

# para análisis de datos

## Tips de dplyr

Roxana N. Villafaña | LEMyP |  @data\_datum  
Florencia D'Andrea | INTA-CONICET |  @cantoflor\_87



Página web del curso en [https://flor14.github.io/Curso\\_r\\_unne\\_2020/](https://flor14.github.io/Curso_r_unne_2020/) 

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



```
library(dplyr)
library(gapminder)
cols<-c("country", "lifeExp", "gdpPercap") #selecciono variables
gapminder %>%
  select(!cols) #selecciono según el vector creado
```

```
## # 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"))%>% #la expresión regular es "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
```

# Tip 4: Si quisiera borrar una columna



Con la función `select` y como argumento el nombre de la columna, antecedida por el signo menos.

```
gapminder %>%  
  select(-pop)
```

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



# 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 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
```

# 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.
```



# Funciones útiles para combinar con `summarise()`:



## rbase

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

## dplyr

dplyr	descripción
<code>first()</code>	primer valor de un vector
<code>last()</code>	último valor de un vector
<code>n()</code>	el numero de valores en un vector
<code>n_distinct()</code>	número de valores distintos en un vector
<code>nth()</code>	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
```

```
#resumo variables que contengan #Sepal
```



# summarise\_if():

- Requiere dos argumentos

```
gapminder %>%  
  group_by(continent) %>%  
  summarise_if(is.numeric, mean)
```

```
## # 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.
```

# Si tenemos dudas



Podemos consultar la documentación

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

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

```
## - Session info -----  
## setting value  
## version R version 3.6.2 (2019-12-12)  
## 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 2020-02-18  
##  
## - Packages -----  
## package * version date lib source  
## assertthat 0.2.1 2019-03-21 [1] CRAN (R 3.6.1)  
## backports 1.1.5 2019-10-02 [1] CRAN (R 3.6.1)  
## callr 3.4.1 2020-01-24 [1] CRAN (R 3.6.2)  
## cli 2.0.1 2020-01-08 [1] CRAN (R 3.6.2)  
## 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 2.2.1 2019-09-24 [1] CRAN (R 3.6.2)  
## digest 0.6.23 2019-11-23 [1] CRAN (R 3.6.1)  
## dplyr * 0.8.3 2019-07-04 [1] CRAN (R 3.6.1)  
## ellipsis 0.3.0 2019-09-20 [1] CRAN (R 3.6.1)  
## emo * 0.0.0.9000 2019-12-07 [1] Github (hadley/emo@02a5206)
```