

# Manipulación de datos en R

**tidyr y dplyr**

**Dr. Samuel D. Gamboa Tuz**

# Tidyverse

- El Tidyverse es un "meta-paquete", una colección de paquetes de R para ciencia de datos que comparten cierta filosofía, gramática y estructura de datos: "tidydata" (<https://www.tidyverse.org/>). Los paquetes "core" son:

Paquete	Descripción
tibble	Alternativa al <i>data frame</i> .
readr	Funciones para importar datos.
ggplot2	Creación de gráficos.
<b>tidyr</b>	Cambios en la estructura de los datos (tidy).
<b>dplyr</b>	Transformación y manipulación de datos.
purrr	Iteración sobre listas. Alternativa a <i>for loops</i> y <i>apply</i> .
forcats	Funciones para trabajar con factores.
stringr	Funciones para trabajar con caracteres ( <i>strings</i> ).

- Los paquetes **magrittr** y **readxl**, entre otros, también se encuentran incluidos al instalar el Tidyverse."

- Se pueden cargar todos los paquetes del tidyverse con `library(tidyverse)`.

```
library(tidyverse)
search()
```

```
## [1] ".GlobalEnv"          "package:gdtools"
## [3] "package:palmerpenguins" "package:forcats"
## [5] "package:stringr"      "package:dplyr"
## [7] "package:purrr"        "package:readr"
## [9] "package:tidyr"        "package:tibble"
## [11] "package:ggplot2"      "package:tidyverse"
## [13] "tools:rstudio"        "package:stats"
## [15] "package:graphics"     "package:grDevices"
## [17] "package:utils"        "package:datasets"
## [19] "package:methods"      "Autoloads"
## [21] "package:base"
```

- También se puede instalar y cargar cada paquete de manera individual si así se requiere.

# magrittr - piping en R

- El paquete *magrittr* instala unos operadores tipo *pipe* (como el "|" de la terminal de linux). Buena alternativa para escribir funciones anidadas.
- El pipe más popular de magrittr es `%>%`, su shortcut en Rstudio es Ctrl + Shit + M.

```
x <- rep(1:3, 100)
x %>%
  head() %>%
  unique() %>%
  as.character()
```

```
## [1] "1" "2" "3"
```

- Ayuda considerablemente a la legibilidad del código.

**tidyr**

*pivot\_longer, pivot\_wider, separate, unite*

# Tidydata

- Cada variable va en una columna.
- Cada observación va en una fila.
- Cada valor tiene su propia celda.

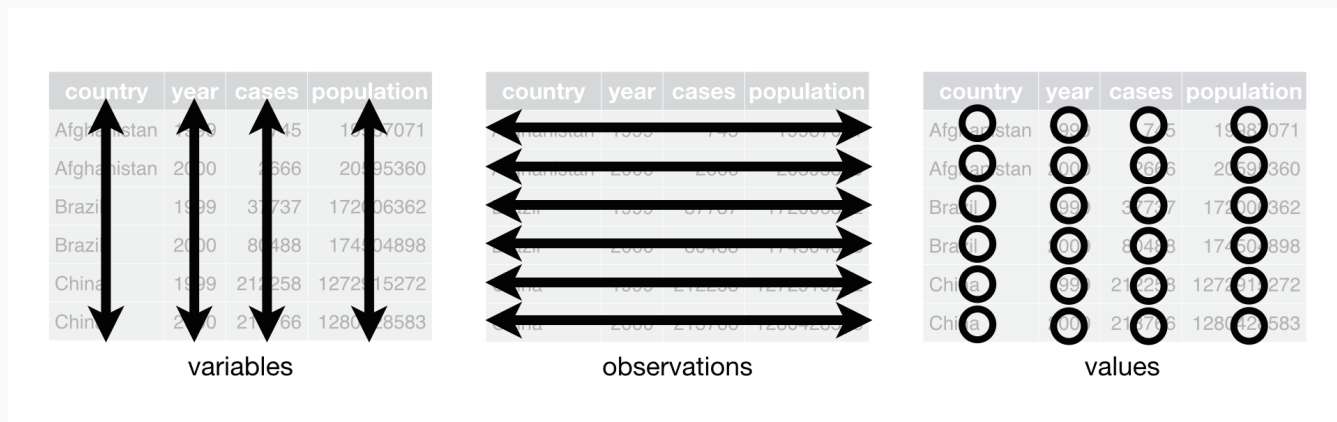


Figura tomada de <https://r4ds.had.co.nz/>

# pivot\_longer - Cambiar a formato largo

- Los valores de una sola variable pueden estar dispersas en varias columnas (formato ancho):

```
maize_data <- readxl::read_excel("datasets.xlsx", sheet = 1)
maize_data
```

```
## # A tibble: 10 x 11
##   Area `2009` `2010` `2011` `2012` `2013` `2014` `2015` `2016`
##   <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Arge... 1.31e7 2.27e7 2.38e7 2.12e7 3.21e7 3.31e7 3.38e7 3.98e7
## 2 Braz... 5.07e7 5.54e7 5.57e7 7.11e7 8.03e7 7.99e7 8.53e7 6.42e7
## 3 China 1.64e8 1.78e8 1.93e8 2.06e8 2.19e8 2.16e8 2.65e8 2.64e8
## 4 Chin... 1.64e8 1.77e8 1.93e8 2.06e8 2.18e8 2.16e8 2.65e8 2.64e8
## 5 Fran... 1.55e7 1.40e7 1.59e7 1.54e7 1.50e7 1.83e7 1.37e7 1.18e7
## 6 India 1.67e7 2.17e7 2.18e7 2.23e7 2.43e7 2.42e7 2.26e7 2.59e7
## 7 Indo... 1.76e7 1.83e7 1.76e7 1.94e7 1.85e7 1.90e7 1.96e7 2.36e7
## 8 Mexi... 2.01e7 2.33e7 1.76e7 2.21e7 2.27e7 2.33e7 2.47e7 2.83e7
## 9 Ukra... 1.05e7 1.20e7 2.28e7 2.10e7 3.09e7 2.85e7 2.33e7 2.81e7
## 10 Unit... 3.32e8 3.16e8 3.13e8 2.73e8 3.51e8 3.61e8 3.45e8 4.12e8
## # ... with 2 more variables: `2017` <dbl>, `2018` <dbl>
```

- Con `pivot_longer()` las podemos transformar a formato largo para poder graficar con `ggplot2` (antes era `gather()`).

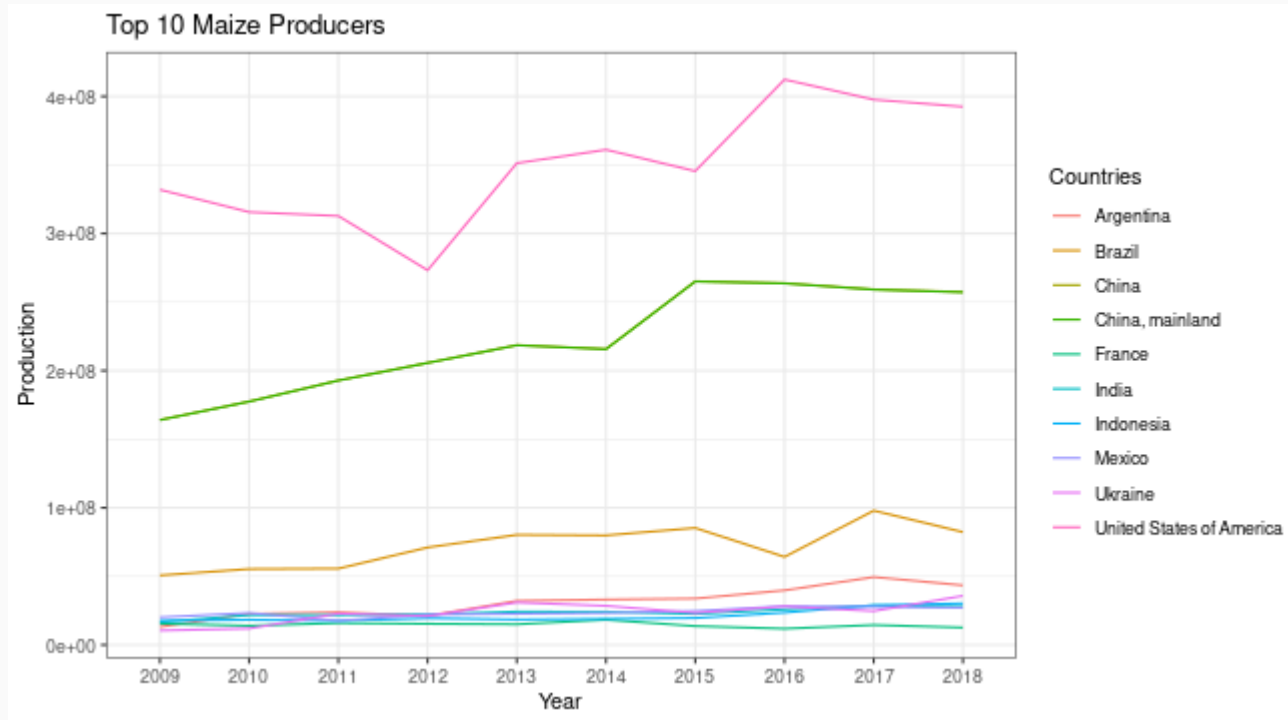
```
maize_data_long <- maize_data %>% pivot_longer(names_to = "Year",
                                                values_to = "Production",
                                                cols = `2009`:`2018`)
maize_data_long$Year <- factor(maize_data_long$Year)
maize_data_long
```

```
## # A tibble: 100 x 3
##   Area      Year Production
##   <chr>    <fct>      <dbl>
## 1 Argentina 2009      13121380
## 2 Argentina 2010      22663095
## 3 Argentina 2011      23799830
## 4 Argentina 2012      21196637
## 5 Argentina 2013      32119211
## 6 Argentina 2014      33087165
## 7 Argentina 2015      33817744
## 8 Argentina 2016      39792854
## 9 Argentina 2017      49475895
## 10 Argentina 2018      43462323
## # ... with 90 more rows
```



- Ahora ya podemos graficar con ggplot2:

```
ggplot(maize_data_long, aes(Year, Production,  
                           color = Area, group = Area)) +  
  geom_line() +  
  labs(title = "Top 10 Maize Producers") +  
  scale_color_discrete(name = "Countries") +  
  theme_bw()
```



# `pivot_wider()` - cambiar a formato ancho

- Supongamos que queremos visualizar los datos anteriores como un heatmap agrupado, podemos regresar los datos al formato ancho con `pivot_wider()` (antes era `spread()`).

```
maize_data_wide <- maize_data_long %>%  
  pivot_wider(names_from = "Year", values_from = "Production")  
maize_data_wide
```

```
## # A tibble: 10 x 11  
##   Area `2009` `2010` `2011` `2012` `2013` `2014` `2015` `2016`  
##   <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 Arge... 1.31e7 2.27e7 2.38e7 2.12e7 3.21e7 3.31e7 3.38e7 3.98e7  
## 2 Braz... 5.07e7 5.54e7 5.57e7 7.11e7 8.03e7 7.99e7 8.53e7 6.42e7  
## 3 China 1.64e8 1.78e8 1.93e8 2.06e8 2.19e8 2.16e8 2.65e8 2.64e8  
## 4 Chin... 1.64e8 1.77e8 1.93e8 2.06e8 2.18e8 2.16e8 2.65e8 2.64e8  
## 5 Fran... 1.55e7 1.40e7 1.59e7 1.54e7 1.50e7 1.83e7 1.37e7 1.18e7  
## 6 India 1.67e7 2.17e7 2.18e7 2.23e7 2.43e7 2.42e7 2.26e7 2.59e7  
## 7 Indo... 1.76e7 1.83e7 1.76e7 1.94e7 1.85e7 1.90e7 1.96e7 2.36e7  
## 8 Mexi... 2.01e7 2.33e7 1.76e7 2.21e7 2.27e7 2.33e7 2.47e7 2.83e7  
## 9 Ukra... 1.05e7 1.20e7 2.28e7 2.10e7 3.09e7 2.85e7 2.33e7 2.81e7  
## 10 Unit... 3.32e8 3.16e8 3.13e8 2.73e8 3.51e8 3.61e8 3.45e8 4.12e8  
## # ... with 2 more variables: `2017` <dbl>, `2018` <dbl>
```

## Paréntesis: covertir de tibble a matrix

- Por lo general, el input para un heatmap es una matriz, no un data frame o tibble.
- Para ello es necesario convertir con `as.matrix()`, pero no se puede hacer en un solo *call* desde un tibble porque **no reconoce rownames**:

```
maize_data_matrix <- as.matrix(maize_data_wide[,-1]) # Matriz
rownames(maize_data_matrix) <- maize_data_wide[[1]] # Nombres
maize_data_matrix %>% class()
```

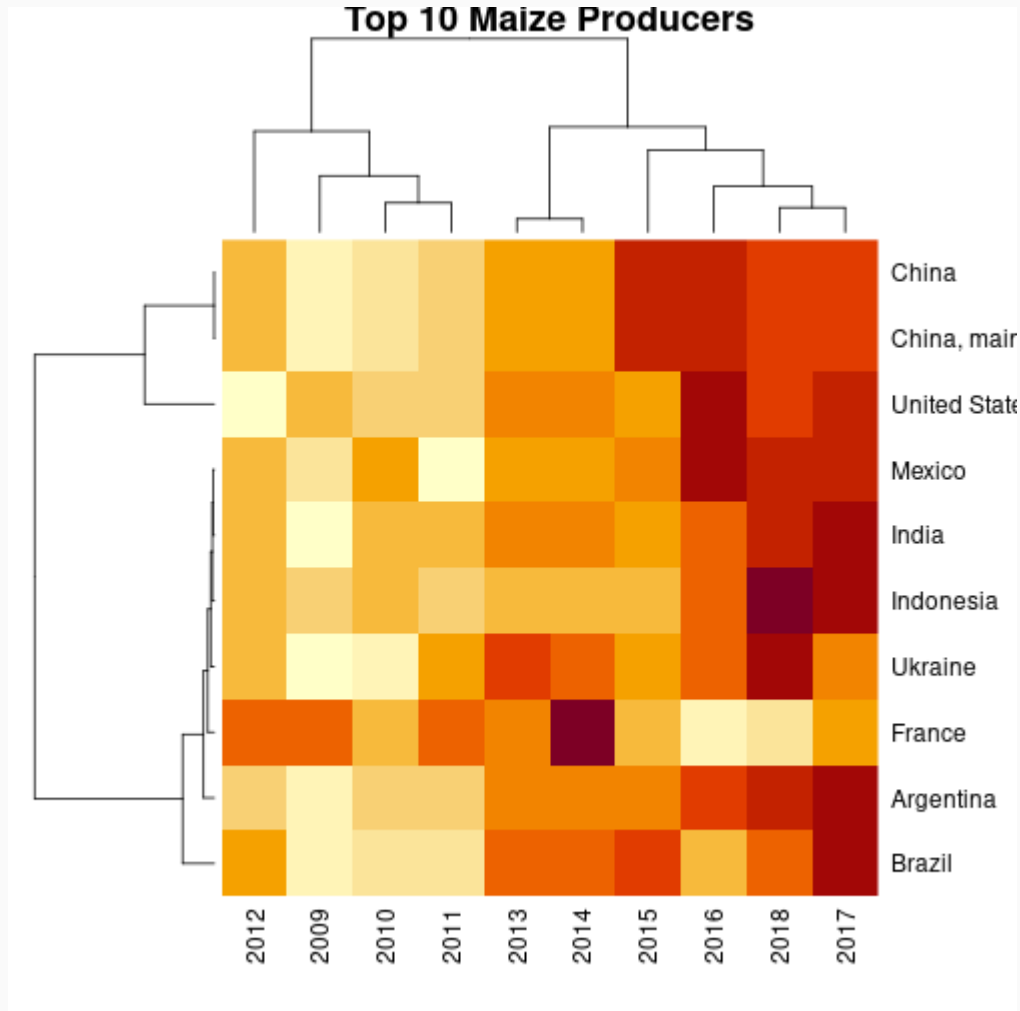
```
## [1] "matrix" "array"
```

```
maize_data_matrix %>% head(4)
```

```
##           2009      2010      2011      2012      2013
## Argentina 13121380 22663095 23799830 21196637 32119211
## Brazil    50719822 55364271 55660235 71072810 80273172
## China     164107560 177540788 192904232 205719284 218621905
## China, mainland 163974000 177425000 192781000 205614100 218489000
##           2014      2015      2016      2017      2018
## Argentina 33087165 33817744 39792854 49475895 43462323
## Brazil    79881614 85283074 64188314 97910658 82288298
## China     215812100 265157307 263777750 259256299 257348659
## China, mainland 215646300 264992000 263613000 259071000 257173900
```

- Ejemplo de heatmap:

```
heatmap(maize_data_matrix, main = "Top 10 Maize Producers")
```



# Separar columnas con `separate()`

- En el siguiente ejemplo, podemos separar la columna "species" en las columnas "genera" y "species" con `separate()`:

```
plants <- readxl::read_excel("datasets.xlsx", sheet = 2)
plants %>% head(3)
```

```
## # A tibble: 3 x 4
##   group      group_color species      genome_size
##   <chr>      <chr>      <chr>          <dbl>
## 1 Green algae #8f9a9b    Chlamydomonas reinhardtii    110
## 2 Green algae #8f9a9b    Volvox carteri                138
## 3 Bryophytes  #008933    Physcomitrella patens        472
```

```
plants_separated <- plants %>%
  separate(species, c("genera", "species"), sep = " ", remove = TRUE)
plants_separated %>% head(3)
```

```
## # A tibble: 3 x 5
##   group      group_color genera      species      genome_size
##   <chr>      <chr>      <chr>      <chr>          <dbl>
## 1 Green algae #8f9a9b    Chlamydomonas reinhardtii    110
## 2 Green algae #8f9a9b    Volvox      carteri                138
## 3 Bryophytes  #008933    Physcomitrella patens        472
```

# Unir columnas con `unite()`

- Podemos unir las columnas de nuevo con `unite()`:

```
plants_separated %>%  
  unite("species", c("genera", "species"), sep = " ", remove = TRUE)
```

```
## # A tibble: 35 x 4  
##   group      group_color species      genome_size  
##   <chr>      <chr>      <chr>      <dbl>  
## 1 Green algae #8f9a9b Chlamydomonas reinhard... 110  
## 2 Green algae #8f9a9b Volvox carteri 138  
## 3 Bryophytes #008933 Physcomitrella patens 472  
## 4 Bryophytes #008933 Selaginella moellendor... 100  
## 5 Basal Angiosper... #a52a2a Amborella trichopoda 870  
## 6 Magnoliids #ff8330 Cinnamomum kanehirae 823  
## 7 Monocots #00d958 Zostera marina 202  
## 8 Monocots #00d958 Lemna minor 481  
## 9 Monocots #00d958 Spirodela polyrhiza 158  
## 10 Monocots #00d958 Xerophyta viscosa 295  
## # ... with 25 more rows
```

# dplyr

*filter and select, arrange, mutate, group\_by, summarise,  
join*

# Subset con `filter()` y `select()`

- dplyr tiene dos funciones que pueden ayudarte a filtrar filas y seleccionar columnas como si estuvieras haciendo un subset, pero de manera más clara y versátil.
- Carguemos los datos crudos del dataset de palmer penguins:

```
penguins_filename <- paste0("https://raw.githubusercontent.com/",  
                             "rfordatascience/tidytuesday/master/data/",  
                             "2020/2020-07-28/penguins_raw.csv")  
penguins_raw <- read_csv(penguins_filename)  
penguins_raw %>% head(5)
```

```
## # A tibble: 5 x 17  
##   studyName `Sample Number` Species Region Island Stage  
##   <chr>          <dbl> <chr>   <chr>  <chr>  <chr>  
## 1 PAL0708          1 Adelie... Anvers Torge... Adul...  
## 2 PAL0708          2 Adelie... Anvers Torge... Adul...  
## 3 PAL0708          3 Adelie... Anvers Torge... Adul...  
## 4 PAL0708          4 Adelie... Anvers Torge... Adul...  
## 5 PAL0708          5 Adelie... Anvers Torge... Adul...  
## # ... with 11 more variables: `Individual ID` <chr>, `Clutch  
## #   Completion` <chr>, `Date Egg` <date>, `Culmen Length  
## #   (mm)` <dbl>, `Culmen Depth (mm)` <dbl>, `Flipper Length  
## #   (mm)` <dbl>, `Body Mass (g)` <dbl>, Sex <chr>, `Delta 15 N  
## #   (o/oo)` <dbl>, `Delta 13 C (o/oo)` <dbl>, Comments <chr>
```



- El dataset contiene varias columnas que no nos serían útiles y en la columna "Sex" hay algunos valores con NA. Filtremos esos valores y seleccionemos solo las columnas que necesitamos.
- Con base R podría ser:

```
select_columns <- c("Sample Number", "Species", "Island",  
                    "Date Egg", "Culmen Length (mm)",  
                    "Culmen Depth (mm)", "Flipper Length (mm)",  
                    "Body Mass (g)", "Sex")  
penguins1 <- penguins_raw[!is.na(penguins_raw$Sex), select_columns]  
penguins1 %>% head(3)
```

```
## # A tibble: 3 x 9  
##   `Sample Number` Species Island `Date Egg` `Culmen Length ...  
##           <dbl> <chr>   <chr>   <date>           <dbl>  
## 1             1 Adelie... Torge... 2007-11-11           39.1  
## 2             2 Adelie... Torge... 2007-11-11           39.5  
## 3             3 Adelie... Torge... 2007-11-16           40.3  
## # ... with 4 more variables: `Culmen Depth (mm)` <dbl>, `Flipper  
## #   Length (mm)` <dbl>, `Body Mass (g)` <dbl>, Sex <chr>
```

- Con dplyr podría ser:

```
penguins2 <- penguins_raw %>%
  filter(!is.na(Sex)) %>%
  select("Sample Number", "Species", "Island", "Date Egg",
         matches("(\\(mm\\))|\\(g\\)", "Sex") #tidyselect
penguins2 %>% head(3)
```

```
## # A tibble: 3 x 9
##   `Sample Number` Species Island `Date Egg` `Culmen Length ...
##           <dbl> <chr>   <chr>   <date>           <dbl>
## 1             1 Adelie... Torge... 2007-11-11           39.1
## 2             2 Adelie... Torge... 2007-11-11           39.5
## 3             3 Adelie... Torge... 2007-11-16           40.3
## # ... with 4 more variables: `Culmen Depth (mm)` <dbl>, `Flipper
## #   Length (mm)` <dbl>, `Body Mass (g)` <dbl>, Sex <chr>
```

- Con dplyr también puede ser fácil renombrar:

```
penguins <- penguins2 %>% rename(
  Sample = "Sample Number", Date = "Date Egg",
  Culm_len = "Culmen Length (mm)", Culm_dep = "Culmen Depth (mm)",
  Flip_len = "Flipper Length (mm)", Mass = "Body Mass (g)")
names(penguins)
```

```
## [1] "Sample" "Species" "Island" "Date" "Culm_len"
## [6] "Culm_dep" "Flip_len" "Mass" "Sex"
```

# Reordena las filas con `arrange()`

- Podemos invertir el orden de las muestras con:

```
penguins %>% arrange(desc(Sample)) %>% head(3)
```

```
## # A tibble: 3 x 9
##   Sample Species Island Date       Culm_len Culm_dep Flip_len  Mass
##   <dbl> <chr>   <chr>   <date>         <dbl>    <dbl>    <dbl> <dbl>
## 1    152 Adelie... Dream  2009-11-17     41.5     18.5     201  4000
## 2    151 Adelie... Dream  2009-11-17     36      17.1     187  3700
## 3    150 Adelie... Dream  2009-11-17     37.8     18.1     193  3750
## # ... with 1 more variable: Sex <chr>
```

- Podemos ordenar de acuerdo a varias variables con:

```
penguins_reordered <- penguins %>%
  arrange(Sex, Culm_len)
penguins_reordered %>% head(3)
```

```
## # A tibble: 3 x 9
##   Sample Species Island Date       Culm_len Culm_dep Flip_len  Mass
##   <dbl> <chr>   <chr>   <date>         <dbl>    <dbl>    <dbl> <dbl>
## 1    143 Adelie... Dream  2009-11-16     32.1     15.5     188  3050
## 2     99 Adelie... Dream  2008-11-10     33.1     16.1     178  2900
## 3     71 Adelie... Torge... 2008-11-14     33.5      19      190  3600
## # ... with 1 more variable: Sex <chr>
```

# Modifica o crea nuevas variables con `mutate()`

- `mutate()` puede utilizarse para modificar o crear varias variables a la vez:

```
penguins <- penguins %>%
  separate(Species, c("Common_name", "Species"),
           sep = " \\(", remove = TRUE) %>%
  mutate(
    Common_name = str_remove(Common_name, "(P|p)enguin"),
    Species = str_remove(Species, "\\)$"),
    Sex = ifelse(Sex == "FEMALE", "F", "M"),
    Year = as.numeric(format(Date, '%Y'))
  ) %>%
  select("Sample", "Common_name", "Species", "Island", "Year",
        everything(), -Date)
penguins %>% head(5)
```

```
## # A tibble: 5 x 10
##   Sample Common_name Species Island  Year Culm_len Culm_dep
##   <dbl> <chr>      <chr> <chr> <dbl>   <dbl>   <dbl>
## 1     1 "Adelie "   Pygosc... Torge... 2007    39.1    18.7
## 2     2 "Adelie "   Pygosc... Torge... 2007    39.5    17.4
## 3     3 "Adelie "   Pygosc... Torge... 2007    40.3     18
## 4     5 "Adelie "   Pygosc... Torge... 2007    36.7    19.3
## 5     6 "Adelie "   Pygosc... Torge... 2007    39.3    20.6
## # ... with 3 more variables: Flip_len <dbl>, Mass <dbl>, Sex <chr>
```

# group\_by y summarise()

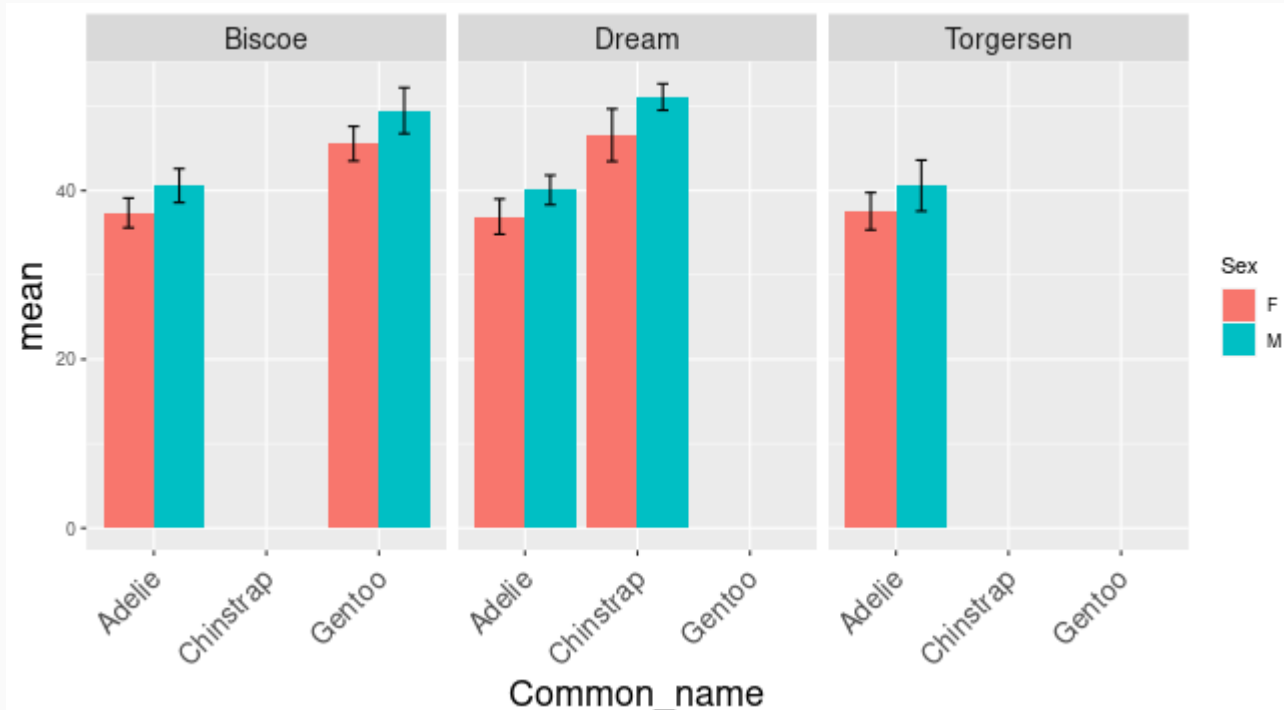
- Podemos crear resúmenes de nuestros datos (media, desviación estándar, etc.) por grupos combinando `group_by()` y `summarise()`:

```
penguins_summary <- penguins %>%  
  group_by(Island, Common_name, Sex) %>%  
  summarise(  
    mean = mean(Culm_len, na.rm = TRUE),  
    sd = sd(Culm_len, na.rm = TRUE),  
    n = n()  
  ) %>%  
  ungroup() # No olvidar ungroup()  
penguins_summary %>% head(5)
```

```
## # A tibble: 5 x 6  
##   Island Common_name Sex    mean    sd     n  
##   <chr>   <chr>      <chr> <dbl> <dbl> <int>  
## 1 Biscoe "Adelie  "    F    37.4  1.76    22  
## 2 Biscoe "Adelie  "    M    40.6  2.01    22  
## 3 Biscoe "Gentoo  "    F    45.6  2.05    58  
## 4 Biscoe "Gentoo  "    M    49.5  2.72    61  
## 5 Dream  "Adelie  "    F    36.9  2.09    27
```

- Ahora podemos graficar los datos fácilmente con ggplot2:

```
penguins_summary %>%  
  ggplot(aes(Common_name, mean, fill = Sex)) +  
  geom_col(position = "dodge") +  
  geom_errorbar(aes(ymin = mean - sd, ymax = mean + sd),  
                width = 0.2, position = position_dodge(0.9)) +  
  facet_wrap(~Island) +  
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 14),  
        axis.title = element_text(size = 18),  
        strip.text = element_text(size = 14))
```



# count()

- Si únicamente queremos contar los elementos por grupo, podemos usar `count()`.
- Es un shortcut de `group_by() %>% summarise(n = n()) %>% ungroup()`:

```
penguins %>%  
  count(Island, Common_name, Sex)
```

```
## # A tibble: 10 x 4  
##   Island    Common_name Sex      n  
##   <chr>    <chr>      <chr> <int>  
## 1 Biscoe  "Adelie "    F      22  
## 2 Biscoe  "Adelie "    M      22  
## 3 Biscoe  "Gentoo "    F      58  
## 4 Biscoe  "Gentoo "    M      61  
## 5 Dream   "Adelie "    F      27  
## 6 Dream   "Adelie "    M      28  
## 7 Dream   "Chinstrap " F      34  
## 8 Dream   "Chinstrap " M      34  
## 9 Torgersen "Adelie "    F      24  
## 10 Torgersen "Adelie "    M      23
```

# Uniendo datasets con `*_join()`

- A veces tenemos dos datasets con diferente tipo de información pero relacionados por una o más variables. Por ejemplo, resultados de una búsqueda con ncbi-blast:

```
blast_output <- readxl::read_excel("datasets.xlsx", sheet = "BLAST")
head(blast_output, 2)
```

```
## # A tibble: 2 x 4
##   subject      identity    evalue score
##   <chr>         <dbl>    <dbl> <dbl>
## 1 ath_AT5G47130    100  3.85e-135  381
## 2 AlyrAL7G37010    59.4 1.26e- 71  223
```

- ...y datos de las especies (en este caso plantas) de las secuencias:

```
species_data <- readxl::read_excel("datasets.xlsx", sheet = "PLANTS2")
head(species_data, 2)
```

```
## # A tibble: 2 x 4
##   sp    group      group_color species
##   <chr> <chr>         <chr>         <chr>
## 1 crei  Green algae #8f9a9b Chlamydomonas reinhardtii
## 2 vcar  Green algae #8f9a9b Volvox carteri
```



- Antes de unir, el resultado de blast necesita unas modificaciones:

## 1) Remover duplicados:

```
blast_output %>%
  count(subject) %>%
  arrange(desc(n)) %>% head(3)
```

```
## # A tibble: 3 x 2
##   subject      n
##   <chr>      <int>
## 1 spe_01g035900.1    4
## 2 ach_Achn067781    2
## 3 cchi_rna34811      2
```

```
blast_output <- blast_output %>%
  group_by(subject) %>%
  slice_max(identity) %>%
  ungroup()
head(blast_output, 3)
```

```
## # A tibble: 3 x 4
##   subject      identity  evalue score
##   <chr>      <dbl>    <dbl> <dbl>
## 1 aal_AALBA5B1011210P1    45.6 7.41e- 9  56.2
## 2 aal_AALBA5B1054303P1    46.9 1.66e-51  172
## 3 aar_AA147G00007        51.5 8.68e-32  118
```

```
blast_output %>%
  count(subject) %>%
  arrange(desc(n)) %>% head(3)
```

```
## # A tibble: 3 x 2
##   subject          n
##   <chr>          <int>
## 1 aal_AALBA5B1011210P1      1
## 2 aal_AALBA5B1054303P1      1
## 3 aar_AA147G00007           1
```

2) Crear un identificador único para unir con la columna "sp" de species\_data:

```
blast_output <- blast_output %>%
  mutate(id = str_extract(subject, "^....") %>% str_remove("_$")) %>%
  select(id, everything())
head(blast_output, 3)
```

```
## # A tibble: 3 x 5
##   id      subject          identity  evalue score
##   <chr> <chr>          <dbl>    <dbl> <dbl>
## 1 aal    aal_AALBA5B1011210P1    45.6 7.41e- 9  56.2
## 2 aal    aal_AALBA5B1054303P1    46.9 1.66e-51  172
## 3 aar    aar_AA147G00007         51.5 8.68e-32  118
```

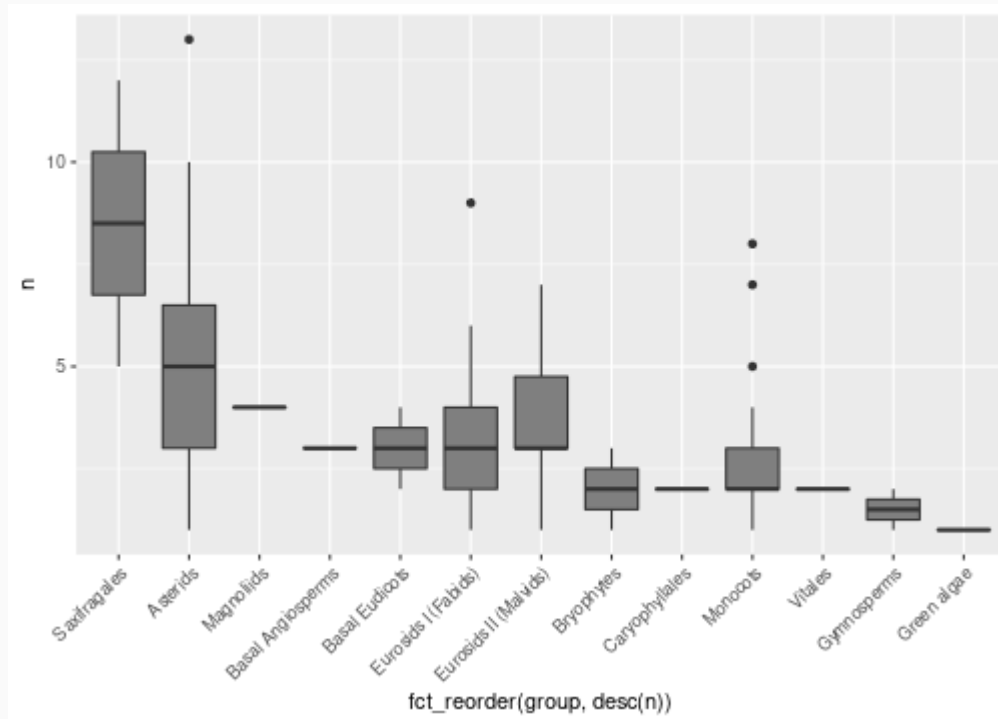
- Ahora podemos añadir los datos de las especies a las secuencias que encontramos por blast:

```
joined_data <- left_join(blast_output, species_data,
                        by = c("id" = "sp"))
joined_data
```

```
## # A tibble: 470 x 8
##   id    subject identity  evalue score group group_color species
##   <chr> <chr>      <dbl>    <dbl> <dbl> <chr> <chr>      <chr>
## 1 aal    aal_AAL...   45.6 7.41e- 9  56.2 Gymn... #b5b35c    Abies ...
## 2 aal    aal_AAL...   46.9 1.66e-51  172  Gymn... #b5b35c    Abies ...
## 3 aar    aar_AA1...   51.5 8.68e-32  118  Euro... #e80060    Aethio...
## 4 aar    aar_AA1...   58.2 1.51e-63  202  Euro... #e80060    Aethio...
## 5 aar    aar_AA5...   51.3 1.06e-52  176  Euro... #e80060    Aethio...
## 6 ach    ach_Ach...   55.7 1.59e-22  100  Aste... #00ccb4    Actini...
## 7 ach    ach_Ach...   48.1 6.64e-24  102  Aste... #00ccb4    Actini...
## 8 ach    ach_Ach...   26.1 1.70e+ 0   35.8 Aste... #00ccb4    Actini...
## 9 ach    ach_Ach...   57.6 4.94e-43  148  Aste... #00ccb4    Actini...
## 10 ach   ach_Ach...   58.2 6.97e-25  104  Aste... #00ccb4    Actini...
## # ... with 460 more rows
```

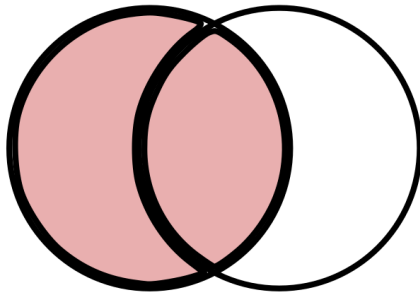
- ... y podemos ver cuantas secuencias obtuvimos por especie o grupo:

```
joined_data %>%  
  count(group, species) %>%  
  ggplot(aes(fct_reorder(group, desc(n)), n)) +  
  geom_boxplot(fill = "gray50") +  
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

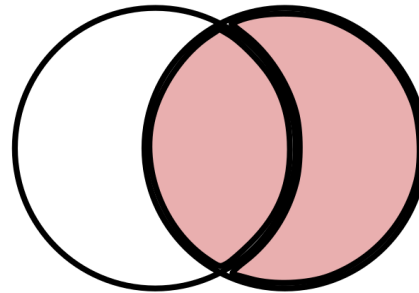


## Hay otros tipos de join

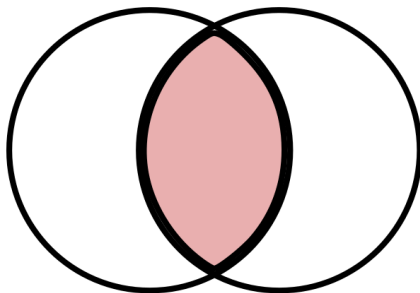
`left_join()`



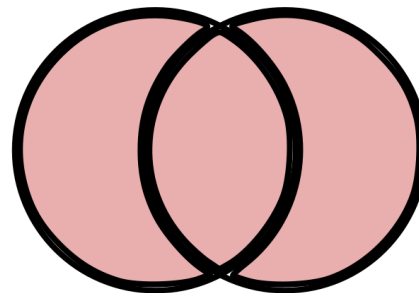
`right_join()`



`inner_join()`



`full_join()`

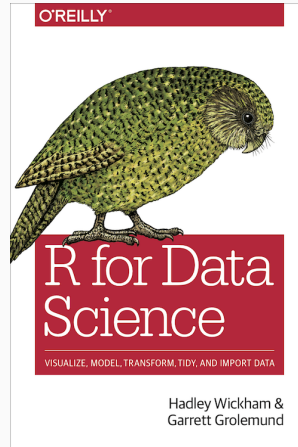


# Actividades recomendadas

- ¿Cómo se utilizan las funciones `complete()` y `fill()` del paquete `tidyr` para lidiar con NAs?
- Investiga sobre **grouped mutates**.
- Investiga acerca **tidyselect**.
- Compara `case_when()` (`dplyr`) con `ifelse()` (`base`).
- Investiga `bind_rows()` y `bind_cols()`.
- Investiga sobre el paquete **stringr** y expresiones regulares en R para manipular datos de tipo *character*.
- Practica con los diferentes tipos de `*join_()`.

# Bibliografía recomendada

R for data Science - Hadley Wickham



Capítulos 5, 12 y 13

<https://r4ds.had.co.nz/>