepal.Length    0.750    0.0630    11.9   6.30e-16
```

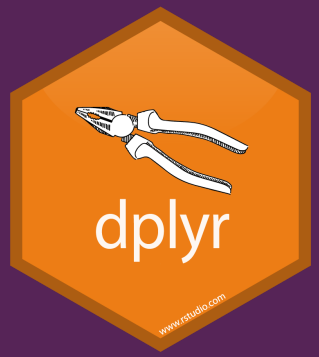
Tambien podemos hacerlo con una función anónima



- Una función anónima es una función que se define al momento que la estamos usando, y generalmente no tiene un nombre. En Python estas funciones se llaman funciones lambda.

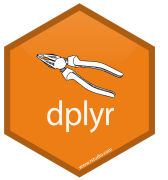
```
iris %>%
  group_by(Species) %>%
  group_modify(~ broom::tidy(lm(Petal.Length ~ Sepal.Length, data = .x))
)
```

```
## # A tibble: 6 x 6
## # Groups:   Species [3]
##   Species    term         estimate std.error statistic  p.value
##   <fct>      <chr>           <dbl>     <dbl>     <dbl>    <dbl>
## 1 setosa     (Intercept)     0.803    0.344      2.34  2.38e- 2
## 2 setosa     Sepal.Length    0.132    0.0685     1.92  6.07e- 2
## 3 versicolor (Intercept)     0.185    0.514      0.360 7.20e- 1
## 4 versicolor Sepal.Length    0.686    0.0863     7.95  2.59e-10
## 5 virginica  (Intercept)     0.610    0.417      1.46  1.50e- 1
## 6 virginica  Sepal.Length    0.750    0.0630    11.9   6.30e-16
```



Joins

Joins



- Cuando trabajamos con más de un dataset y necesitamos crear una base de datos a partir de dos o más set de datos, necesitamos realizar una operación llamada **JOIN**
- Existen distintos tipos de **JOIN** según que dataset querramos generar, y, también de acuerdo a si se crean columnas nuevas (**mutating joins**) o solo filtramos filas (**filtering joins**)

Mutating Joins

`left_join()` retains all cases in **left** data set

`right_join()` retains all cases in **right** data set

`full_join()` retains all cases in **either** data set

`inner_join()` retains all cases in **both** data set

[*] Mas información en https://dplyr.tidyverse.org/reference/join.html

Filtering Joins

`semi_join()` extracts cases that **have a match**

`anti_join()` extracts cases that **do not have a match**

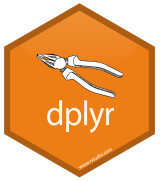
```
dplyr::band_members
```

```
## # A tibble: 3 x 2
##   name  band
##   <chr> <chr>
## 1 Mick  Stones
## 2 John  Beatles
## 3 Paul  Beatles
```

```
dplyr::band_instruments
```

```
## # A tibble: 3 x 2
##   name  plays
##   <chr> <chr>
## 1 John  guitar
## 2 Paul  bass
## 3 Keith guitar
```

Mutating Joins



Left Join

```
band_members %>% left_join(band_instruments)
```

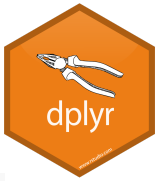
```
## # A tibble: 3 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 Mick  Stones  <NA>
## 2 John  Beatles guitar
## 3 Paul  Beatles bass
```

Right Join

```
band_members %>% right_join(band_instruments)
```

```
## # A tibble: 3 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 John  Beatles guitar
## 2 Paul  Beatles bass
## 3 Keith <NA>    guitar
```

Full Join




```
band_members %>% full_join(band_instruments)
```

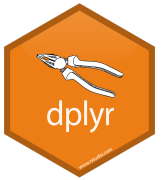
```
## # A tibble: 4 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 Mick  Stones  <NA>
## 2 John  Beatles guitar
## 3 Paul  Beatles bass
## 4 Keith <NA>    guitar
```

Inner Join

```
band_members %>% full_join(band_instruments)
```

```
## # A tibble: 4 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 Mick  Stones  <NA>
## 2 John  Beatles guitar
## 3 Paul  Beatles bass
## 4 Keith <NA>    guitar
```

Filtering Joins



Semi Join

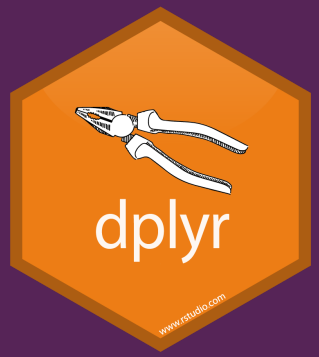
```
band_members %>% semi_join(band_instruments)
```

```
## # A tibble: 2 x 2
##   name  band
##   <chr> <chr>
## 1 John  Beatles
## 2 Paul  Beatles
```

Anti Join

```
band_members %>% anti_join(band_instruments)
```

```
## # A tibble: 1 x 2
##   name  band
##   <chr> <chr>
## 1 Mick  Stones
```



¿Qué vimos?

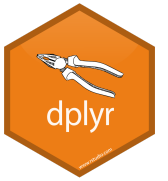
Funciones básicas de dplyr

Trucos de dplyr

Split - Apply - Combine con dplyr

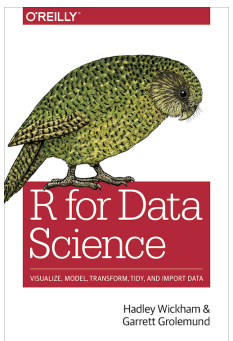
Joins

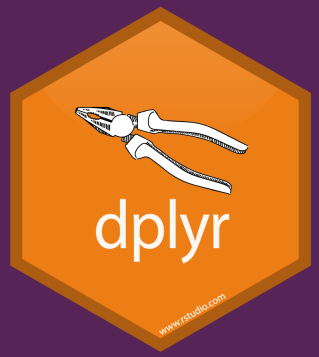
Si tenemos dudas



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     2019-08-22
##
## - 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)
##  broom          0.5.2   2019-04-07 [1] CRAN (R 3.6.1)
##  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)
##  fansi          0.4.0   2018-10-05 [1] CRAN (R 3.6.1)
##  farver         1.1.0   2018-11-20 [1] CRAN (R 3.6.1)
##  fs             1.3.1   2019-05-06 [1] CRAN (R 3.6.1)
##  gapminder    * 0.3.0   2017-10-31 [1] CRAN (R 3.6.1)
##  generics       0.0.2   2018-11-29 [1] CRAN (R 3.6.1)
##  gganimate    * 1.0.3   2019-04-02 [1] CRAN (R 3.6.1)
```