Clase 5

Ordenar, unir, exportar

Miriam Lerma

Marzo 2021

1. Intro

Funciones del tidyverse.
Para manipular data frames:

- filter
- mutate
- summarise
- unique
- drop_na
- Unir data frames
- Exportar data frames

Ustedes

- Conocimientos básicos de R (saben abrirlo, cargar paquetes y datos).
- Quieren manipular sus data frames.
- Quieren exportar su nuevo data frame.

Créditos

-Material basado en el libro:
R4DS, editado por Riva Quiroga

-Y materiales de RLadies
Zero to Hero

-Presentaciones de tidyverse:
María Paula Caldas
RLadiesBuenosAires

-Imágenes adicionales
Portada por Jess Bailey
Unsplash

Ordenar

Datos pinguinos.

🐧

Si van a hacer los ejercicios a la par, cargen sus datos.

```
library(datos)
Pingus<-pinguinos
```

Otra opcion es usar read_csv desde su computador.

```
library(tidyverse)
```

1.1. Tidyverse

Tidyverse engloba varios paquetes, la mayoria para especificamente para inspeccionar y manipular tus datos.

1.2. Pipe

Vamos a usar mucho el **pipe** un argumento que se usa para encadenar funciones.

En su teclado: strg+alt+M

```
%>%
```

1.3. Funciones

El paquete dplyr nos da una serie de herramientas para **manipular** datos

Las principales funciones, o **verbos** de dplyr, son:

- **count()** para contar
- **select()**, para seleccionar columnas
- **filter()**, para filtrar filas
- **mutate()**, para crear o modificar columnas
- **summarise()**, para resumir información de las columnas

count y select las vimos en **la clase pasada**

1.4. filter()

Podemos filtrar columnas de acuerdo a valores que nos interesen.

Pero para esto hay que conocer algunos operadores lógicos:

- El símbolo `==` es para decir 'igual a'
- El símbolo `!=` es para decir 'distinto a'
- El símbolo `>` es para decir 'mayor que'
- El símbolo `<` es para decir 'menor que'
- El símbolo `>=` es para decir 'mayor o igual que'
- El símbolo `<=` es para decir 'menor o igual que'
- El símbolo `&` es para decir 'y'
- El símbolo `|` es para decir 'o'

1.4. filter(==)

```
Pingus %>%
  filter(sexo == 'hembra')
```

Nota: las variables son sin comillas y las categorías en comillas.

Vayan a su environment... y revisen... se cambio mi tabla?

No. Hay que crear un nuevo objeto.

```
PingusHembras<-Pingus %>%
  filter(sexo == 'hembra')
```

Miren su environment.

1.4. filter(<=)

```
Pingus %>%
  filter(largo_pico_mm <= 39.1)
```

```
## # A tibble: 83 x 8
##    especie isla  largo_pico_mm alto_pico_mm largo_aleta_mm masa_corporal_g
##    <fct>   <fct>         <dbl>        <dbl>           <int>           <int>
##  1 Adelia  Torg~          39.1         18.7             181            3750
##  2 Adelia  Torg~          36.7         19.3             193            3450
##  3 Adelia  Torg~          38.9         17.8             181            3625
##  4 Adelia  Torg~          34.1         18.1             193            3475
##  5 Adelia  Torg~          37.8         17.1             186            3300
##  6 Adelia  Torg~          37.8         17.3             180            3700
##  7 Adelia  Torg~          38.6         21.2             191            3800
##  8 Adelia  Torg~          34.6         21.1             198            4400
##  9 Adelia  Torg~          36.6         17.8             185            3700
## 10 Adelia  Torg~          38.7         19               195            3450
## # ... with 73 more rows, and 1 more variable: anio <int>
```

1.4. filter(>=)

```
Pingus %>%
  filter(largo_pico_mm >= 39.1)
```

```
## # A tibble: 260 x 8
##    especie isla  largo_pico_mm alto_pico_mm largo_aleta_mm masa_corporal_g
##    <fct>   <fct>         <dbl>        <dbl>           <int>           <int>
##  1 Adelia  Torg~          39.1         18.7             181            3750
##  2 Adelia  Torg~          39.5         17.4             186            3800
##  3 Adelia  Torg~          40.3         18               195            3250
##  4 Adelia  Torg~          39.3         20.6             190            3650
##  5 Adelia  Torg~          39.2         19.6             195            4675
##  6 Adelia  Torg~          42           20.2             190            4250
##  7 Adelia  Torg~          41.1         17.6             182            3200
##  8 Adelia  Torg~          42.5         20.7             197            4500
##  9 Adelia  Torg~          46           21.5             194            4200
## 10 Adelia  Bisc~          40.6         18.6             183            3550
## # ... with 250 more rows, and 1 more variable: anio <int>
```

1.4. filter(&)

```
Pingus %>%
  filter(isla == 'Biscoe' & especie =='Adelia')
```

```
## # A tibble: 44 x 8
##    especie isla  largo_pico_mm alto_pico_mm largo_aleta_mm masa_corporal_g
##    <fct>   <fct>         <dbl>        <dbl>           <int>           <int>
##  1 Adelia  Bisc~          37.8         18.3             174            3400
##  2 Adelia  Bisc~          37.7         18.7             180            3600
##  3 Adelia  Bisc~          35.9         19.2             189            3800
##  4 Adelia  Bisc~          38.2         18.1             185            3950
##  5 Adelia  Bisc~          38.8         17.2             180            3800
##  6 Adelia  Bisc~          35.3         18.9             187            3800
##  7 Adelia  Bisc~          40.6         18.6             183            3550
##  8 Adelia  Bisc~          40.5         17.9             187            3200
##  9 Adelia  Bisc~          37.9         18.6             172            3150
## 10 Adelia  Bisc~          40.5         18.9             180            3950
## # ... with 34 more rows, and 1 more variable: anio <int>
```

1.5. mutate()

Es para crear o modificar columnas.
Podemos crear una columna a partir de los valores de otra.

```
Pingus<-Pingus %>%
  mutate(kilos = masa_corporal_g / 1000)
```

1.6. lubridate

Facilita el trabajo con fechas y horas, ya que te permite decirle a R que una cadena de caracteres, es tiempo y horas.

```
library(lubridate)
```

```
ymd_hms("2010-12-13 15:30:30")
```

Ademas, te permite **extraer** componentes de fechas y horas.

```
ymd_hms("2010-12-13 15:30:30") %>% month()
```

```
## [1] 12
```

1.6. ymd_hms

Datos de hora inventados, solo nos interesa ver la hora, los minutos y los segundos.

```
DatosPorHora<-c("2010-12-13 13:30:30","2010-12-13 14:30:30","2010-12-13
Horas<-data.frame(DatosPorHora=DatosPorHora)
```

Usando **mutate** podemos separar esa informacion y extraer la que nos interesa.

```
Horas %>%
  mutate(
    hora = hour(DatosPorHora),
    minuto = minute(DatosPorHora),
    segundo = second(DatosPorHora)
  )
```

1.7. group_by() y summarise()

summarise() para resumir información de las columnas

```
Pingus %>%
  group_by(anio) %>%
  summarise(PromedioLargoPico=mean(largo_pico_mm))
```

```
## # A tibble: 3 x 2
##    anio PromedioLargoPico
##   <int>             <dbl>
## 1  2007              NA
## 2  2008              43.5
## 3  2009              NA
```

1.8. top_n()

Obserservaciones maximas que le pidamos de una variable.

```
Pingus %>%
  top_n(2,largo_pico_mm)
```

```
## # A tibble: 2 x 9
##   especie isla   largo_pico_mm alto_pico_mm largo_aleta_mm masa_corporal_g
##   <fct>   <fct>          <dbl>        <dbl>          <int>           <int>
## 1 Papúa   Biscoe          59.6         17             230            6050 
## 2 Barbijo Dream           58           17.8           181            3700 
## # ... with 2 more variables: anio <int>, kilos <dbl>
```

1.9. unique() o distinct()

Valores únicos en esa columna.

Hay varias maneras de ver este tipo de informacion:

```
unique(Pingus$especie)
```

```
## [1] Adelia  Papúa   Barbijo
## Levels: Adelia Barbijo Papúa
```

```
pinguinos %>%
  distinct(especie)
```

```
## # A tibble: 3 x 1
##   especie
##   <fct>
## 1 Adelia
## 2 Papúa
## 3 Barbijo
```

1.9. unique() o distinct()

Valores únicos en esa columna.

```
Pingus %>%
  distinct(isla)
```

```
## # A tibble: 3 x 1
##   isla
##   <fct>
## 1 Torgersen
## 2 Biscoe
## 3 Dream
```

1.10. drop_na

Hay que tener cuidado con las nas para varias operaciones.

Para filtrar nas en pipe:

```
Pingus %>%
  drop_na(largo_pico_mm)
```

Otra opcion:

```
PingusSinNA <- Pingus %>%
  filter(!is.na(largo_pico_mm))
```

Miren que ahora tenemos menos observaciones en pingus sin NAs.

Ejercicios

Usar funciones de tidyverse

- count
- distinct
- group_by
- drop_na
- summarise
- mutate
- select
- filter

1.11. Ejercicios

```
library(datos)
Pingus<-pinguinos
```

Cuantas observaciones tenemos?

```
pinguinos %>%
  count()
```

Cuantas observaciones por isla?

```
Pingus %>%
  distinct(isla)
```

Cuantas observaciones por sexo e isla?

```
Pingus %>%
  group_by(sexo, isla) %>%
  count()
```

1.11. Ejercicios

Cual es el promedio de peso entre hembras y machos por especie? y en base a cuantas observaciones?

```
Pingus %>%
  group_by(especie,sexo) %>%
  drop_na(masa_corporal_g)%>%
  summarise(mean = mean(masa_corporal_g), n = n())
```

Cambia el peso de los pinguinos a kilos.

```
Pingus<-Pingus %>%
  mutate(kilos = masa_corporal_g / 1000)
```

1.12. Ejercicios solos

- Crea un objeto con solo dos columnas, las de sexo y anio.

```
#_____ <-Pingus %>%
#  select(____,_____)
```

- Crea un objeto con las diferentes especies de pinguinos que hay en el data.frame y a que especies corresponden.

```
#______ <-Pingus %>%
#  distinct(_____)
```

- Crea un objeto con solo machos.

```
#______ <-Pingus %>%
#  filter(_____ == '______')
```

- Crea un objeto con solo hembras que pesen (masa_corporal_g) igual o mas de 3800 g.

```
#______ <-Pingus %>%
#  filter(_____ == '______')%>%
#  filter(______ >= ______)
```

Unir

2.1. left_join()

Datos inventados de 10 individuos, pesos de los individuos y la concentracion de pesticida.

```
ID<-c("ID01","ID02","ID03","ID04","ID05",
      "ID06","ID07","ID08","ID09","ID10")
Pesos<-c(1.5,2.0,3.5,4.1,2.6,3.7,8.9,2.5,6.3,1.0)
Pesticidas<-c(10,20,35,1,6,3,8,2,3,1)
```

2.1. left_join()

Muchas veces tengo datos de laboratorio por un lado...

```
Laboratorio<-data.frame(ID,Pesticidas)
head(Laboratorio)
```

```
##     ID Pesticidas
## 1 ID01         10
## 2 ID02         20
## 3 ID03         35
## 4 ID04          1
## 5 ID05          6
## 6 ID06          3
```

... y de campo por otro.

```
Libreta <- data.frame(ID,Pesos)
head(Libreta)
```

```
##     ID Pesos
## 1 ID01   1.5
## 2 ID02   2.0
## 3 ID03   3.5
```

2.1. left_join()

Pero los necesitamos juntos!

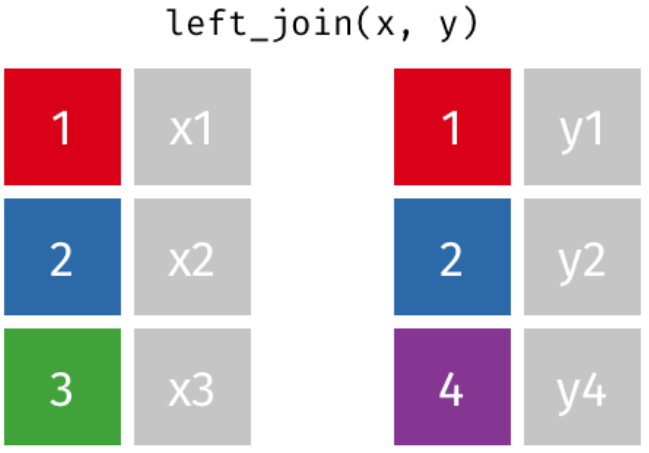
- **left_join()** me permite unirlos. Pero es importante tener un ID para poder unir las tablas.

```
DatosJuntos<-left_join(Libreta, Laboratorio,
                       by = "ID")
DatosJuntos
```

```
##       ID Pesos Pesticidas
## 1   ID01   1.5         10
## 2   ID02   2.0         20
## 3   ID03   3.5         35
## 4   ID04   4.1          1
## 5   ID05   2.6          6
## 6   ID06   3.7          3
## 7   ID07   8.9          8
## 8   ID08   2.5          2
## 9   ID09   6.3          3
## 10 ID10   1.0          1
```

2.1. left_join()

Lo que hace left_join() es usar ese ID para saber como unir la tabla.



Fuente: tidyexplain

Otras opciones de unión ilustradas

2.2. pivot_longer

A veces los datos se toman de manera **no ordenada** en el sentido de que las observaciones estan en cada columna y no en cada fila.

Esto es normal, particularmente en campo o en el laboratorio y tenemos una libreta con espacio limitado.

Entonces hay que re-organizarlos.

pais	anio	casos
Afganistán	1999	745
Afganistán	2000	2666
Brasil	1999	37737
Brasil	2000	80488
China	1999	212258
China	2000	213766

pais	1999	2000
Afganistán	745	2666
Brasil	37737	80488
China	212258	213766

Tabla 4

2.2. pivot_longer

Datos inventados de 5 especies y sus riquezas en tres anios diferentes.

```
Especies<-c('Especie1','Especie2','Especie3','Especie4','Especie5')
Anio2010<-c(5,4,5,6,7)
Anio2011<-c(3,2,1,9,4)
Anio2012<-c(6,2,3,7,8)
```

Creamos un nuevo data frame usando esos datos.

```
EspecieAnio<-data.frame(Especie=Especies,
                        Anio2010=Anio2010,Anio2011=Anio2011,Anio2012)
```

Quedar algo así

```
head(EspecieAnio)
```

```
##    Especie Anio2010 Anio2011 Anio2012
## 1 Especie1        5        3        6
## 2 Especie2        4        2        2
## 3 Especie3        5        1        3
## 4 Especie4        6        9        7
```

2.2. pivot_longer

El argumento **pivot_longer** te permite reorganizarlas.

```
Especies_largo <- EspecieAnio %>%
  pivot_longer(c(Anio2010, Anio2011, Anio2012),
    names_to = "Anio",
    values_to = "Riqueza" )
Especies_largo
```

```
## # A tibble: 15 x 3
##    Especie  Anio     Riqueza
##    <chr>    <chr>      <dbl>
##  1 Especie1 Anio2010       5
##  2 Especie1 Anio2011       3
##  3 Especie1 Anio2012       6
##  4 Especie2 Anio2010       4
##  5 Especie2 Anio2011       2
##  6 Especie2 Anio2012       2
##  7 Especie3 Anio2010       5
##  8 Especie3 Anio2011       1
##  9 Especie3 Anio2012       3
## 10 Especie4 Anio2010       6
## 11 Especie4 Anio2011       9
```

2.3. pivot_wider

Si por alguna razón quiero tenerlos separados, puedo también extraerlos a lo ancho.

pais	anio	tipo	casos
Afganistán	1999	casos	745
Afganistán	1999	población	19987071
Afganistán	2000	casos	2666
Afganistán	2000	población	20595360
Brasil	1999	casos	37737
Brasil	1999	población	172006362
Brasil	2000	casos	80488
Brasil	2000	población	174504898
China	1999	casos	212258
China	1999	población	1272915272
China	2000	casos	213766
China	2000	población	1280428583

Tabla 2

pais	anio	casos	poblacion
Afganistán	1999	745	19987071
Afganistán	2000	2666	20595360
Brasil	1999	37737	172006362
Brasil	2000	80488	17504898
China	1999	212258	1272915272
China	2000	213766	1280428583

2.3. pivot_wider

Datos inventados.

```
Especies_ancho<-Especies_largo %>%
  pivot_wider(names_from = Anio,
              values_from = Riqueza)
Especies_ancho
```

```
## # A tibble: 5 x 4
##   Especie  Anio2010 Anio2011 Anio2012
##   <chr>       <dbl>    <dbl>    <dbl>
## 1 Especie1        5        3        6
## 2 Especie2        4        2        2
## 3 Especie3        5        1        3
## 4 Especie4        6        9        7
## 5 Especie5        7        4        8
```

2.4. Unite

Este argumento sirve para **unir** valores de dos columnas en una columna.

```
(Pingus<-Pingus %>%
  unite(col = especie_isla,
        c("especie", "isla"),
        sep = ": "))
```

```
## # A tibble: 344 x 7
##    especie_isla  largo_pico_mm alto_pico_mm largo_aleta_mm masa_corporal_g
##    <chr>                 <dbl>        <dbl>          <int>           <int>
##  1 Adelia: Torg~          39.1         18.7            181            3750
##  2 Adelia: Torg~          39.5         17.4            186            3800
##  3 Adelia: Torg~          40.3         18              195            3250
##  4 Adelia: Torg~          NA           NA               NA              NA
##  5 Adelia: Torg~          36.7         19.3            193            3450
##  6 Adelia: Torg~          39.3         20.6            190            3650
##  7 Adelia: Torg~          38.9         17.8            181            3625
##  8 Adelia: Torg~          39.2         19.6            195            4675
##  9 Adelia: Torg~          34.1         18.1            193            3475
## 10 Adelia: Torg~          42           20.2            190            4250
## # ... with 334 more rows, and 1 more variable: anio <int>
```

2.5. Separate

Este argumento sirve para **separar** valores de una columna en dos columnas.

```
Pingus %>%
  separate(col = especie_isla,
           into = c("especie", "isla"),
           sep = ":")
```

Ojo: borra las columnas originales, si no queremos que las borre debemos agregar **remove = FALSE.**

```
(Pingus<-Pingus %>%
  separate(col = especie_isla,
           into = c("especie", "isla"),
           sep = ":",
           remove = FALSE))
```

```
## # A tibble: 344 x 9
##    especie_isla     especie isla       largo_pico_mm alto_pico_mm largo_ale
##    <chr>            <chr>   <chr>              <dbl>        <dbl>
##  1 Adelia: Torgers~ Adelia  " Torgers~         39.1         18.7
##  2 Adelia: Torgers~ Adelia  " Torgers~         39.5         17.4
##  3 Adelia: Torgers~ Adelia  " Torgers~         40.3         18
```

2.6. Rename

Este argumento **cambia** el nombre de la columna.
El nuevo nombre se debe poner primero y el nombre anterior después.

```
Pingus %>%
  rename(peso = masa_corporal_g)
```

```
## # A tibble: 344 x 9
##    especie_isla   especie isla   largo_pico_mm alto_pico_mm largo_aleta_mm
##    <chr>          <chr>   <chr>          <dbl>        <dbl>          <int>
##  1 Adelia: Torge~ Adelia  " Tor~          39.1         18.7            181
##  2 Adelia: Torge~ Adelia  " Tor~          39.5         17.4            186
##  3 Adelia: Torge~ Adelia  " Tor~          40.3         18              195
##  4 Adelia: Torge~ Adelia  " Tor~          NA           NA               NA
##  5 Adelia: Torge~ Adelia  " Tor~          36.7         19.3            193
##  6 Adelia: Torge~ Adelia  " Tor~          39.3         20.6            190
##  7 Adelia: Torge~ Adelia  " Tor~          38.9         17.8            181
##  8 Adelia: Torge~ Adelia  " Tor~          39.2         19.6            195
##  9 Adelia: Torge~ Adelia  " Tor~          34.1         18.1            193
## 10 Adelia: Torge~ Adelia  " Tor~          42           20.2            190
## # ... with 334 more rows, and 2 more variables: sexo <fct>, anio <int>
```

2.6. Rename

Este argumento sirve para cambiar el nombre de varias columnas.
El nuevo nombre se pone primero y luego se pone el nombre anterior.

```
Pingus %>%
  rename(peso=masa_corporal_g,
         temporada=anio)
```

```
## # A tibble: 344 x 9
##    especie_isla   especie isla   largo_pico_mm alto_pico_mm largo_aleta_mm
##    <chr>          <chr>   <chr>          <dbl>        <dbl>          <int>
##  1 Adelia: Torge~ Adelia  " Tor~          39.1         18.7            181
##  2 Adelia: Torge~ Adelia  " Tor~          39.5         17.4            186
##  3 Adelia: Torge~ Adelia  " Tor~          40.3         18              195
##  4 Adelia: Torge~ Adelia  " Tor~          NA           NA               NA
##  5 Adelia: Torge~ Adelia  " Tor~          36.7         19.3            193
##  6 Adelia: Torge~ Adelia  " Tor~          39.3         20.6            190
##  7 Adelia: Torge~ Adelia  " Tor~          38.9         17.8            181
##  8 Adelia: Torge~ Adelia  " Tor~          39.2         19.6            195
##  9 Adelia: Torge~ Adelia  " Tor~          34.1         18.1            193
## 10 Adelia: Torge~ Adelia  " Tor~          42           20.2            190
## # ... with 334 more rows, and 2 more variables: sexo <fct>, temporada <int>
```

2.7. Relocate

Este argumento sirve para mover columnas al inicio del data frame.

```
Pingus %>%
  relocate(sexo, anio)
```

```
## # A tibble: 344 x 9
##    sexo    anio especie_isla      especie isla        largo_pico_mm alto_pi
##    <fct>  <int> <chr>             <chr>   <chr>               <dbl>
##  1 macho   2007 Adelia: Torgersen Adelia  " Torgerse~          39.1
##  2 hembra  2007 Adelia: Torgersen Adelia  " Torgerse~          39.5
##  3 hembra  2007 Adelia: Torgersen Adelia  " Torgerse~          40.3
##  4 <NA>    2007 Adelia: Torgersen Adelia  " Torgerse~          NA
##  5 hembra  2007 Adelia: Torgersen Adelia  " Torgerse~          36.7
##  6 macho   2007 Adelia: Torgersen Adelia  " Torgerse~          39.3
##  7 hembra  2007 Adelia: Torgersen Adelia  " Torgerse~          38.9
##  8 macho   2007 Adelia: Torgersen Adelia  " Torgerse~          39.2
##  9 <NA>    2007 Adelia: Torgersen Adelia  " Torgerse~          34.1
## 10 <NA>    2007 Adelia: Torgersen Adelia  " Torgerse~          42
## # ... with 334 more rows, and 2 more variables: largo_aleta_mm <int>,
## #   masa_corporal_g <int>
```

2.8 ggplot()

Noten que se pueden **encadenar** funciones y crear graficos.

```
Pingus %>%
  filter(sexo=='hembra') %>%
  ggplot(aes(x = largo_pico_mm)) +
  geom_histogram()
```

Ejercicios

Usar funciones de tidyverse

- left_join()
- pivot_longer()
- pivot_wider()

2.9. Ejercicios

left_join()

Datos inventados.

```
ID<-c("ID01","ID02","ID03","ID04","ID05",
      "ID06","ID07","ID08","ID09","ID10")
Pesos<-c(1.5,2.0,3.5,4.1,2.6,3.7,8.9,2.5,6.3,1.0)
Pesticidas<-c(10,20,35,1,6,3,8,2,3,1)
```

Datos del laboratorio.

```
Laboratorio<-data.frame(ID,Pesticidas)
```

Datos de campo.

```
Libreta <- data.frame(ID,Pesos)
```

DatosJuntos.

```
DatosJuntos<-left_join(Libreta, Laboratorio,
                       by = "ID")
```

2.9. Ejercicios

pivot_longer

Datos inventados de 5 especies y sus riquezas.

```
Especies<-c('Especie1','Especie2','Especie3','Especie4','Especie5')
Anio2010<-c(5,4,5,6,7)
Anio2011<-c(3,2,1,9,4)
Anio2012<-c(6,2,3,7,8)
```

Un nuevo data frame.

```
EspecieAnio<-data.frame(Especie=Especies,Anio2010=Anio2010,Anio2011=Anio
```

```
head(EspecieAnio)
```

Este argumento me permite reorganizarlas.

```
EspeciesLargo <- EspecieAnio %>%
  pivot_longer(c(Anio2010, Anio2011, Anio2012),
    names_to = "Anio",
```

Exportar

3. Exportar

Igual que los argumentos para importar tus datos (read_csv), cada uno tiene su contra parte para exportarlos.

- write_csv()
- write_csv2()
- write_tsv()
- write_delim()

Ejercicios

1- Define una carpeta.

```
library(here)
ResultsFolder<-here::here("Clase3 wrangling")
```

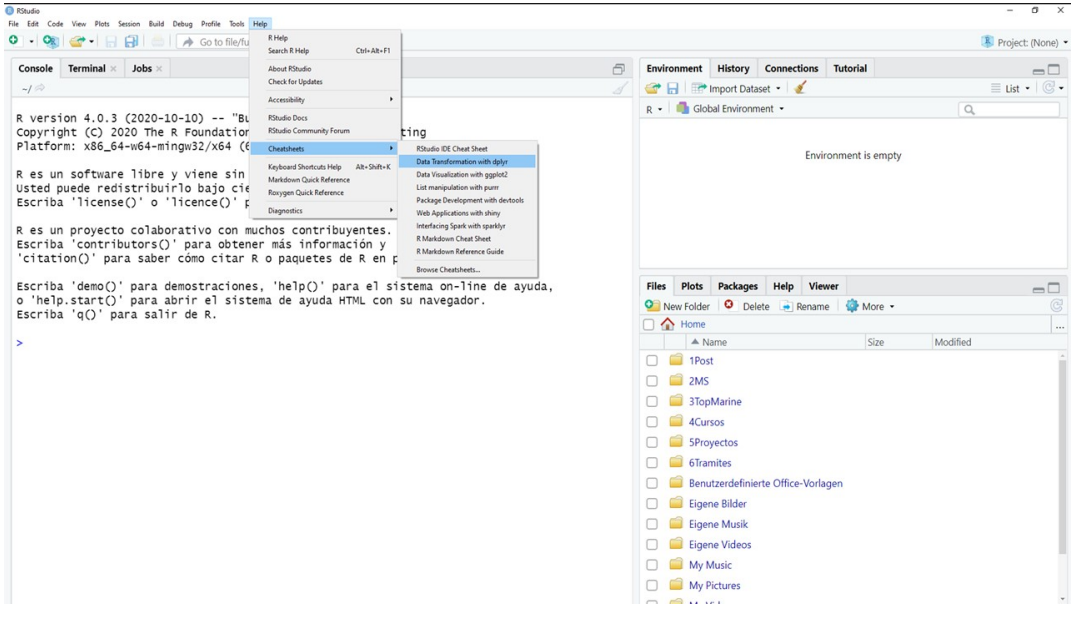
2- Exporta tu tabla.

```
write_csv(
  DatosJuntos,
  file =paste0(ResultsFolder,'/DatosJuntos.csv'))
```

4. Continuar aprendiendo

Usar CheatSheet, que son acordeones para tener la información mas a la mano.
Realizar ejercicios.

En R>



Recapitulando

Esta clase:

- Funciones de filter, mutate, summarise, unique, drop_na.
- Unir data frames.
- Exportar data frames

Siguiente clase:

- Trabajar por proyectos

Contacto

Para dudas, comentarios y sugerencias:
Escríbeme a miriamjlerma@gmail.com

Este material esta accesible
y se encuentra en mi github
y mi página www.miriam-lerma.com