Gráficas bonitas: La elegancia de tidyverse

Isaac Arroyo

Septiembre 28, Octubre 1º y Octubre 5 del 2022

¿Qué es y que tiene de bonito el ✨`tidyverse`✨?

De acuerdo a la página oficial de `tidyverse`:

> El `tidyverse` es una colección de paqueterías de **R** diseñadas para la ciencia de datos.
>
> **Todas las paqueterías comparten una filosofía, diseño, gramática y estructura de datos.**

Paqueterías del `tidyverse`

- `dplyr`
- `tidyr`
- `stringr`
- `readr`
- `purr`
- `forcats`
- `tibble`
- `ggplot2`

¿Qué es lo que veremos durante la clase?

No hay visualización de **datos** sin datos. Previo a visualizarlos hay que encontrarlos y limpiarlos, dejarlos listos para que la visualización sea más fácil de hacer.

En esta sesión aprenderemos a limpiarlos así como darles sentido al manipularlos.

Instalar y cargar todo el ecosistema de `tidyverse`

Para instalar las 8 paqueterías que forman parte del ecosistema del `tidyverse` usaremos la siguiente función

```
install.packages("tidyverse")
```

Y para cargarla en nuestro documento usaremos la siguiente:

```
library(tidyverse)
```

Lo que estaremos usando en la sesión

Es decisión de cada persona si desea cargar todo tidyverse

```
library(tidyverse)
library(lubridate)
```

o seleccionar únicamente las paqueterías/bibliotecas que estará usando.

```
library(dplyr)
library(tidyr)
library(readr)
library(stringr)
```

Cargar datos

En esta sesión estaremos usando diferentes *datasets* o conjuntos de datos. En la presentación veran las diferentes maneras de cargar los datos, pero en el *script* que estarán usando será por la manera de **URL**.

En casi todas las ocasiones, la función `read_csv` de `readr` será más que suficiente.

```
1 data_nyt_bestsellers <- read_csv("./../datos/nyt_bestsellers-clean.csv")
2 head(data_nyt_bestsellers, 5)
```

```
# A tibble: 5 × 12
  amazon_prod…¹ author descr…² publi…³ title oid   bestsell…⁴ publishe…⁵ rank.…⁶
  <chr>         <chr>  <chr>   <chr>   <chr> <chr> <date>     <date>       <dbl>
1 http://www.a… Steph… "Alien… Little… THE … 5b4a… 2008-05-24 2008-06-08       2
2 http://www.a… Emily… "A wom… St. Ma… LOVE… 5b4a… 2008-05-24 2008-06-08       3
3 http://www.a… Patri… "A Mas… Putnam  THE … 5b4a… 2008-05-24 2008-06-08       4
4 http://www.a… Chuck… "An ag… Double… SNUFF 5b4a… 2008-05-24 2008-06-08       5
5 http://www.a… James… "A wom… Little… SUND… 5b4a… 2008-05-24 2008-06-08       6
# … with 3 more variables: rank_last_week.numberInt <dbl>,
#   weeks_on_list.numberInt <dbl>, price.numberDouble <dbl>, and abbreviated
#   variable names ¹amazon_product_url, ²description, ³publisher,
#   ⁴bestsellers_date.numberLong, ⁵published_date.numberLong, ⁶rank.numberInt
# ℹ Use `colnames()` to see all variable names
```

Conocer el `data.frame` o `tibble`

Quizás no siempre vayamos a querer ver todo el `data.frame`, para eso existen otras maneras de explorar el *dataset*.

`base::names` ➡️ nombres de columnas

Con `names` vamos a obtener el nombre de las columnas (sin saber el tipo de dato que contiene cada una)

```
1 names(data_nyt_bestsellers)
```

```
 [1] "amazon_product_url"           "author"
 [3] "description"                  "publisher"
 [5] "title"                        "oid"
 [7] "bestsellers_date.numberLong" "published_date.numberLong"
 [9] "rank.numberInt"               "rank_last_week.numberInt"
[11] "weeks_on_list.numberInt"      "price.numberDouble"
```

`dplyr::glimpse` ➡️ una idea de los datos

Y para tener una idea (o como dicen en inglés, *a glimpse*) usaremos `glimpse`

```
1  glimpse(data_nyt_bestsellers)
```

```
Rows: 3,033
Columns: 12
$ amazon_product_url          <chr> "http://www.amazon.com/The-Host-Novel-Step…
$ author                      <chr> "Stephenie Meyer", "Emily Giffin", "Patric…
$ description                 <chr> "Aliens have taken control of the minds an…
$ publisher                   <chr> "Little, Brown", "St. Martin's", "Putnam",…
$ title                       <chr> "THE HOST", "LOVE THE ONE YOU'RE WITH", "T…
$ oid                         <chr> "5b4aa4ead3089013507db18c", "5b4aa4ead3089…
$ bestsellers_date.numberLong <date> 2008-05-24, 2008-05-24, 2008-05-24, 2008-…
$ published_date.numberLong   <date> 2008-06-08, 2008-06-08, 2008-06-08, 2008-…
$ rank.numberInt              <dbl> 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 1,…
$ rank_last_week.numberInt    <dbl> 1, 2, 0, 0, 3, 4, 6, 8, 7, 9, 0, 10, 0, 2,…
$ weeks_on_list.numberInt     <dbl> 3, 2, 1, 1, 4, 3, 2, 3, 5, 2, 1, 4, 1, 4, …
$ price.numberDouble          <dbl> 25.99, 24.95, 22.95, 24.95, 24.99, 26.95, …
```

`utils::str` ➡️ estructura del *dataset*

Si queremos saber un poco más del *dataset* podemos obtener su **estructura** con `str`.

> La función `str` funciona mejor —nos da una respuesta un poco más clara— cuando transformamos nuestra `tibble` a un `data.frame`

```
1  str(as.data.frame(data_nyt_bestsellers))
```

```
'data.frame':   3033 obs. of  12 variables:
 $ amazon_product_url         : chr  "http://www.amazon.com/The-Host-Novel-Stephenie-Meyer/dp/0316218502?tag=NYTBS-
20" "http://www.amazon.com/Love-Youre-With-Emily-Giffin/dp/0312348665?tag=NYTBS-20" "http://www.amazon.com/The-
Front-Garano-Patricia-Cornwell-ebook/dp/B0017T0C9M?tag=NYTBS-20" "http://www.amazon.com/Snuff-Chuck-
Palahniuk/dp/0385517882?tag=NYTBS-20" ...
 $ author                     : chr  "Stephenie Meyer" "Emily Giffin" "Patricia Cornwell" "Chuck Palahniuk" ...
 $ description                : chr  "Aliens have taken control of the minds and bodies of most humans, but one
woman won't surrender." "A woman's happy marriage is shaken when she encounters an old boyfriend." "A Massachusetts
state investigator and his team from \"At Risk\" confront a rogue association of municipal police departments." "An
aging porn queens aims to cap her career by having sex on film with 600 men in one day." ...
 $ publisher                  : chr  "Little, Brown" "St. Martin's" "Putnam" "Doubleday" ...
 $ title                      : chr  "THE HOST" "LOVE THE ONE YOU'RE WITH" "THE FRONT" "SNUFF" ...
 $ oid                        : chr  "5b4aa4ead3089013507db18c" "5b4aa4ead3089013507db18d"
"5b4aa4ead3089013507db18e" "5b4aa4ead3089013507db18f" ...
 $ bestsellers_date.numberLong: Date, format: "2008-05-24" "2008-05-24" ...
 $ published_date.numberLong  : Date, format: "2008-06-08" "2008-06-08" ...
 $ rank.numberInt             : num  2 3 4 5 6 7 8 9 10 12 ...
 $ rank_last_week.numberInt   : num  1 2 0 0 3 4 6 8 7 9 ...
 $ weeks_on_list.numberInt    : num  3 2 1 1 4 3 2 3 5 2 ...
 $ price.numberDouble         : num  26.24 9.22 9.24 9.25
```

`dplyr::select` ➡️

Seleccionar columnas/variables

No toda la información nos importa, a veces solo queremos trabajar con algunas columnas y ya. Para eso usamos `select` y lo que hace es **seleccionar las columnas/variables** que le pidamos.

Seleccionar `author`, `title` y `price.numberDouble`

```
1 data_nyt_bestsellers
```

```
# A tibble: 3,033 × 12
   amazon_pro…¹ author descr…² publi…³ title oid   bestsell…⁴ publishe…⁵ rank.…⁶
   <chr>        <chr>  <chr>   <chr>   <chr> <chr> <date>     <date>       <dbl>
 1 http://www.… Steph… "Alien… Little… THE … 5b4a… 2008-05-24 2008-06-08       2
 2 http://www.… Emily… "A wom… St. Ma… LOVE… 5b4a… 2008-05-24 2008-06-08       3
 3 http://www.… Patri… "A Mas… Putnam  THE … 5b4a… 2008-05-24 2008-06-08       4
 4 http://www.… Chuck… "An ag… Double… SNUFF 5b4a… 2008-05-24 2008-06-08       5
 5 http://www.… James… "A wom… Little… SUND… 5b4a… 2008-05-24 2008-06-08       6
 6 http://www.… John … "The M… Putnam  PHAN… 5b4a… 2008-05-24 2008-06-08       7
 7 http://www.… Jimmy… "A Sou… Little… SWIN… 5b4a… 2008-05-24 2008-06-08       8
 8 http://www.… Eliza… "In Co… Harper  CARE… 5b4a… 2008-05-24 2008-06-08       9
 9 http://www.… David… "An in… Grand … THE … 5b4a… 2008-05-24 2008-06-08      10
10 http://www.… James… "A nov… Harper  BRIG… 5b4a… 2008-05-24 2008-06-08      12
# … with 3,023 more rows, 3 more variables: rank_last_week.numberInt <dbl>,
#   weeks_on_list.numberInt <dbl>, price.numberDouble <dbl>, and abbreviated
#   variable names ¹amazon_product_url, ²description, ³publisher,
#   ⁴bestsellers_date.numberLong, ⁵published_date.numberLong, ⁶rank.numberInt
# ℹ Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names
```

Seleccionar `author`, `title` y `price.numberDouble`

```
1  data_nyt_bestsellers %>%
2    select(author, title, price.numberDouble)
```

```
# A tibble: 3,033 × 3
   author                                  title                     price.num…¹
   <chr>                                   <chr>                           <dbl>
 1 Stephenie Meyer                         THE HOST                         26.0
 2 Emily Giffin                            LOVE THE ONE YOU'RE WITH         25.0
 3 Patricia Cornwell                       THE FRONT                        23.0
 4 Chuck Palahniuk                         SNUFF                            25.0
 5 James Patterson and Gabrielle Charbonnet SUNDAYS AT TIFFANY'S            25.0
 6 John Sandford                           PHANTOM PREY                     27.0
 7 Jimmy Buffett                           SWINE NOT?                       22.0
 8 Elizabeth George                        CARELESS IN RED                  28.0
 9 David Baldacci                          THE WHOLE TRUTH                  27.0
10 James Frey                              BRIGHT SHINY MORNING             27.0
# … with 3,023 more rows, and abbreviated variable name ¹price.numberDouble
# ℹ Use `print(n = ...)` to see more rows
```

Ejercicio con `dplyr::select`

Crea un nuevo *data frame* llamado **`df_nyt_original`**, con las siguientes columnas:

- `author`
- `title`
- `description`
- `publisher`
- `description`
- `publisher`
- `rank.numberInt`
- `weeks_on_list.numberInt`
- `price.numberDouble`

```
1  df_nyt_original <- data_nyt_bestsellers %>%
2    select(author, title, description, publisher, rank.numberInt,
3           weeks_on_list.numberInt, price.numberDouble)
4  # Ver el data frame
5  glimpse(df_nyt_original)
```

```
Rows: 3,033
Columns: 7
$ author                 <chr> "Stephenie Meyer", "Emily Giffin", "Patricia C…
$ title                  <chr> "THE HOST", "LOVE THE ONE YOU'RE WITH", "THE F…
$ description            <chr> "Aliens have taken control of the minds and bo…
$ publisher              <chr> "Little, Brown", "St. Martin's", "Putnam", "Do…
$ rank.numberInt         <dbl> 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 1, 2, …
$ weeks_on_list.numberInt <dbl> 3, 2, 1, 1, 4, 3, 2, 3, 5, 2, 1, 4, 1, 4, 3, 5…
$ price.numberDouble     <dbl> 25.99, 24.95, 22.95, 24.95, 24.99, 26.95, 21.9…
```

`dplyr::mutate` ➡️

Crear nuevas columnas

En muchas ocasiones vamos a tener que cambiar columnas o crear nuestras propias columnas basándonas en una o más de las que tengamos en el *dataset*. A este proceso `dplyr` lo conoce como **mutar** y se hará con `mutate`

Crear columna que tome en cuenta la inflación

La inflación más grande que ha tenido Estados Unidos de América ha sido del 8.5%, por lo que los precios estan un poco más altos de lo que aparece en el conjunto de datos, vamos a modificarlos.

```
df_nyt_original_ajustePrecio <- df_nyt_original %>%
  mutate(precio_inflacion = price.numberDouble * 1.085)
```

Crear columna que tome en cuenta la inflación

La inflación más grande que ha tenido Estados Unidos de América ha sido del 8.5%, por lo que los precios estan un poco más altos de lo que aparece en el conjunto de datos, vamos a modificarlos.

```
df_nyt_original_ajustePrecio <- df_nyt_original %>%
  mutate(precio_inflacion = price.numberDouble * 1.085)

# Vamos a ver los precios de antes y despues del cambio
df_nyt_original_ajustePrecio %>%
  select(price.numberDouble, precio_inflacion) %>%
  head(10)
```

```
# A tibble: 10 × 2
   price.numberDouble precio_inflacion
                <dbl>             <dbl>
 1                 26.0              28.2
 2                 25.0              27.1
 3                 23.0              24.9
 4                 25.0              27.1
 5                 25.0              27.1
```

6	27.0	29.2
7	22.0	23.9
8	28.0	30.3
9	27.0	29.3
10	27.0	29.2

Ejercicio con `dplyr::mutate`

Crea una columna llamada `precio_mxn` que contenga el precio de la inflación (`precio_inflacion`) en pesos mexicanos. La conversión es la siguiente: **1 USD = 20 MXN** [1].

Guardar la nueva columna en la misma variable `df_nyt_original_ajustePrecio`

1. Es una conversión aproximada, si quieren el dato real pueden preguntar o investigar. La presentación se hizo antes y el precio del dolar varía

```r
df_nyt_original_ajustePrecio <- df_nyt_original_ajustePrecio %>%
  mutate(precio_mxn =  precio_inflacion * 20)

# Ver los 3 precios
df_nyt_original_ajustePrecio %>%
  select(price.numberDouble, precio_inflacion, precio_mxn) %>%
  head(5)
```

```
# A tibble: 5 × 3
  price.numberDouble precio_inflacion precio_mxn
               <dbl>            <dbl>      <dbl>
1               26.0             28.2       564.
2               25.0             27.1       541.
3               23.0             24.9       498.
4               25.0             27.1       541.
5               25.0             27.1       542.
```

`tidyr::separate` ➡️

Generar columnas de información de texto

Algunas veces tendremos que lidiar con datos tipo texto, *string*, *characther*. algunas veces, resulta mejor crear columnas por separado de esos *strings*.Por ejemplo nombres y apellidos, direcciones, o fechas escritas.

Separar nombre y apellido de la columna `author`

En nuetro *dataset* contamos con el nombre y apellido de las y los autores de los libros en una misma columna, pero nos gustaría separarlos por nombre y apellidos.

```
df_nyt_original_ajustePrecio_autorxs <- df_nyt_original_ajustePrecio %>%
  separate(author,
           sep = ' ',
           into = c("autorx_nombre","autorx_apellido"),
           extra = "merge"
           )

# Ver el resultado
df_nyt_original_ajustePrecio_autorxs %>% glimpse()
```

```
Rows: 3,033
Columns: 10
$ autorx_nombre           <chr> "Stephenie", "Emily", "Patricia", "Chuck",
"Ja…
$ autorx_apellido         <chr> "Meyer", "Giffin", "Cornwell", "Palahniuk",
"P…
$ title                   <chr> "THE HOST", "LOVE THE ONE YOU'RE WITH", "THE
F…
```

```
$ description              <chr> "Aliens have taken control of the minds and
bo…
$ publisher                <chr> "Little, Brown", "St. Martin's", "Putnam",
"Do…
$ rank.numberInt           <dbl> 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 1, 2,
…
$ weeks_on_list.numberInt <dbl> 3, 2, 1, 1, 4, 3, 2, 3, 5, 2, 1, 4, 1, 4, 3,
5
```

`dplyr::case_when`

➡️ Crear nueva columna de acuerdo a condiciones

En algunas ocasiones será necesario hacer más que manipulaciones matemáticas, separación de palabras o cambio de estructura de datos. Necesitaremos de condiciones.

Clasificar precios de libros

Todos sabemos que los precios de dólares a pesos mexicanos crecen demasidado. Ahora vamos a "clasificar" esos precios de acuerdo a rangos: el primero ***Menos de 520 pesos***, el segundo de ***Menos de 580 pesos pero más de 520 pesos*** y el último ***Mayor a 580 pesos***

```
df_nyt_original_ajustePrecio_autorxs_categorias <- df_nyt_original_ajustePr
  mutate(categoria_precio = case_when(
    precio_mxn < 520 ~ "Se puede pagar",
    precio_mxn > 520 & precio_mxn < 580 ~ "Ya se empieza a poner caro",
    T ~ "A caso vuela?"))
# Ver las primras lineas
df_nyt_original_ajustePrecio_autorxs_categorias %>%
  select(precio_mxn, categoria_precio) %>%
  head(8)
```

```
# A tibble: 8 × 2
  precio_mxn categoria_precio
       <dbl> <chr>
1       564. Ya se empieza a poner caro
2       541. Ya se empieza a poner caro
3       498. Se puede pagar
4       541. Ya se empieza a poner caro
```

```
5       542. Ya se empieza a poner caro
6       585. A caso vuela?
7       477. Se puede pagar
8       607. A caso vuela?
```

`dplyr::rename` ➡️

Renombrar columnas

Algunos nombres de columnas tienen caracteres especiales, son nombres poco claros o simplemente son muy largos, oara ello hay que **renombrar**.

Cambio a nombres más claros

Vamos a cambiar el nombre de algunas columnas:

```r
df_nyt_renombrado <- df_nyt_original_ajustePrecio_autorxs_categorias %>%
  rename(# nombre_nuevo = nombre_viejo
         descripcion = description,
         titulo_libro = title,
         editorial = publisher,
         rank_lista = rank.numberInt,
         semanas_en_lista = weeks_on_list.numberInt) %>%
  # Vamos a quedarnos con los siguientes
  select(autorx_nombre, autorx_apellido, titulo_libro, descripcion,
         editorial, rank_lista, semanas_en_lista, precio_mxn,
         categoria_precio)

glimpse(df_nyt_renombrado)
```

```
Rows: 3,033
Columns: 9
$ autorx_nombre    <chr> "Stephenie", "Emily", "Patricia", "Chuck", "James",
"…
$ autorx_apellido  <chr> "Meyer", "Giffin", "Cornwell", "Palahniuk",
"Patterso…
$ titulo_libro     <chr> "THE HOST", "LOVE THE ONE YOU'RE WITH", "THE FRONT",
…
$ descripcion      <chr> "Aliens have taken control of the minds and bodies
of…
$ editorial        <chr> "Little, Brown", "St. Martin's", "Putnam",
```

```
"Doubleday…
$ rank_lista       <dbl> 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 1, 2, 4, 5,
6…
$ semanas_en_lista <dbl> 3, 2, 1, 1, 4, 3, 2, 3, 5, 2, 1, 4, 1, 4, 3, 5, 1,
2
```

`dplyr::filter` ➡️

Filtrar de acuerdo a condiciones

`filter` nos ayudará a seccionar nuestro *dataset* de acuerdo a una o más condiciones, será un **filtro**.

Cómo se usa

Pseudocódigo

```
datos:
  FILTRAR por condicion
```

Como se vé con código

```r
datos %>%
  filter(condicion)
```

Libros con más de 14 semanas en la lista del ranking

```
df_nyt_renombrado %>%
  filter(semanas_en_lista > 14)
```

```
# A tibble: 290 × 9
   autorx_nombre autor…¹ titul…² descr…³ edito…⁴ rank_…⁵ seman…⁶ preci…⁷
categ…⁸
   <chr>         <chr>   <chr>   <chr>   <chr>     <dbl>   <dbl>   <dbl> <chr>
 1 Stephenie     Meyer   THE HO… Aliens… Little…       6      15    564. Ya se
…
 2 Stephenie     Meyer   THE HO… Aliens… Little…       5      16    564. Ya se
…
 3 Stephenie     Meyer   THE HO… Aliens… Little…       5      17    564. Ya se
…
 4 Stephenie     Meyer   THE HO… Aliens… Little…       6      18    564. Ya se
…
 5 Stephenie     Meyer   THE HO… Aliens… Little…       6      19    564. Ya se
…
 6 David         Wroble… THE ST… A mute… Ecco          1      15    563. Ya se
```

Filtrar por apellidos

```r
df_nyt_renombrado %>%
  filter(autorx_apellido %in% c("Meyer", "King"))
```

```
# A tibble: 68 × 9
   autorx_nombre autor…¹ titul…² descr…³ edito…⁴ rank_…⁵ seman…⁶ preci…⁷
categ…⁸
   <chr>         <chr>   <chr>   <chr>   <chr>     <dbl>   <dbl>   <dbl> <chr>
 1 Stephenie     Meyer   THE HO… Aliens… Little…       2       3    564. Ya se
…
 2 Stephenie     Meyer   THE HO… Aliens… Little…       2       4    564. Ya se
…
 3 Stephenie     Meyer   THE HO… Aliens… Little…       2       5    564. Ya se
…
 4 Stephenie     Meyer   THE HO… Aliens… Little…       3       6    564. Ya se
…
 5 Stephenie     Meyer   THE HO… Aliens… Little…       3       7    564. Ya se
…
 6 Stephenie     Meyer   THE HO… Aliens… Little…       5       8    564. Ya se
```

Por palabras clave en las descripciones

```r
df_nyt_renombrado %>%
  filter(str_detect(descripcion, "woman|child")) %>%
  select(titulo_libro, descripcion)
```

```
# A tibble: 486 × 2
   titulo_libro             descripcion
   <chr>                    <chr>
 1 THE HOST                 Aliens have taken control of the minds and bodies
o…
 2 LOVE THE ONE YOU'RE WITH A woman's happy marriage is shaken when she
encount…
 3 SUNDAYS AT TIFFANY’S     A woman finds an unexpected love
 4 THE HOST                 Aliens have taken control of the minds and bodies
o…
 5 LOVE THE ONE YOU'RE WITH A woman’s happy marriage is shaken when she
encount…
 6 SUNDAYS AT TIFFANY’S     A woman finds an unexpected love.
 7 THE HOST                 Aliens have taken control of the minds and bodies
o…
 8 LOVE THE ONE YOU'RE WITH A woman’s happy marriage is shaken when she
```

Por palabras clave y ranking semanal

Pseudocódigo

```
datos:
  FILTRAR por condicion_1 Y condicion_2
```

Como se vé con código

```
1 datos %>%
2   filter(condicion_1, condicion_2)
```

Por palabras clave y ranking semanal

```r
df_nyt_renombrado %>%
  select(autorx_apellido, titulo_libro, descripcion, semanas_en_lista)
```

```
# A tibble: 3,033 × 4
   autorx_apellido                    titulo_libro            descrip…¹ seman…²
   <chr>                              <chr>                   <chr>       <dbl>
 1 Meyer                              THE HOST                "Aliens …       3
 2 Giffin                             LOVE THE ONE YOU'RE WITH "A woman…       2
 3 Cornwell                           THE FRONT               "A Massa…       1
 4 Palahniuk                          SNUFF                   "An agin…       1
 5 Patterson and Gabrielle Charbonnet SUNDAYS AT TIFFANY’S    "A woman…       4
 6 Sandford                           PHANTOM PREY            "The Min
```

Por palabras clave y ranking semanal

```r
df_nyt_renombrado %>%
  filter(str_detect(descripcion, "woman|child"),# Primer filtro
         ) %>%
  select(autorx_apellido, titulo_libro, descripcion, semanas_en_lista)
```

```
# A tibble: 486 × 4
   autorx_apellido                titulo_libro             descrip…¹ seman…²
   <chr>                          <chr>                    <chr>       <dbl>
 1 Meyer                          THE HOST                 Aliens h…       3
 2 Giffin                         LOVE THE ONE YOU'RE WITH A woman'…       2
 3 Patterson and Gabrielle Charbonnet SUNDAYS AT TIFFANY’S     A woman …       4
 4 Meyer                          THE HOST                 Aliens h…       4
 5 Giffin                         LOVE THE ONE YOU'RE WITH A woman’…       3
 6 Patterson and Gabrielle Charbonnet SUNDAYS AT TIFFANY’S     A woman
```

Por palabras clave y ranking semanal

```r
df_nyt_renombrado %>%
  filter(str_detect(descripcion, "woman|child"),# Primer filtro
         semanas_en_lista >= 10 # Segundo filtro
         ) %>%
  select(autorx_apellido, titulo_libro, descripcion, semanas_en_lista)
```

```
# A tibble: 175 × 4
   autorx_apellido titulo_libro             descripcion                 seman…¹
   <chr>           <chr>                    <chr>                         <dbl>
 1 Meyer           THE HOST                 Aliens have taken control o…     10
 2 Meyer           THE HOST                 Aliens have taken control o…     11
 3 Giffin          LOVE THE ONE YOU'RE WITH A woman’s happy marriage is…     10
 4 Meyer           THE HOST                 Aliens have taken control o…     12
 5 Giffin          LOVE THE ONE YOU'RE WITH A woman’s happy marriage is…     11
 6 Meyer           THE HOST                 Aliens have taken control o
```

Cambio de *dataset*

Para los siguientes ejercicios vamos a estar usando un conjunto de datos diferente (para irle variando y que vayan practicando)

```
1 data_mortality_infants <- read_csv("https://raw.githubusercontent.com/isaac
```

Ejercicio: Preparación de los datos

Antes de aprender los otros conceptos, un pequeño ejercicio para repasar las funciones que vimos con anterioridad.

1. **Seleccionar** las columnas:

- ***Entity***
- ***Year***
- ***Deaths - Cardiovascular diseases - Sex: Both - Age: Under 5 (Number)***
- ***Deaths - Environmental heat and cold exposure - Sex: Both - Age: Under 5 (Number)***.

2. **Filtrar** la columna ***Entity*** para que solo esten los continentes del mundo.
3. **Renombrar** las columnas seleccionadas.
4. Almacenar todos los cambios anteriores en una **nueva variable llamada *df_mi***

```
1 data_mortality_infants
```

```
# A tibble: 8,220 × 32
   Entity    Code   Year Death…¹ Death…² Death…³ Death…⁴ Death…⁵ Death…⁶
Death…⁷
   <chr>     <chr> <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
<dbl>
 1 Afghanis… AFG    1990      48     105    1779     718     431    8649
477
 2 Afghanis… AFG    1991      55     130    1822     741     439    8669
495
 3 Afghanis… AFG    1992      68     155    2069     836     486    8539
554
 4 Afghanis… AFG    1993      78     178    2427     970     549    8949
630
 5 Afghanis… AFG    1994      83     194    2649    1063     589   10642
681
 6 Afghanis… AFG    1995      86     199    2681    1098     605   12121
```

```r
data_mortality_infants %>%
  select(Entity, Year,
         `Deaths - Cardiovascular diseases - Sex: Both - Age: Under 5 (Numb
         `Deaths - Environmental heat and cold exposure - Sex: Both - Age:
```

```
# A tibble: 8,220 × 4
   Entity       Year Deaths - Cardiovascular diseases - Sex: Both - Ag…¹
Death…²
   <chr>       <dbl>                                              <dbl>
<dbl>
 1 Afghanistan  1990                                              225
5
 2 Afghanistan  1991                                              229
3
 3 Afghanistan  1992                                              255
1
 4 Afghanistan  1993                                              291
1
 5 Afghanistan  1994                                              313
1
 6 Afghanistan  1995                                              319
```

```r
vector_entity <- c("Africa", "Oceania","America", "Asia", "Europe")

data_mortality_infants %>%
  select(Entity, Year,
         `Deaths - Cardiovascular diseases - Sex: Both - Age: Under 5 (Numb
         `Deaths - Environmental heat and cold exposure - Sex: Both - Age:
  filter(Entity %in% vector_entity)
```

```
# A tibble: 150 × 4
   Entity  Year Deaths - Cardiovascular diseases - Sex: Both - Age: Un…¹
Death…²
   <chr>  <dbl>                                                    <dbl>
<dbl>
 1 Africa  1990                                                    31606
1057
 2 Africa  1991                                                    30792
1057
 3 Africa  1992                                                    30185
1063
 4 Africa  1993                                                    29317
1063
 5 Africa  1994                                                    28235
1064
 6 Africa  1995                                                    27346
```

```r
vector_entity <- c("Africa", "Oceania","America", "Asia", "Europe")

data_mortality_infants %>%
  select(Entity, Year,
         `Deaths - Cardiovascular diseases - Sex: Both - Age: Under 5 (Numb
         `Deaths - Environmental heat and cold exposure - Sex: Both - Age:
  filter(Entity %in% vector_entity) %>%
  rename(country = Entity, date_year = Year,
         deaths_cardiovascular = `Deaths - Cardiovascular diseases - Sex: B
         deaths_env = `Deaths - Environmental heat and cold exposure - Sex:
```

```
# A tibble: 150 × 4
   country date_year deaths_cardiovascular deaths_env
   <chr>       <dbl>                 <dbl>      <dbl>
 1 Africa       1990                 31606       1057
 2 Africa       1991                 30792       1057
 3 Africa       1992                 30185       1063
 4 Africa       1993                 29317       1063
 5 Africa       1994                 28235       1064
 6 Africa       1995                 27346       1062
 7 Africa       1996                 26006       1052
 8 Africa       1997                 24982       1043
 9 Africa       1998                 24253       1038
10 Africa       1999                 23851       1036
# … with 140 more rows
# ℹ Use `print(n = ...)` to see more rows
```

```r
vector_entity <- c("Africa", "Oceania","America", "Asia", "Europe")

df_mi <- data_mortality_infants %>%
  select(Entity, Year,
         `Deaths - Cardiovascular diseases - Sex: Both - Age: Under 5 (Numb
         `Deaths - Environmental heat and cold exposure - Sex: Both - Age:
  filter(Entity %in% vector_entity) %>%
  rename(country = Entity, date_year = Year,
         deaths_cardiovascular = `Deaths - Cardiovascular diseases - Sex: B
         deaths_env = `Deaths - Environmental heat and cold exposure - Sex:

glimpse(df_mi)
```

```
Rows: 150
Columns: 4
$ country              <chr> "Africa", "Africa", "Africa", "Africa",
"Africa"…
$ date_year            <dbl> 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997,
…
$ deaths_cardiovascular <dbl> 31606, 30792, 30185, 29317, 28235, 27346,
26006,…
$ deaths_env           <dbl> 1057, 1057, 1063, 1063, 1064, 1062, 1052, 1043,
…
```

`dplyr::summarise`

➡️ Resumir información de columnas

A veces solo queremos conocer el "comportamiento" general de los datos en las columnas de nuestro *dataset*, para eso usamos **agregaciones** o **reducciones**. En `dplyr` se realiza bajo la función `summarise` (del inglés, resumir)

Obtener las estadísticas básicas del conjuntos de datos

Calcular la media y desviación estándar de las columnas de `df_mi`

```
df_mi %>%
  summarise(media_death_cardio = mean(deaths_cardiovascular),
            desviacion_death_cardio = sd(deaths_cardiovascular),
            media_death_env = mean(deaths_env),
            desviacion_death_env = sd(deaths_env))
```

```
# A tibble: 1 × 4
  media_death_cardio desviacion_death_cardio media_death_env
desviacion_death_…¹
               <dbl>                   <dbl>           <dbl>
<dbl>
1              9699.                  10661.            699.
1012.
# … with abbreviated variable name ¹desviacion_death_env
```

`dplyr::group_by` ➡️

Agrupar por categorías

Crear uno o más grupos para manipular datos de una manera más específica y descubrir más detalles en ellos.

Cambio o diferencia de fallecimientos con respecto al año anterior en cada uno de los continentes

Para esta tarea vamos a crear **un solo grupo** ➡️`country`

```
1 df_mi
```

```
# A tibble: 150 × 4
   country date_year deaths_cardiovascular deaths_env
   <chr>       <dbl>                 <dbl>      <dbl>
 1 Africa       1990                 31606       1057
 2 Africa       1991                 30792       1057
 3 Africa       1992                 30185       1063
 4 Africa       1993                 29317       1063
 5 Africa       1994                 28235       1064
 6 Africa       1995                 27346       1062
 7 Africa       1996                 26006       1052
 8 Africa       1997                 24982       1043
 9 Africa       1998                 24253       1038
10 Africa       1999                 23851       1036
# … with 140 more rows
# ℹ Use `print(n = ...)` to see more rows
```

```r
df_mi %>%
  group_by(country)
```

```
# A tibble: 150 × 4
   country date_year deaths_cardiovascular deaths_env
   <chr>       <dbl>                 <dbl>      <dbl>
 1 Africa       1990                 31606       1057
 2 Africa       1991                 30792       1057
 3 Africa       1992                 30185       1063
 4 Africa       1993                 29317       1063
 5 Africa       1994                 28235       1064
 6 Africa       1995                 27346       1062
 7 Africa       1996                 26006       1052
 8 Africa       1997                 24982       1043
 9 Africa       1998                 24253       1038
10 Africa       1999                 23851       1036
# … with 140 more rows
# ℹ Use `print(n = ...)` to see more rows
```

```r
df_mi %>%
  group_by(country) %>%
  summarise(change_death_cardio = diff(deaths_cardiovascular),
            change_death_env = diff(deaths_env))
```

```
# A tibble: 145 × 3
   country change_death_cardio change_death_env
   <chr>                 <dbl>            <dbl>
 1 Africa                 -814                0
 2 Africa                 -607                6
 3 Africa                 -868                0
 4 Africa                -1082                1
 5 Africa                 -889               -2
 6 Africa                -1340              -10
 7 Africa                -1024               -9
 8 Africa                 -729               -5
 9 Africa                 -402               -2
10 Africa                 -468               -2
# … with 135 more rows
# ℹ Use `print(n = ...)` to see more rows
```

```r
df_mi %>%
  group_by(country) %>%
  summarise(change_death_cardio = diff(deaths_cardiovascular),
            change_death_env = diff(deaths_env)) %>%
  mutate(date_year = seq(1991,2019))
```

```
# A tibble: 145 × 4
   country change_death_cardio change_death_env date_year
   <chr>                 <dbl>             <dbl>     <int>
 1 Africa                 -814                 0      1991
 2 Africa                 -607                 6      1992
 3 Africa                 -868                 0      1993
 4 Africa                -1082                 1      1994
 5 Africa                 -889                -2      1995
 6 Africa                -1340               -10      1996
 7 Africa                -1024                -9      1997
 8 Africa                 -729                -5      1998
 9 Africa                 -402                -2      1999
10 Africa                 -468                -2      2000
# … with 135 more rows
# ℹ Use `print(n = ...)` to see more rows
```

```r
df_mi %>%
  group_by(country) %>%
  summarise(change_death_cardio = diff(deaths_cardiovascular),
            change_death_env = diff(deaths_env)) %>%
  mutate(date_year = seq(1991,2019)) %>%
  ungroup()
```

```
# A tibble: 145 × 4
   country change_death_cardio change_death_env date_year
   <chr>                 <dbl>             <dbl>     <int>
 1 Africa                 -814                 0      1991
 2 Africa                 -607                 6      1992
 3 Africa                 -868                 0      1993
 4 Africa                -1082                 1      1994
 5 Africa                 -889                -2      1995
 6 Africa                -1340               -10      1996
 7 Africa                -1024                -9      1997
 8 Africa                 -729                -5      1998
 9 Africa                 -402                -2      1999
10 Africa                 -468                -2      2000
# … with 135 more rows
# ℹ Use `print(n = ...)` to see more rows
```

Ejercicio con `dplyr::group_by` y `dplyr::summarise`

Obten la media y desviación estándar de `df_mi` por continente

```r
df_mi %>%
  summarise(media_death_cardio = mean(deaths_cardiovascular),
            desviacion_death_cardio = sd(deaths_cardiovascular),
            media_death_env = mean(deaths_env),
            desviacion_death_env = sd(deaths_env))
```

```
# A tibble: 1 × 4
  media_death_cardio desviacion_death_cardio media_death_env
desviacion_death_…¹
               <dbl>                   <dbl>           <dbl>
<dbl>
1              9699.                  10661.            699.
1012.
# … with abbreviated variable name ¹desviacion_death_env
```

```r
df_mi %>%
  group_by(country) %>%
  summarise(media_death_cardio = mean(deaths_cardiovascular),
            desviacion_death_cardio = sd(deaths_cardiovascular),
            media_death_env = mean(deaths_env),
            desviacion_death_env = sd(deaths_env)) %>%
  ungroup()
```

```
# A tibble: 5 × 5
  country media_death_cardio desviacion_death_cardio media_death_env
desviacio…¹
  <chr>                <dbl>                   <dbl>           <dbl>
<dbl>
1 Africa              22974.                  4082.           1026.
45.3
2 America              4145.                   964.             87.8
24.0
3 Asia                20014.                  8559.           2261.      1178.
4 Europe               1122.                   480.            108.
54.7
5 Oceania               241.                    50.3            10.3
2.16
# … with abbreviated variable name ¹desviacion_death_env
```

`dplyr::lubridate`

➡️ manejo de fechas

Las fechas y series de tiempo son tipos de datos que siempre estarán en demanda y esta paquetería facilita muchas de las tareas.

Instalar y cargar `lubridate`

```
1 install.packages("lubridate")
2 library(lubridate)
```

Además, vamos a hacer otro cambio de *dataset* (específico para ejemplos de manipulación del tiempo)

```
1 data_vuelos_tt <- read_csv("https://raw.githubusercontent.com/rfordatascien
2 glimpse(data_vuelos_tt)
```

```
Rows: 688,099
Columns: 14
$ YEAR          <dbl> 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016,
20…
$ MONTH_NUM     <chr> "01", "01", "01", "01", "01", "01", "01", "01", "01",
"0…
$ MONTH_MON     <chr> "JAN", "JAN", "JAN", "JAN", "JAN", "JAN", "JAN", "JAN",
…
$ FLT_DATE      <dttm> 2016-01-01, 2016-01-01, 2016-01-01, 2016-01-01, 2016-
01…
$ APT_ICAO      <chr> "EBAW", "EBBR", "EBCI", "EBLG", "EBOS", "EDDB", "EDDC",
…
$ APT_NAME      <chr> "Antwerp", "Brussels", "Charleroi", "Liège", "Ostend-
Bru…
```

Seleccionar y renombrar columnas

- FLT_DATE ➡️ full_date
- APT_NAME ➡️ airport_name
- STATE_NAME ➡️ country
- FLT_TOT_1 ➡️ total_movements

```
1 data_vuelos_tt
```

```
# A tibble: 688,099 × 14
    YEAR MONTH_NUM MONTH_MON FLT_DATE            APT_I…¹ APT_N…² STATE…³
FLT_D…⁴
   <dbl> <chr>     <chr>     <dttm>              <chr>   <chr>   <chr>
<dbl>
 1  2016 01        JAN       2016-01-01 00:00:00 EBAW    Antwerp Belgium
4
 2  2016 01        JAN       2016-01-01 00:00:00 EBBR    Brusse… Belgium
174
 3  2016 01        JAN       2016-01-01 00:00:00 EBCI    Charle… Belgium
45
 4  2016 01        JAN       2016-01-01 00:00:00 EBLG    Liège   Belgium
6
 5  2016 01        JAN       2016-01-01 00:00:00 EBOS    Ostend… Belgium
```

```
7
 6  2016-01          JAN         2016-01-01 00:00:00 EDDB    Berlin, Germany
```

Seleccionar y renombrar columnas

- FLT_DATE ➡️ full_date
- APT_NAME ➡️ airport_name
- STATE_NAME ➡️ country
- FLT_TOT_1 ➡️ total_movements

```
1 data_vuelos_tt %>%
2   select(FLT_DATE, APT_NAME, STATE_NAME, FLT_TOT_1)
```

```
# A tibble: 688,099 × 4
   FLT_DATE            APT_NAME             STATE_NAME FLT_TOT_1
   <dttm>              <chr>                <chr>           <dbl>
 1 2016-01-01 00:00:00 Antwerp              Belgium             7
 2 2016-01-01 00:00:00 Brussels             Belgium           345
 3 2016-01-01 00:00:00 Charleroi            Belgium            92
 4 2016-01-01 00:00:00 Liège                Belgium            13
 5 2016-01-01 00:00:00 Ostend-Bruges        Belgium            14
 6 2016-01-01 00:00:00 Berlin - Brandenburg Germany           197
 7 2016-01-01 00:00:00 Dresden              Germany            39
 8 2016-01-01 00:00:00 Erfurt               Germany             2
 9 2016-01-01 00:00:00 Frankfurt            Germany           742
10 2016-01-01 00:00:00 Muenster-Osnabrueck  Germany             7
```

```
# … with 688,089 more rows
# ℹ Use `print(n = ...)` to see more rows
```

Seleccionar y renombrar columnas

- FLT_DATE ➡️ full_date
- APT_NAME ➡️ airport_name
- STATE_NAME ➡️ country
- FLT_TOT_1 ➡️ total_movements

```
1 data_vuelos_tt %>%
2   select(FLT_DATE, APT_NAME, STATE_NAME, FLT_TOT_1) %>%
3   rename(full_date = FLT_DATE, airport_name = APT_NAME,
4          country = STATE_NAME, total_movements = FLT_TOT_1)
```

```
# A tibble: 688,099 × 4
   full_date           airport_name         country total_movements
   <dttm>              <chr>                <chr>             <dbl>
 1 2016-01-01 00:00:00 Antwerp              Belgium               7
 2 2016-01-01 00:00:00 Brussels             Belgium             345
 3 2016-01-01 00:00:00 Charleroi            Belgium              92
 4 2016-01-01 00:00:00 Liège                Belgium              13
 5 2016-01-01 00:00:00 Ostend-Bruges        Belgium              14
 6 2016-01-01 00:00:00 Berlin - Brandenburg Germany             197
 7 2016-01-01 00:00:00 Dresden              Germany              39
 8 2016-01-01 00:00:00 Erfurt               Germany               2
```

```
 9 2016-01-01 00:00:00 Frankfurt             Germany              742
10 2016-01-01 00:00:00 Muenster-Osnabrueck   Germany                7
# … with 688,089 more rows
# ℹ Use `print(n = ...)` to see more rows
```

Seleccionar y renombrar columnas

- FLT_DATE ➡️ full_date
- APT_NAME ➡️ airport_name
- STATE_NAME ➡️ country
- FLT_TOT_1 ➡️ total_movements

```
df_flies <- data_vuelos_tt %>%
  select(FLT_DATE, APT_NAME, STATE_NAME, FLT_TOT_1) %>%
  rename(full_date = FLT_DATE, airport_name = APT_NAME,
         country = STATE_NAME, total_movements = FLT_TOT_1)

glimpse(df_flies)
```

```
Rows: 688,099
Columns: 4
$ full_date       <dttm> 2016-01-01, 2016-01-01, 2016-01-01, 2016-01-01,
2016-…
$ airport_name    <chr> "Antwerp", "Brussels", "Charleroi", "Liège", "Ostend-
B…
```

Formato e información de fechas con `ymd()`, `year()`, `month()`, entre otras

El formato de fecha con el que actualmente contamos no es malo, pero podemos cambiarlo dependiendo de la situación/necesidades. Por ejemplo, podemos obtener el mes (número y nombre), año, semana e inclusive cambiar el formato de la fecha.

lubridate + dplyr::mutate

```
1 df_flies
```

```
# A tibble: 688,099 × 4
   full_date           airport_name         country total_movements
   <dttm>              <chr>                <chr>             <dbl>
 1 2016-01-01 00:00:00 Antwerp              Belgium               7
 2 2016-01-01 00:00:00 Brussels             Belgium             345
 3 2016-01-01 00:00:00 Charleroi            Belgium              92
 4 2016-01-01 00:00:00 Liège                Belgium              13
 5 2016-01-01 00:00:00 Ostend-Bruges        Belgium              14
 6 2016-01-01 00:00:00 Berlin - Brandenburg Germany             197
 7 2016-01-01 00:00:00 Dresden              Germany              39
 8 2016-01-01 00:00:00 Erfurt               Germany               2
 9 2016-01-01 00:00:00 Frankfurt            Germany             742
10 2016-01-01 00:00:00 Muenster-Osnabrueck  Germany               7
# … with 688,089 more rows
# ℹ Use `print(n = ...)` to see more rows
```

lubridate + dplyr::mutate

```r
df_flies %>%
  mutate(full_date = ymd(full_date)
         )
```

```
# A tibble: 688,099 × 4
   full_date  airport_name          country total_movements
   <date>     <chr>                 <chr>              <dbl>
 1 2016-01-01 Antwerp               Belgium                7
 2 2016-01-01 Brussels              Belgium              345
 3 2016-01-01 Charleroi             Belgium               92
 4 2016-01-01 Liège                 Belgium               13
 5 2016-01-01 Ostend-Bruges         Belgium               14
 6 2016-01-01 Berlin - Brandenburg  Germany              197
 7 2016-01-01 Dresden               Germany               39
 8 2016-01-01 Erfurt                Germany                2
 9 2016-01-01 Frankfurt             Germany              742
10 2016-01-01 Muenster-Osnabrueck   Germany                7
# … with 688,089 more rows
# ℹ Use `print(n = ...)` to see more rows
```

lubridate + dplyr::mutate

```r
df_flies %>%
  mutate(full_date = ymd(full_date),
         date_year = year(full_date)
         )
```

```
# A tibble: 688,099 × 5
   full_date  airport_name          country total_movements date_year
   <date>     <chr>                 <chr>             <dbl>     <dbl>
 1 2016-01-01 Antwerp               Belgium               7      2016
 2 2016-01-01 Brussels              Belgium             345      2016
 3 2016-01-01 Charleroi             Belgium              92      2016
 4 2016-01-01 Liège                 Belgium              13      2016
 5 2016-01-01 Ostend-Bruges         Belgium              14      2016
 6 2016-01-01 Berlin - Brandenburg  Germany             197      2016
 7 2016-01-01 Dresden               Germany              39      2016
 8 2016-01-01 Erfurt                Germany               2      2016
 9 2016-01-01 Frankfurt             Germany             742      2016
10 2016-01-01 Muenster-Osnabrueck   Germany               7      2016
# … with 688,089 more rows
# ℹ Use `print(n = ...)` to see more rows
```

lubridate + dplyr::mutate

```r
df_flies %>%
  mutate(full_date = ymd(full_date),
         date_year = year(full_date),
         date_month = month(full_date)
         )
```

```
# A tibble: 688,099 × 6
   full_date  airport_name         country total_movements date_year
date_month
   <date>     <chr>                <chr>             <dbl>     <dbl>
<dbl>
 1 2016-01-01 Antwerp              Belgium               7      2016
1
 2 2016-01-01 Brussels             Belgium             345      2016
1
 3 2016-01-01 Charleroi            Belgium              92      2016
1
 4 2016-01-01 Liège                Belgium              13      2016
1
 5 2016-01-01 Ostend-Bruges        Belgium              14      2016
1
 6 2016-01-01 Berlin - Brandenburg Germany             197      2016
```

lubridate + dplyr::mutate

```r
df_flies %>%
  mutate(full_date = ymd(full_date),
         date_year = year(full_date),
         date_month = month(full_date, label = T)
         )
```

```
# A tibble: 688,099 × 6
   full_date  airport_name         country total_movements date_year
date_month
   <date>     <chr>                <chr>             <dbl>     <dbl> <ord>
 1 2016-01-01 Antwerp              Belgium               7      2016 Jan
 2 2016-01-01 Brussels             Belgium             345      2016 Jan
 3 2016-01-01 Charleroi            Belgium              92      2016 Jan
 4 2016-01-01 Liège                Belgium              13      2016 Jan
 5 2016-01-01 Ostend-Bruges        Belgium              14      2016 Jan
 6 2016-01-01 Berlin - Brandenburg Germany             197      2016 Jan
 7 2016-01-01 Dresden              Germany              39      2016 Jan
 8 2016-01-01 Erfurt               Germany               2      2016 Jan
 9 2016-01-01 Frankfurt            Germany             742      2016 Jan
10 2016-01-01 Muenster-Osnabrueck  Germany               7      2016 Jan
# … with 688,089 more rows
# ℹ Use `print(n = ...)` to see more rows
```

lubridate + dplyr::mutate

```r
df_flies %>%
  mutate(full_date = ymd(full_date),
         date_year = year(full_date),
         date_month = month(full_date, label = T, abbr = F)
         )
```

```
# A tibble: 688,099 × 6
   full_date  airport_name         country total_movements date_year
date_month
   <date>     <chr>                <chr>             <dbl>     <dbl> <ord>
 1 2016-01-01 Antwerp              Belgium               7      2016 January
 2 2016-01-01 Brussels             Belgium             345      2016 January
 3 2016-01-01 Charleroi            Belgium              92      2016 January
 4 2016-01-01 Liège                Belgium              13      2016 January
 5 2016-01-01 Ostend-Bruges        Belgium              14      2016 January
 6 2016-01-01 Berlin - Brandenburg Germany             197      2016 January
 7 2016-01-01 Dresden              Germany              39      2016 January
 8 2016-01-01 Erfurt               Germany               2      2016 January
 9 2016-01-01 Frankfurt            Germany             742      2016 January
10 2016-01-01 Muenster-Osnabrueck  Germany               7      2016 January
# … with 688,089 more rows
# ℹ Use `print(n = ...)` to see more rows
```

lubridate + dplyr::mutate

```
df_flies %>%
  mutate(full_date = ymd(full_date),
         date_year = year(full_date),
         date_month = month(full_date, label = T, abbr = F)
         ) %>%
  str()
```

```
tibble [688,099 × 6] (S3: tbl_df/tbl/data.frame)
 $ full_date      : Date[1:688099], format: "2016-01-01" "2016-01-01" ...
 $ airport_name   : chr [1:688099] "Antwerp" "Brussels" "Charleroi" "Liège"
...
 $ country        : chr [1:688099] "Belgium" "Belgium" "Belgium" "Belgium" ...
 $ total_movements: num [1:688099] 7 345 92 13 14 197 39 2 742 7 ...
 $ date_year      : num [1:688099] 2016 2016 2016 2016 2016 ...
 $ date_month     : Ord.factor w/ 12 levels "January"<"February"<..: 1 1 1 1 1
1 1 1 1 1 ...
```

lubridate + dplyr::mutate

```r
df_flies <- df_flies %>%
  mutate(full_date = ymd(full_date),
         date_year = year(full_date),
         date_month = month(full_date, label = T, abbr = F)
         )

glimpse(df_flies)
```

```
Rows: 688,099
Columns: 6
$ full_date       <date> 2016-01-01, 2016-01-01, 2016-01-01, 2016-01-01,
2016-…
$ airport_name    <chr> "Antwerp", "Brussels", "Charleroi", "Liège", "Ostend-
B…
$ country         <chr> "Belgium", "Belgium", "Belgium", "Belgium",
"Belgium",…
$ total_movements <dbl> 7, 345, 92, 13, 14, 197, 39, 2, 742, 7, 252, 182,
347,…
$ date_year       <dbl> 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016,
…
$ date_month      <ord> January, January, January, January, January, January,
…
```

Agrupar y resumir los vuelos del mes por cada país (1a manera)

Existen dos maneras de agrupar por fechas, la primera es tratar el año o mes como una categoría.

```
df_flies %>%
  group_by(country, date_year, date_month) %>%
  summarise(sum_movements_month = sum(total_movements))
```

```
# A tibble: 3,184 × 4
   country date_year date_month sum_movements_month
   <chr>       <dbl> <ord>                     <dbl>
 1 Albania      2016 January                    1666
 2 Albania      2016 February                   1507
 3 Albania      2016 March                      1676
 4 Albania      2016 April                      1680
 5 Albania      2016 May                        1710
 6 Albania      2016 June                       1720
 7 Albania      2016 July                       2104
 8 Albania      2016 August                     2353
 9 Albania      2016 September                  2150
10 Albania      2016 October                    2021
# … with 3,174 more rows
# ℹ Use `print(n = ...)` to see more rows
```

`lubridate::floor_date()` ➡️

Agrupar y resumir los vuelos del mes por cada país (2a manera)

La segunda manera es usando funciones de modificación de fechas (así les llama `lubridate`)

```
df_flies %>%
  group_by(country, month_sample = floor_date(full_date, "month")) %>%
  summarise(sum_movements_month = sum(total_movements))
```

```
# A tibble: 3,184 × 3
   country month_sample sum_movements_month
   <chr>   <date>                      <dbl>
 1 Albania 2016-01-01                   1666
 2 Albania 2016-02-01                   1507
 3 Albania 2016-03-01                   1676
 4 Albania 2016-04-01                   1680
 5 Albania 2016-05-01                   1710
 6 Albania 2016-06-01                   1720
 7 Albania 2016-07-01                   2104
 8 Albania 2016-08-01                   2353
 9 Albania 2016-09-01                   2150
10 Albania 2016-10-01                   2021
# … with 3,174 more rows
```

```
# … with 3,174 more rows
# ℹ Use `print(n = ...)` to see more rows
```

¿Qué los diferencia?

En la primera manera

```
df_flies %>%
  group_by(country, date_year, date_month)
```

se tiene por “cortado” el formato de fecha al tener la columna de año y de mes de manera independiente.

Mientras que la segunda manera

```
df_flies %>%
  group_by(country, date_sample = floor_date(full_date, "month")
```

se tiene en conjunto, conserva la continuidad de línea del tiempo.

¿Qué son los datos *wide* y *long format*?

En pocas palabras, el ***long format*** viene en la forma **clave**, **atributo**, **valor** mientras que el ***wide format*** **incorpora los atributos como nombres de las columnas.** Para entender mejor, usaremos más ejemplos

Ejemplo de *wide format*: Cargamos más datos

Para comprender mejor lo que significa los datos en *wide format* usaremos el siguiente conjunto de datos.

```
1 data_motomami <- read_csv("https://raw.githubusercontent.com/isaacarroyov/3
2 data_motomami
```

```
# A tibble: 16 × 14
   song_name artist album numbe…¹ song_…² dance…³ energy loudn…⁴ durat…⁵
speec…⁶
   <chr>     <chr>  <chr>   <dbl> <chr>     <dbl>  <dbl>   <dbl>   <dbl>
<dbl>
 1 SAOKO     ROSAL… MOTO…       1 spotif…   0.827 0.768    -5.70  137533
0.265
 2 CANDY     ROSAL… MOTO…       2 spotif…   0.638 0.49     -5.73  193480
0.226
 3 LA FAMA … ROSAL… MOTO…       3 spotif…   0.766 0.295    -7.89  188107
0.0464
 4 BULERÍAS  ROSAL… MOTO…       4 spotif…   0.774 0.458    -6.30  155880
0.303
 5 CHICKEN … ROSAL… MOTO…       5 spotif…   0.788 0.4      -6.45  122227
0.115
```

Ejemplo de *long format*: Datos mensuales del número de los vuelos de un país de Europa (Italia)

df_flies

```r
df_flies_it <- df_flies %>%
  filter(country == "Italy") %>%
  select(date_year, date_month, total_movements) %>%
  group_by(date_year, date_month) %>%
  summarise(sum_total_movements = sum(total_movements)) %>%
  ungroup()

df_flies_it
```

```
# A tibble: 77 × 3
   date_year date_month sum_total_movements
       <dbl> <ord>                     <dbl>
 1      2016 January                   77573
 2      2016 February                  76599
 3      2016 March                     86649
 4      2016 April                     91934
 5      2016 May                      102369
 6      2016 June                     104086
 7      2016 July                     113218
```

```
 8      2016 August                   109481
 9      2016 September                106287
10      2016 October                   97973
# … with 67 more rows
# ℹ Use `print(n = ...)` to see more rows
```

`tidyr::pivot_longer` ➡️ De *wide* a *long format* (Ejemplo)

El cambio que haremos será que cada atributo/variable, que en este caso son los *audio features*, estén en una sola columna llamada ***audio_features***

Para esto está la función `tidyr::pivot_longer`, cuyos principales argumentos/parámetros son:

- **`cols`** ➡️ Las columnas que serán *pivoteadas* al *long format*.
- **`names_to`** ➡️ El nombre de la **nueva columna** de las columnas puestas en **`cols`**.
- **`values_to`** ➡️ El nombre de la **columna** de donde se encontrarán los **valores** de la **nueva columna**.

`tidyr::pivot_longer` ➡️ De *wide* a *long format* (código)

```
1 data_motomami
```

```
# A tibble: 16 × 14
   song_name artist album numbe…¹ song_…² dance…³ energy loudn…⁴ durat…⁵
speec…⁶
   <chr>     <chr>  <chr>   <dbl> <chr>     <dbl>  <dbl>   <dbl>   <dbl>
<dbl>
 1 SAOKO     ROSAL… MOTO…       1 spotif…   0.827 0.768    -5.70  137533
0.265
 2 CANDY     ROSAL… MOTO…       2 spotif…   0.638 0.49     -5.73  193480
0.226
 3 LA FAMA … ROSAL… MOTO…       3 spotif…   0.766 0.295    -7.89  188107
0.0464
 4 BULERÍAS  ROSAL… MOTO…       4 spotif…   0.774 0.458    -6.30  155880
0.303
 5 CHICKEN … ROSAL… MOTO…       5 spotif…   0.788 0.4      -6.45  122227
0.115
 6 HENTAI    ROSAL… MOTO…       6 spotif…   0.583 0.297   -11.3   162907
```

`tidyr::pivot_longer` ➡️ De *wide* a *long format* (código)

```
1 data_motomami %>%
2   pivot_longer(cols = 6:14, names_to = "audio_feature", values_to = "score"
```

```
# A tibble: 144 × 7
   song_name artist  album    number_song song_uri              audio…¹
score
   <chr>     <chr>   <chr>          <dbl> <chr>                 <chr>
<dbl>
 1 SAOKO     ROSALÍA MOTOMAMI           1 spotify:track:2FYGZD… dancea…
8.27e-1
 2 SAOKO     ROSALÍA MOTOMAMI           1 spotify:track:2FYGZD… energy
7.68e-1
 3 SAOKO     ROSALÍA MOTOMAMI           1 spotify:track:2FYGZD… loudne…
-5.70e+0
 4 SAOKO     ROSALÍA MOTOMAMI           1 spotify:track:2FYGZD… durati…
1.38e+5
 5 SAOKO     ROSALÍA MOTOMAMI           1 spotify:track:2FYGZD… speech…
2.65e-1
 6 SAOKO     ROSALÍA MOTOMAMI           1 spotify:track:2FYGZD… acoust…  7.9
```

`tidyr::pivot_longer` ➡️ De *wide* a *long format* (código)

```
1  data_motomami %>%
2    pivot_longer(cols = 6:14, names_to = "audio_feature", values_to = "score"
3    select(song_name, artist, audio_feature, score)
```

```
# A tibble: 144 × 4
   song_name artist  audio_feature       score
   <chr>     <chr>   <chr>               <dbl>
 1 SAOKO     ROSALÍA danceability      8.27e-1
 2 SAOKO     ROSALÍA energy            7.68e-1
 3 SAOKO     ROSALÍA loudness         -5.70e+0
 4 SAOKO     ROSALÍA duration_ms       1.38e+5
 5 SAOKO     ROSALÍA speechiness       2.65e-1
 6 SAOKO     ROSALÍA acousticness      7.9 e-1
 7 SAOKO     ROSALÍA instrumentalness  2.45e-5
 8 SAOKO     ROSALÍA valence           7.34e-1
 9 SAOKO     ROSALÍA tempo             1.00e+2
10 CANDY     ROSALÍA danceability      6.38e-1
# … with 134 more rows
# ℹ Use `print(n = ...)` to see more rows
```

`tidyr::pivot_longer` ➡️ De *wide* a *long format*

¿Y esto de qué sirve?

Bueno, cuando entremos al apartado de **Gramática de Gráficos** nos daremos cuenta que en varias ocasiones este tipo de formato de datos nos ayudará a transmitir mejor nuestras ideas para darle color, forma o separar los datos.

`tidyr::pivot_wider` ➡️ De *long* a *wide format* (Ejemplo)

Para el caso de los vuelos de Italia el cambio será que cada **columna ➡️ mes**, cada **fila ➡️ año** y que los **valores dentro de la tabla ➡️ número de vuelos**.

Vamos a usar la función `tidyr::pivot_wider`, cuyos principales argumentos son los siguientes:

- **`names_from`** ➡️ La variable de donde serán la **nuevas columnas**.
- **`values_from`** ➡️ La variable de donde serán los **valores dentro de la tabla**.

`tidyr::pivot_wider` ➡️ De *long* a *wide format* (código)

```
1 df_flies_it
```

```
# A tibble: 77 × 3
   date_year date_month sum_total_movements
       <dbl> <ord>                     <dbl>
 1      2016 January                   77573
 2      2016 February                  76599
 3      2016 March                     86649
 4      2016 April                     91934
 5      2016 May                      102369
 6      2016 June                     104086
 7      2016 July                     113218
 8      2016 August                   109481
 9      2016 September                106287
10      2016 October                   97973
# … with 67 more rows
# ℹ Use `print(n = ...)` to see more rows
```

`tidyr::pivot_wider` ➡️ De *long* a *wide format* (código)

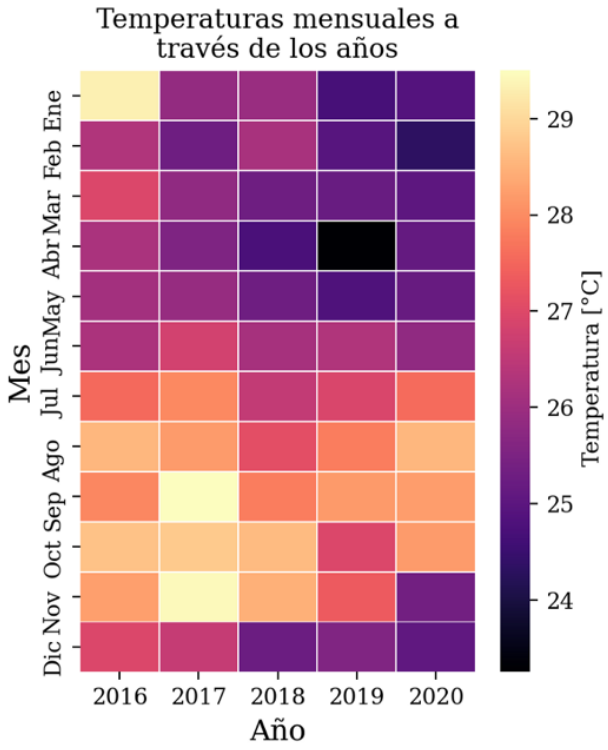
```r
df_flies_it %>%
  pivot_wider(names_from = date_month, values_from = sum_total_movements)
```

```
# A tibble: 7 × 13
  date_year January February March  April    May   June   July August
September
      <dbl>   <dbl>    <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
<dbl>
1      2016   77573    76599 86649  91934 102369 104086 113218 109481
106287
2      2017   79890    74298 87258  95123 105649 109524 117255 112253
110089
3      2018   83455    77071 90878  99219 106404 110593 119111 115888
112506
4      2019   87776    82137 92802 103566 110738 115142 121828 117164
113633
5      2020   86861    81904 31660   7653   9342  16499  43897  57998
48702
6      2021   23980    19934 24748  30022  34853  57765  82950  91947
```

`tidyr::pivot_wider` ➡️ De *long* a *wide format*

¿Y esto de qué sirve?

En algunas ocasiones, vamos a querer mostrar tablas con colores (conocidos como *heat maps*) como una manera alterna de representar clasificaciones, composición de los datos o series de tiempo. En estas ocasiones el formato *wide* es más amable con nosotrxs.



El kit básico para manipular datos

Cargar, conocer, seleccinar, crear/modificar, filtrar, agrupar y manejar fechas son de las tareas que estaremos realizando continuamente.

¿De memoria? No

No se las tienen que aprender de memoria, pero si es importante conocerlas y que sepan que existen.

En las siguientes sesiones vamos a estar usándolas para seguir practicando, conoceremos más tips para facilitar el filtrado o selección así como otras funciones que en esta sesión decidí no poner porque son en casos específicos.

¿Qué veremos en la siguiente sesión?

Después de tener datos limpios y “presentables”, llega la hora de **presentarlos** y **comunicarlos** de una manera visual.

En la semana les daré algún *spoiler* de las gráficas que haremos.

Muchas gracias por su atención, paciencia y por estar aquí 😊

los tqm