

# Análisis de Datos

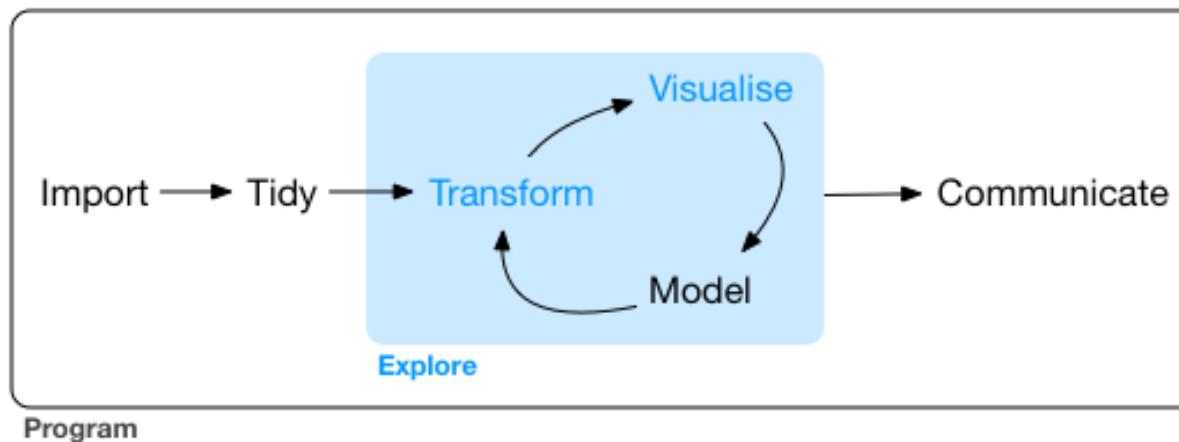
## Tema 2 - Análisis Exploratorio de los Datos

### 2.2 Transformación

Roi Naveiro

# Análisis Exploratorio de los Datos

- El arte de observar los datos, generar hipótesis y testearlas.
- **Objetivo:** generar preguntas prometedoras para, posteriormente, explorarlas en mayor profundidad



# Transformación de datos

# Transformación de datos

La visualización es clave para un buen entendimiento de los datos...

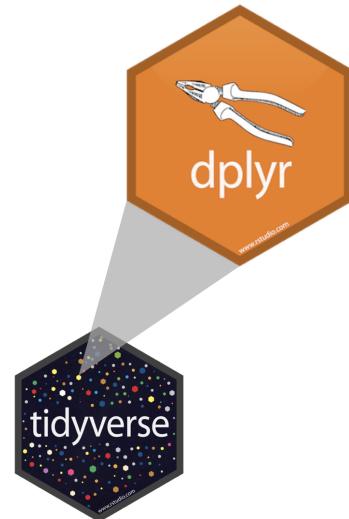
... no obstante, rara vez encontramos los datos en el formato necesario.

En general, necesitamos

- Crear nuevas variables
- Resumir variables
- Renombrar
- Reordenar observaciones
- Agrupar
- ...

# Gramática de transformación de datos

Aprenderemos cómo hacer esto de forma **reproducible** con el paquete **dplyr**



Similar a **ggplot2**, **dplyr** introduce una gramática para la transformación de datos

# Gramática de transformación de datos

Algunas de las funciones (verbos) que aprenderemos

- **select**: seleccionar columnas por nombre
- **rename**: renombrar variables
- **arrange**: ordenar filas según criterio
- **slice**: escoger filas con índices
- **filter**: escoger filas que cumplan cierta condición
- **distinct**: filtrar por filas únicas
- **mutate**: añadir nuevas variables
- **summarise**: resumir variables según ciertos estadísticos
- **group\_by**: para realizar operaciones por grupos
- ...

# Reglas de dplyr

Todos los verbos funcionan de manera similar:

- Primer argumento es el data frame
- Los siguientes argumentos especifican qué hacer con el data frame, usando los nombres de las variables (sin comillas)
- La salida es otro data frame

Los verbos pueden concatenarse

# Datos

Para aprender las diferentes transformaciones que podemos hacer sobre los datos, usaremos de nuevo **gapminder** (más información en [? gapminder](#))

```
library(tidyverse)
library(gapminder)
glimpse(gapminder)

## #> #> Rows: 1,704
## #> #> Columns: 6
## #> #> $ country    <fct> "Afghanistan", "Afghanistan", "Afghanist...
## #> #> $ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia...
## #> #> $ year       <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982...
## #> #> $ lifeExp    <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, ...
## #> #> $ pop        <int> 8425333, 9240934, 10267083, 11537966, 13...
## #> #> $ gdpPercap   <dbl> 779.4453, 820.8530, 853.1007, 836.1971, ...
```

# Transformación de datos: **select**

# Verbo **select**

- En múltiples ocasiones tenemos conjuntos de datos con miles de variables
- Nos puede interesar un subconjunto de las mismas
- Para esto utilizamos **select**

# Seleccionar una columna

Observar únicamente la variable referente a esperanza de vida (**lifeExp**)

```
gapminder %>%  
  select(lifeExp)
```

```
## # A tibble: 1,704 × 1  
##   lifeExp  
##     <dbl>  
## 1    28.8  
## 2    30.3  
## 3    32.0  
## 4    34.0  
## 5    36.1  
## 6    38.4  
## 7    39.9  
## 8    40.8  
## 9    41.7  
## 10   41.8  
## # ... with 1,694 more rows
```

- Pasamos el data frame a través del pipe (`%>%`) a la función **select()**
- El argumento es nombre de variable (sin comillas): **lifeExp**
- Esto de lugar a un dataframe de 1704 filas y 1 columna

# Seleccionar múltiples columnas

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

```
## # A tibble: 1,704 × 2  
##   lifeExp country  
##   <dbl> <fct>  
## 1 28.8 Afghanistan  
## 2 30.3 Afghanistan  
## 3 32.0 Afghanistan  
## 4 34.0 Afghanistan  
## 5 36.1 Afghanistan  
## 6 38.4 Afghanistan  
## 7 39.9 Afghanistan  
## 8 40.8 Afghanistan  
## 9 41.7 Afghanistan  
## 10 41.8 Afghanistan  
## # ... with 1,694 more rows
```

```
gapminder %>%  
  select(lifeExp:gdpPercap)
```

```
## # A tibble: 1,704 × 3  
##   lifeExp     pop gdpPercap  
##   <dbl>     <int>    <dbl>  
## 1 28.8 8425333 779.  
## 2 30.3 9240934 821.  
## 3 32.0 10267083 853.  
## 4 34.0 11537966 836.  
## 5 36.1 13079460 740.  
## 6 38.4 14880372 786.  
## 7 39.9 12881816 978.  
## 8 40.8 13867957 852.  
## 9 41.7 16317921 649.  
## 10 41.8 22227415 635.  
## # ... with 1,694 more rows
```

**NOTA:** los verbos no modifican el data frames. Para guardar el output como un nuevo dataframe

```
esperanza_vida <- gapminder %>%  
  select(lifeExp)
```

# Selección inversa

```
gapminder %>%  
  select(-lifeExp, -pop, -year )
```

```
## # A tibble: 1,704 × 3  
##   country     continent gdpPercap  
##   <fct>       <fct>      <dbl>  
## 1 Afghanistan Asia        779.  
## 2 Afghanistan Asia        821.  
## 3 Afghanistan Asia        853.  
## 4 Afghanistan Asia        836.  
## 5 Afghanistan Asia        740.  
## 6 Afghanistan Asia        786.  
## 7 Afghanistan Asia        978.  
## 8 Afghanistan Asia        852.  
## 9 Afghanistan Asia        649.  
## 10 Afghanistan Asia       635.  
## # ... with 1,694 more rows
```

```
gapminder %>%  
  select(-(continent:gdpPercap) )
```

```
## # A tibble: 1,704 × 1  
##   country  
##   <fct>  
## 1 Afghanistan  
## 2 Afghanistan  
## 3 Afghanistan  
## 4 Afghanistan  
## 5 Afghanistan  
## 6 Afghanistan  
## 7 Afghanistan  
## 8 Afghanistan  
## 9 Afghanistan  
## 10 Afghanistan  
## # ... with 1,694 more rows
```

# Otras formas de selección

Las siguientes funciones son útiles

- **starts\_with("abc")**: nombres que empiezan con "abc"
- **ends\_with("abc")**: nombres que terminan con "abc"
- **contains("abc")**: nombres que contienen "abc"
- **matches("(.)//1")**: selecciona según expresiones regulares
- **num\_range("x", 1:3)**: selecciona x1, x2 y x3

# Ejemplo

```
gapminder %>%  
  select( starts_with("co"))
```

```
## # A tibble: 1,704 × 2  
##   country     continent  
##   <fct>       <fct>  
## 1 Afghanistan Asia  
## 2 Afghanistan Asia  
## 3 Afghanistan Asia  
## 4 Afghanistan Asia  
## 5 Afghanistan Asia  
## 6 Afghanistan Asia  
## 7 Afghanistan Asia  
## 8 Afghanistan Asia  
## 9 Afghanistan Asia  
## 10 Afghanistan Asia  
## # ... with 1,694 more rows
```

# Seleccionar todas

**everything()** es útil para colocar ciertas variables al inicio del data frame

```
gapminder %>%  
  select( year, everything() )
```

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

# Transformación de datos: `rename`

# Renombrar

Para renombrar variables, utilizar **rename**

```
gapminder %>%  
  rename( pais = country )
```

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

# Transformación de datos: `arrange`

# Ordenar datos

Seleccionemos país, año y esperanza de vida y ordenemos en orden creciente de esperanza de vida

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

```
## # A tibble: 1,704 × 3
##   country      year  lifeExp
##   <fct>        <int>   <dbl>
## 1 Rwanda       1992    23.6
## 2 Afghanistan 1952    28.8
## 3 Gambia       1952     30
## 4 Angola        1952    30.0
## 5 Sierra Leone 1952    30.3
## 6 Afghanistan  1957    30.3
## 7 Cambodia      1977    31.2
## 8 Mozambique    1952    31.3
## 9 Sierra Leone  1957    31.6
## 10 Burkina Faso 1952   32.0
## # ... with 1,694 more rows
```

# Ordenar datos

Seleccionemos país, año y esperanza de vida y ordenemos en orden decreciente de esperanzada de vida

```
gapminder %>%
  select( country, year, lifeExp ) %>%
  arrange(desc(lifeExp))
```

```
## # A tibble: 1,704 × 3
##   country      year lifeExp
##   <fct>        <int>   <dbl>
## 1 Japan          2007    82.6
## 2 Hong Kong, China 2007    82.2
## 3 Japan          2002     82
## 4 Iceland         2007    81.8
## 5 Switzerland     2007    81.7
## 6 Hong Kong, China 2002    81.5
## 7 Australia        2007    81.2
## 8 Spain           2007    80.9
## 9 Sweden           2007    80.9
## 10 Israel          2007    80.7
## # ... with 1,694 more rows
```

# Ordenar datos

¿Qué sucede?

```
gapminder %>%
  select( country, year, lifeExp ) %>%
  arrange(desc(country))
```

```
## # A tibble: 1,704 × 3
##   country   year lifeExp
##   <fct>     <int>   <dbl>
## 1 Zimbabwe  1952    48.5
## 2 Zimbabwe  1957    50.5
## 3 Zimbabwe  1962    52.4
## 4 Zimbabwe  1967    54.0
## 5 Zimbabwe  1972    55.6
## 6 Zimbabwe  1977    57.7
## 7 Zimbabwe  1982    60.4
## 8 Zimbabwe  1987    62.4
## 9 Zimbabwe  1992    60.4
## 10 Zimbabwe 1997    46.8
## # ... with 1,694 more rows
```

# Ordenar datos

- **arrange** por defecto ordena de manera ascendente (si son números)
- Ante strings, **arrange** ordena por defecto de manera alfabética
- **desc()** cambia el orden
- Si se pasa más de una variable, se ordena en función de la primera. Las siguientes se utilizan, secuencialmente, para romper empates
- Los valores ausentes pasan al final

# Transformación de datos: pipes

# Los pipes

- Un pipe es una técnica para pasar información de un proceso a otro
- La estructura pipe está implementada en el paquete **magrittr** (que se carga con **tidyverse**)

## 1. Empezamos con todo el dataset

```
gapminder %>%
  select( country, year, lifeExp ) %>%
  arrange(desc(country))
```

# Los pipes

- Un pipe es una técnica para pasar información de un proceso a otro
- La estructura pipe está implementada en el paquete **magrittr** (que se carga con **tidyverse**)
  1. Este entra en **select**, que selecciona country, year, lifeExp

```
gapminder %>%
  select( country, year, lifeExp ) %>%
  arrange(desc(country))
```

# Los pipes

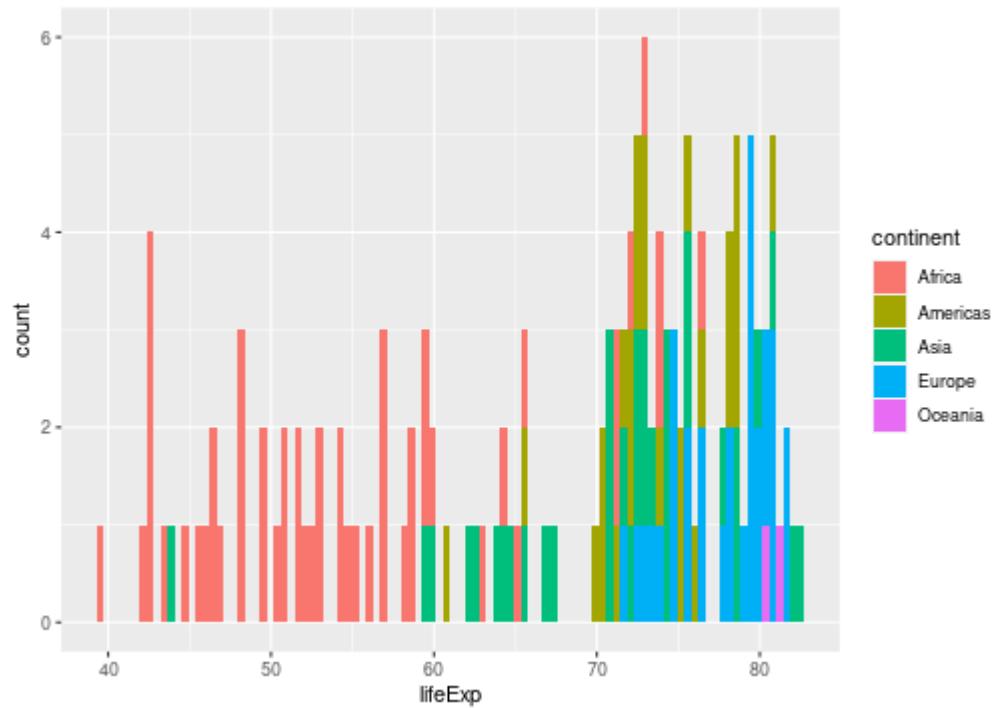
- Un pipe es una técnica para pasar información de un proceso a otro
- La estructura pipe está implementada en el paquete **magrittr** (que se carga con **tidyverse**)
  1. Por último, el resultado del select pasa a **arrange**

```
gapminder %>%
  select( country, year, lifeExp ) %>%
  arrange(desc(country))
```

# Los pipes

Pasando la respuesta a **ggplot2**

```
gapminder %>%
  filter(year == 2007) %>%
  ggplot(., aes(x = lifeExp, fill = continent)) +
  geom_histogram(bins = 100)
```



# Transformación de datos: *slice*

# Seleccionar filas

```
gapminder %>%
  select(country) %>%
  slice(1:3)
```

```
## # A tibble: 3 × 1
##   country
##   <fct>
## 1 Afghanistan
## 2 Afghanistan
## 3 Afghanistan
```

# Ejercicio 1

Usando **slice** selecciona las tres últimas filas del dataset **gapminder**

# Transformación de datos: `filter`

# Filtrado

- Seleccionar filas en función de sus valores
- Los argumentos reflejan las condiciones del filtrado

```
gapminder %>%
  filter(country == "Spain", lifeExp > 80)

## # A tibble: 1 × 6
##   country continent year lifeExp      pop gdpPercap
##   <fct>    <fct>    <int>    <dbl>    <int>    <dbl>
## 1 Spain     Europe     2007     80.9  40448191    28821.
```

# Filtrado

Para construir filtros complejos, hay que manejar los operadores lógicos

Operador	Definición	Operador	Definición
<	Menor que	$x   y$	$x \text{ OR } y$
$\leq$	Menor o igual que	<code>is.na(x)</code>	¿ $x$ es NA?
>	Mayor que	<code>!is.na(x)</code>	¿ $x$ no es NA?
$\geq$	Mayor o igual que	<code>x %in% y</code>	$x$ pertenece a $y$
$==$	Igual a	<code>!(x %in% y)</code>	$x$ no pertenece $y$
$!=$	Distinto a	<code>!x</code>	negación $x$
$x & y$	$x \text{ AND } y$		

# Filtrado

¿Qué está pasando?

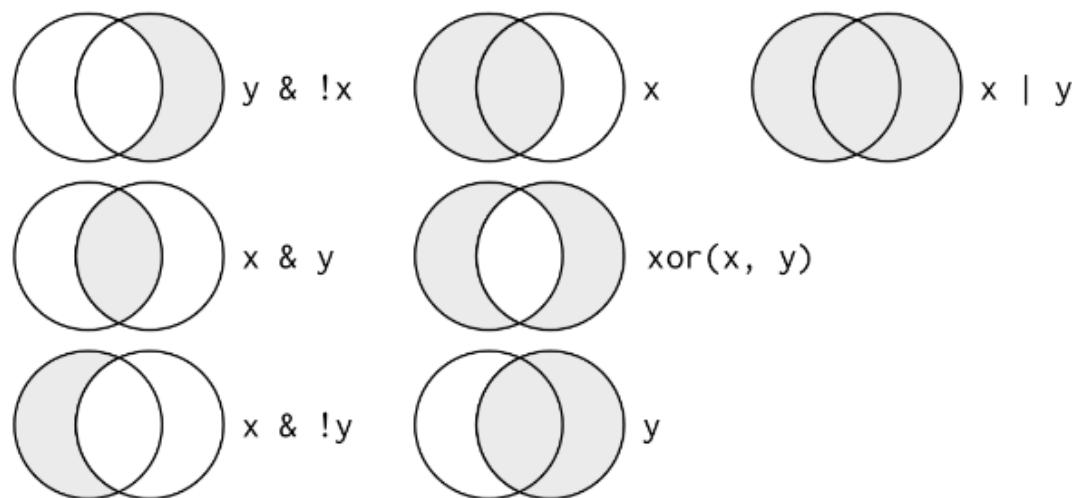
```
sqrt(2)^2 == 2
```

```
## [1] FALSE
```

```
near(sqrt(2)^2, 2)
```

```
## [1] TRUE
```

# Filtrado



# Filtrado

Por defecto, los múltiples argumentos de **filter** se combinan con el operador ?

Ejemplo: seleccionar países de América que en el 2007 tuvieran esperanza de vida superior a 75 o un PIB per cápita inferior a 6000

```
gapminder %>%
  filter(year == 2007, continent == "Americas", (lifeExp > 77 | gdpPercap <

## # A tibble: 10 × 6
##   country      continent year lifeExp      pop gdpPercap
##   <fct>        <fct>    <int>  <dbl>     <int>     <dbl>
## 1 Bolivia      Americas  2007    65.6    9119152     3822.
## 2 Canada       Americas  2007    80.7   33390141     36319.
## 3 Chile        Americas  2007    78.6   16284741     13172.
## 4 Costa Rica   Americas  2007    78.8   4133884      9645.
## 5 Cuba         Americas  2007    78.3   11416987     8948.
## 6 Haiti         Americas  2007    60.9   8502814      1202.
## 7 Honduras     Americas  2007    70.2   7483763      3548.
## 8 Nicaragua     Americas  2007    72.9   5675356      2749.
## 9 Puerto Rico   Americas  2007    78.7   3942491     19329.
## 10 United States Americas 2007    78.2  301139947     42952.
```

# Transformación de datos: valores ausentes

# Valores ausentes

- Muy comunes en ciencia de datos
- Casi todas las operaciones con NA devuelven NA

```
NA == 2
```

```
## [1] NA
```

```
NA + 3
```

```
## [1] NA
```

```
NA == NA
```

```
## [1] NA
```

# Valores ausentes

- En el filtrado, solo se incluyen filas con valores **TRUE** de la condición (se excluyen pues los **FALSE** y los **NA**).
- Es fácil preservarlos

```
df <- tibble(x = c(1, NA, 3))
filter(df, x > 1)
```

```
## # A tibble: 1 × 1
##       x
##   <dbl>
## 1     3
```

```
filter(df, is.na(x) | x > 1)
```

```
## # A tibble: 2 × 1
##       x
##   <dbl>
## 1     NA
## 2     3
```

# Transformación de datos: ***distinct***

# Valores únicos

Filtrar valores únicos de filas

```
gapminder %>%
  distinct(continent, year) %>%
  arrange(continent, desc(year))
```

```
## # A tibble: 60 × 2
##   continent    year
##   <fct>      <int>
## 1 Africa        2007
## 2 Africa        2002
## 3 Africa        1997
## 4 Africa        1992
## 5 Africa        1987
## 6 Africa        1982
## 7 Africa        1977
## 8 Africa        1972
## 9 Africa        1967
## 10 Africa       1962
## # ... with 50 more rows
```

# Transformación de datos: **count**

# Crear tablas de frecuencias

```
gapminder %>%  
  filter(year == 1967) %>%  
  count(continent)
```

```
## # A tibble: 5 × 2  
##   continent     n  
##   <fct>     <int>  
## 1 Africa       52  
## 2 Americas     25  
## 3 Asia         33  
## 4 Europe       30  
## 5 Oceania      2
```

```
gapminder %>%  
  filter(year == 1967) %>%  
  count(continent, sort=TRUE)
```

```
## # A tibble: 5 × 2  
##   continent     n  
##   <fct>     <int>  
## 1 Africa       52  
## 2 Asia         33  
## 3 Europe       30  
## 4 Americas     25  
## 5 Oceania      2
```

# Contar y ordenar

```
gapminder %>%
  filter(year == 1967) %>%
  count(continent) %>%
  arrange(n)
```

```
## # A tibble: 5 × 2
##   continent     n
##   <fct>     <int>
## 1 Oceania      2
## 2 Americas     25
## 3 Europe       30
## 4 Asia         33
## 5 Africa       52
```

# Contar más de una variable

```
gapminder %>%  
  count(continent, year)
```

```
## # A tibble: 60 × 3  
##   continent    year     n  
##   <fct>      <int> <int>  
## 1 Africa        1952     52  
## 2 Africa        1957     52  
## 3 Africa        1962     52  
## 4 Africa        1967     52  
## 5 Africa        1972     52  
## 6 Africa        1977     52  
## 7 Africa        1982     52  
## 8 Africa        1987     52  
## 9 Africa        1992     52  
## 10 Africa       1997     52  
## # ... with 50 more rows
```

# Transformación de datos: `mutate`

# Añadir nuevas columnas

```
gapminder %>%  
  mutate(gdp = gdpPercap * pop) %>%  
  filter(year == 2007) %>%  
  arrange( desc(gdp) )
```

```
## # A tibble: 142 × 7  
##   country continent year lifeExp   pop gdpPercap     gdp  
##   <fct>    <fct>    <int>   <dbl>   <int>      <dbl>    <dbl>  
## 1 United ... Americas  2007     78.2 3.01e8    42952. 1.29e13  
## 2 China     Asia      2007     73.0 1.32e9    4959.  6.54e12  
## 3 Japan     Asia      2007     82.6 1.27e8    31656. 4.04e12  
## 4 India     Asia      2007     64.7 1.11e9    2452.  2.72e12  
## 5 Germany   Europe    2007     79.4 8.24e7    32170. 2.65e12  
## 6 United ... Europe    2007     79.4 6.08e7    33203. 2.02e12  
## 7 France    Europe    2007     80.7 6.11e7    30470. 1.86e12  
## 8 Brazil    Americas  2007     72.4 1.90e8    9066.  1.72e12  
## 9 Italy     Europe    2007     80.5 5.81e7    28570. 1.66e12  
## 10 Mexico   Americas  2007     76.2 1.09e8   11978. 1.30e12  
## # ... with 132 more rows
```

**NOTA 1:** se puede definir más de una variable **NOTA 2:** en la definición, se puede usar variable recién definida

# Añadir nuevas columnas

Múltiples funciones útiles para añadir columnas

- Operadores aritméticos
- Aritmética modular: división entera `%/%`, resto `%%`
- Logaritmos
- Agregaciones acumulativas `cumsum()`, `cumprod()`, `cummin()`,  
`cummax()`
- Comparaciones lógicas
- Rankings: `minrank()`

# Ejemplos

```
x = c(1,3,2,7,6)  
min_rank(x)
```

```
## [1] 1 3 2 5 4
```

```
33 %% 10
```

```
## [1] 3
```

```
cumsum(x)
```

```
## [1] 1 4 6 13 19
```

# Reto

Calcula la proporción de países en cada continente, que en el año 1967 tenía esperanza de vida entre 0-10, 10-20, 20-30, etc.

```
gapminder %>%
  mutate(lifeExpDecade = lifeExp - lifeExp%>%10) %>%
  filter(year == 1967) %>%
  count(continent, lifeExpDecade) %>%
  group_by(continent) %>%
  mutate(prop = n / sum(n))
```

```
## # A tibble: 17 × 4
## # Groups: continent [5]
##   continent lifeExpDecade     n     prop
##   <fct>          <dbl> <int>    <dbl>
## 1 Africa            30     10 0.192
## 2 Africa            40     29 0.558
## 3 Africa            50     11 0.212
## 4 Africa            60      2 0.0385
## 5 Americas          40      2 0.08
## 6 Americas          50      9 0.36
## 7 Americas          60     11 0.44
## 8 Americas          70      3 0.12
## 9 Asia              30      2 0.0606
## 10 Asia             40     10 0.303
## 11 Asia             50     13 0.394
## 12 Asia             60      5 0.152
## 13 Asia             70      3 0.0909
```

# Reto

Calcula la proporción de gente en cada continente, que en el año 1967 tenía esperanza de vida entre 0-10, 10-20, 20-30, etc.

```
gapminder %>%
  mutate(lifeExpDecade = case_when(lifeExp >= 20 & lifeExp< 30 ~ '20',
                                    lifeExp >= 30 & lifeExp< 40 ~ '30',
                                    lifeExp >= 40 & lifeExp< 50 ~ '40',
                                    lifeExp >= 50 & lifeExp< 60 ~ '50',
                                    lifeExp >= 60 & lifeExp< 70 ~ '60',
                                    lifeExp >= 70 & lifeExp< 80 ~ '70',
                                    lifeExp >= 80 & lifeExp< 90 ~ '80',
                                    lifeExp >= 90 & lifeExp< 100 ~ '90',
                                    TRUE ~ '>100')
)) %>%
filter(year == 1967) %>%
count(continent, lifeExpDecade) %>%
group_by(continent) %>%
mutate(prop = n / sum(n))
```

```
## # A tibble: 17 × 4
## # Groups: continent [5]
##   continent lifeExpDecade     n     prop
##   <fct>      <chr>     <int>    <dbl>
## 1 Africa      30          10  0.192
## 2 Africa      40          29  0.558
## 3 Africa      50          11  0.212
## 4 Africa      60           3  0.0385
## 5 Africa      70           1  0.0192
## 6 Africa      80           1  0.0192
## 7 Africa      90           1  0.0192
## 8 Africa     >100          1  0.0192
## 9 Asia        30          10  0.192
## 10 Asia       40          29  0.558
## 11 Asia       50          11  0.212
## 12 Asia       60           3  0.0385
## 13 Asia     >100          1  0.0192
## 14 Europe     30          10  0.192
## 15 Europe     40          29  0.558
## 16 Europe     50          11  0.212
## 17 Europe   >100          1  0.0192
```

# Transformación de datos: **summarise**

# Creación de estadísticos de resumen

```
gapminder %>%
  filter(year == 1967) %>%
  summarise(mean_gdpPerCapita = mean(gdpPercap))
```

```
## # A tibble: 1 × 1
##   mean_gdpPerCapita
##   <dbl>
## 1 5484.
```

**summarise()** colapsa todas las filas en un único estadístico de resumen y elimina las columnas no necesarias para el cálculo

# Resúmenes más útiles

- Localización: `mean()` y `median()`
- Dispersión: `sd()`, `IQR()`
- Rango: `min()`, `quantile(x, 0.25)`, `max()`
- Cuenta: `n()`, `n_distinct()`

# Resúmenes más útiles

¿Recuerdas que és la mediana? ¿Y el percentil 25?

# Transformación de datos: **group\_by**

# Transformación de datos: **group\_by**

# Agrupaciones

**summarise()** es tremadamente útil cuando se utiliza conjuntamente con **group\_by()**. Esto permite obtener estadísticos de resumen para diferentes grupos

```
gapminder %>%  
  filter(year == 1967) %>%  
  group_by(continent) %>%  
  summarise(mean_gdpPerCapita = mean(gdpPercap))
```

```
## # A tibble: 5 × 2  
##   continent mean_gdpPerCapita  
##   <fct>          <dbl>  
## 1 Africa           2050.  
## 2 Americas         5668.  
## 3 Asia             5971.  
## 4 Europe          10144.  
## 5 Oceania          14495.
```

# Agrupaciones

```
gapminder %>%
  filter(year == 1967) %>%
  group_by(continent) %>%
  summarise(n = n(),
            mean_gdpPerCapita = mean(gdpPercap),
            median_gdpPercapita = median(gdpPercap),
            )
```

```
## # A tibble: 5 × 4
##   continent      n  mean_gdpPerCapita median_gdpPercapita
##   <fct>     <int>        <dbl>              <dbl>
## 1 Africa       52        2050.             1210.
## 2 Americas     25        5668.             4643.
## 3 Asia         33        5971.             2029.
## 4 Europe       30       10144.            9366.
## 5 Oceania      2        14495.            14495.
```

**NOTA:** si hay valores ausentes, añadir **na.rm = TRUE**, e.g. **mean(x, na.rm = TRUE)**

# Agrupaciones por varias variables

```
gapminder %>%
  group_by(continent, year) %>%
  summarise(
    max_gdpPercapita      = max(gdpPercap),
    min_gdpPercapita      = min(gdpPercap)
  ) %>%
  filter(year %in% c(1967, 2007))
```

```
## # A tibble: 10 × 4
## # Groups:   continent [5]
##   continent   year max_gdpPercapita min_gdpPercapita
##   <fct>     <int>          <dbl>            <dbl>
## 1 Africa       1967        18773.         413.
## 2 Africa       2007        13206.         278.
## 3 Americas     1967        19530.         1452.
## 4 Americas     2007        42952.         1202.
## 5 Asia          1967        80895.          349
## 6 Asia          2007        47307.          944
## 7 Europe        1967        22966.         2172.
## 8 Europe        2007        49357.         5937.
## 9 Oceania       1967        14526.        14464.
## 10 Oceania      2007        34435.        25185.
```

# Bibliografía

- R for Data Science, Wickham and Grolemund (2016)