

OPSO79-1-UCSH2021

Tipos de datos cuantitativos e introducción a dplyr en R. Bloque práctico.

27/08/2021

Nuestro mundo en datos (provación)

Our World in data

OurWorldInData es una publicación en-línea que presenta datos y resultados empíricos que muestran el cambio en las condiciones de vida en todo mundo.

Un estudiante de magíster en políticas públicas está desarrollando un [paquete en R](#) para descargar y visualizar directamente estos datos.

Como está en desarrollo aún no se encuentra en la CRAN. Si se quiere ocupar tiene que descargarse así:

```
install.packages("devtools")  
devtools::install_github("piersyork/owidR")  
library(owidR)
```

Más adelante entenderemos bien. Por ahora veamos lo que nos permite hacer en [R Cloud](#).



Un poco más de R base

Insertar lenguaje R en .rmd

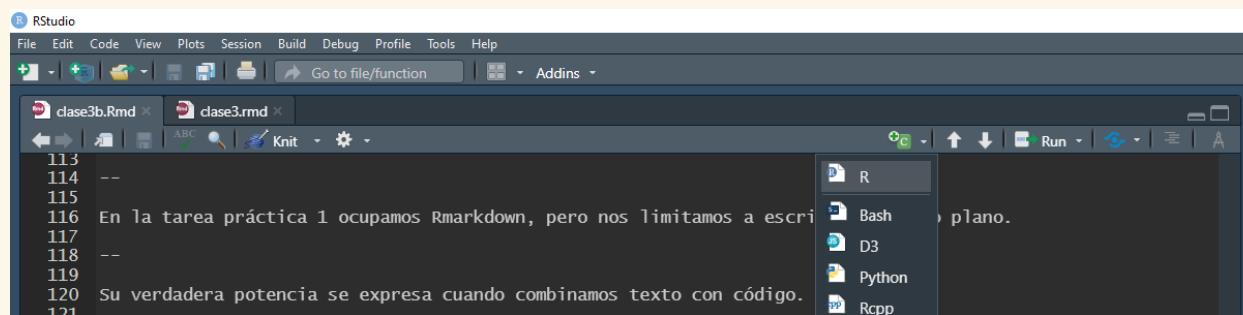
Hemos visto tres formas de interactuar con R:

- Consola.
- Script .R.
- Script .rmd o Rmarkdown.

En la tarea práctica 1 ocupamos Rmarkdown, pero nos limitamos a escribir en texto plano.

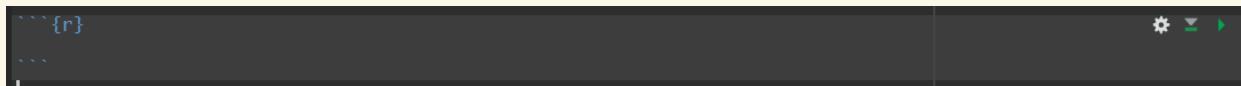
Su verdadera potencia se expresa cuando combinamos texto con código.

Para agregar un chunk o trozo de código de lenguaje R en RMarkdown:



Insertar lenguaje R en .rmd

Aparecerá lo siguiente:



En el espacio interno podemos escribir código R:

A screenshot of an RStudio interface showing an R Markdown file named "clase3b.Rmd". The code editor contains the following R code:

```
9
10 ## Guaguas
11
12 ```{r}
13 library(guaguas)
14 head(guaguas)
15 table(guaguas$sexo)
16
17
18 ...
19
```

The code results in the following output:

	F	M
523623	321777	

```
20
21 ```{r}
22 1 + 1
23
```

The output of the second code chunk is: [1] 2

Algunos ejemplos

Operadores lógicos

Los operadores lógicos son muy importantes para la programación.

R cuenta con operadores de comparación binaria.

```
x < y      # menor que  
x > y      # mayor que  
x <= y     # menor o igual que  
x >= y     # mayor o igual que  
x == y     # igual a  
x != y     # distinto a
```

@@: Nota que == permite comparar si dos valores son iguales. Ten cuidado de NO usar = que es interpretado como un operador de asignación (es como usar <-).

Operadores lógicos

Algunos ejemplos con números:

```
x <- c(1,2,5)
y <- c(4,4,3)
x == y
#> [1] FALSE FALSE FALSE
x != y
#> [1] TRUE TRUE TRUE
x < y
#> [1] TRUE TRUE FALSE
```

Otros operadores muy importantes son | (o) e & (y)

```
guaguas[guaguas$nombre=="Salvador" &
        (guaguas$anio==1972 | guaguas$anio==1979),]
```

```
## # A tibble: 2 x 5
##   anio nombre sexo     n  proporcion
##   <dbl> <chr>  <chr> <dbl>      <dbl>
## 1 1972 Salvador M     183  0.000576
## 2 1979 Salvador M      59  0.000218
```

Introducción a paquete dplyr

El paquete dplyr

Instalar y cargar como cualquier otro paquete:

```
install.packages("dplyr")
```

```
library(dplyr)
```



El paquete dplyr

Nos proporciona una "gramática" particular para manipular y aplicar funciones sobre bases de datos.

En términos de la ciencia de datos, es un paquete para **transformar** la data.

El paquete fue desarrollado por [Hadley Wickham](#) de RStudio.



Ocuparemos muchos de sus paquetes (ggplot, tidyverse, haven, readxl, httr, lubridate, etc.).

Manipulación básica

Revisaremos 5 verbos básicos de dplyr:

- `select()`
- `filter()`
- `arrange()`
- `rename()`
- `mutate()`

Lo que nos permiten hacer se puede hacer en R base, pero nos simplificarán la vida.

select()

Nos permite seleccionar de forma intuitiva las columnas que indiquemos.

```
select(data, columna)
```

Selecciona 1 columnas de la base guaguas:

```
select(guaguas,nombre)
```

```
## # A tibble: 845,400 x 1
##       nombre
##   <chr>
## 1 María
## 2 José
## 3 Juan
## 4 Luis
## 5 Rosa
## 6 Ana
## 7 Manuel
## 8 Olga
## 9 Carlos
## 10 Pedro
```

select()

Podemos seleccionar más de una columna agregandolas como argumento

```
select(guaguas,nombre,n)
```

```
## # A tibble: 845,400 x 2
##       nombre      n
##   <chr>    <dbl>
## 1 María     2130
## 2 José      984
## 3 Juan      636
## 4 Luis      631
## 5 Rosa      426
## 6 Ana       340
## 7 Manuel    326
## 8 Olga      289
## 9 Carlos    277
## 10 Pedro     269
## # ... with 845,390 more rows
```

select()

E incluso reordenar las columnas, ocupando sus nombres o sus posiciones

```
select(guaguas, 5:1)
```

```
## # A tibble: 845,400 x 5
##   proporcion     n sexo nombre  anio
##       <dbl> <dbl> <chr> <chr>  <dbl>
## 1     0.104    2130 F     María  1920
## 2     0.0483   984 M     José   1920
## 3     0.0312   636 M     Juan   1920
## 4     0.0310   631 M     Luis   1920
## 5     0.0209   426 F     Rosa   1920
## 6     0.0167   340 F     Ana    1920
## 7     0.0160   326 M     Manuel 1920
## 8     0.0142   289 F     Olga   1920
## 9     0.0136   277 M     Carlos  1920
## 10    0.0132   269 M     Pedro   1920
## # ... with 845,390 more rows
```

```
guaguas[,c(5:1)] # Lo mismo con R base
```

filter()

Nos permite seleccionar las filas que satisfacen nuestras condiciones.

Para eso utilizamos los operadores lógicos ya revisados (`>`, `<`, `>=`, `<=`, `==`, `!=`, `&`, `|`)

```
filter(data, condicion)
```

```
filter(guaguas, anio==1960 & nombre=="Fidel")
```

```
## # A tibble: 1 x 5
##   anio  nombre  sexo      n  proporcion
##   <dbl> <chr>    <chr> <dbl>        <dbl>
## 1  1960 Fidel    M       118     0.000448
```

En R base es bastante más tedioso:

```
guaguas[guaguas$anio==1960 & guaguas$nombre=="Fidel", ]
```

filter()

dplyr::filter()

KEEP ROWS THAT
satisfy
your CONDITIONS

keep rows from... this data... ONLY IF... type is "otter" AND site is "bay"
filter(df, type == "otter" & site == "bay")

The illustration features two cartoon characters. On the left, a yellow, round character with a smiling face and simple legs points towards a map of a coastal area with a blue bay labeled 'BAY'. On the right, a purple, irregularly shaped character with a question mark on its head stands next to a green, rounded character. Between them is a table with three columns: 'type', 'food', and 'site'. The rows show the following data:

type	food	site
otter	urchin	bay
Shark	seal	channel
otter	abalone	bay
otter	crab	wharf

Annotations with orange arrows and checkmarks indicate that the rows for 'otter' with 'urchin' and 'abalone' are selected, while the other rows are crossed out.

arrange()

Ordena las filas de una base de datos según una variable.

Dentro de la función hay que indicar por cuál variable, y sí el orden será ascendente o descendente.

Por defecto el orden es ascendente.

```
arrange(data, variable) # ascendente  
arrange(data, -variable) # descendente
```

Ordenar guaguas desde el nombre más popular al menos popular:

```
arrange(guaguas, -n)
```

```
## # A tibble: 2 x 5  
##   anio  nombre sexo      n  proporcion  
##   <dbl> <chr>  <chr> <dbl>        <dbl>  
## 1 1955 María    F     21448      0.0851  
## 2 1956 María    F     21013      0.0826
```

arrange()

Ordenar con R base es más complicado.

Orden ascendente

```
guaguas[order(guaguas[, "n"]), ]
```

Orden descendente

```
guaguas[order(guaguas[, "n"], decreasing = TRUE), ]
```

rename()

Permite cambiar el nombre de las variables en la data.

```
rename(data, nuevonombre = antiguonombre)
```

Dado que "n" es poco claro, le llamaremos "frecuencia".

```
rename(guaguas, frecuencia=n)
```

```
## # A tibble: 845,400 x 5
##       anio  nombre sexo   frecuencia proporcion
##       <dbl> <chr>  <chr>     <dbl>        <dbl>
## 1  1920 María    F        2130      0.104
## 2  1920 José     M        984       0.0483
## 3  1920 Juan    M        636       0.0312
## 4  1920 Luis    M        631       0.0310
## 5  1920 Rosa    F        426       0.0209
## 6  1920 Ana     F        340       0.0167
## 7  1920 Manuel  M        326       0.0160
## 8  1920 Olga    F        289       0.0142
## 9  1920 Carlos  M        277       0.0136
## 10 1920 Pedro   M        269       0.0132
```

rename()

Como siempre, para que el cambio se haga efectivo hay que sobrescribir la data inicial.

```
guaguas <- rename(guaguas, frecuencia=n)
```

Lo mismo para select(), filter() y todo lo que hagamos en R.

```
guaguas <- select(guaguas, 5:1)
```

Si aplicamos una función veremos resultados en la consola, pero estos no se guardarán a menos que guardemos en un objeto o "sobrescribamos" la data inicial.

rename()

Con R base el cambio de nombre podría haber sido así:

```
guaguas$frecuencia <- guaguas$n  
guaguas <- guaguas[, -4]
```

Primera linea para nueva variable, la segunda para eliminar la variable antigua.

mutate()

"Muta" nuestra data, agregando una nueva columna.

```
mutate(data, nuevavariable = valorotorgado)
```

Por ahora solo la utilizaremos para cosas simples, como replicar una variable

```
mutate(guaguas, nueva=n)
```

```
## # A tibble: 1 x 6
##   anio nombre sexo     n proporcion nueva
##   <dbl> <chr>  <chr> <dbl>      <dbl> <dbl>
## 1 1920 María   F     2130      0.104  2130
```

O multiplicar por 100:

```
mutate(guaguas, porcentaje=proporcion*100)
```

```
## # A tibble: 1 x 6
##   anio nombre sexo     n proporcion porcentaje
##   <dbl> <chr>  <chr> <dbl>      <dbl>        <dbl>
## 1 1920 María   F     2130      0.104       2130
```

mutate()



mutate()

Podemos también darle valores a nuestro gusto a la nueva variable.

Crear nueva data más pequeña

```
guaguas2 <- guaguas[1:7, ]
```

Crear variable que identifique si los nombres son de personajes de la biblia:

```
guaguas2 <- mutate(guaguas2, biblicos=c("si", "si", "si", "si", "no", "no", "r
```

Ver el resultado tabulado

```
table(guaguas2$biblicos)
```

```
##  
## no si  
## 3 4
```

mutate()

Con R base también lo logramos de forma similar y más simple:

```
guaguas2$biblicos <- c("si","si","si","si","no","no","no")
```

Sin embargo mutate() nos permite asignar valores en base a **condiciones** y en base a **agrupamientos**.

mutate() para condiciones

Muchas formas.

Por ahora veremos una que en su forma simple asigna dos valores, uno para la condición verdadera y otra para la condición falsa.

```
mutate(data, nuevavariable = if_else(condicion,  
                                      verdadero,  
                                      falso))
```

Quiero clasificar los nombres según su popularidad

Nombres usados para el 5% o más de las inscripciones.

```
guaguas <- mutate(guaguas, populares = if_else(proporcion >= 0.03,  
                                              "populares",  
                                              "no populares"))
```

mutate() para condiciones

```
head(guaguas, 3)
```

```
## # A tibble: 3 x 6
##   anio nombre sexo     n proporcion populares
##   <dbl> <chr>  <chr> <dbl>      <dbl> <chr>
## 1 1920 María   F     2130      0.104  populares
## 2 1920 José    M     984       0.0483 populares
## 3 1920 Juan    M     636       0.0312 populares
```

```
table(filter(guaguas, populares == "populares")$nombre)
```

```
##
## Benjamín   Camila    José    Juan    Luis    María
##          4         2        54       60      51      67
```

Operador pipe

(Solo si queda tiempo en la clase, su uso no es necesario en tarea N°2).

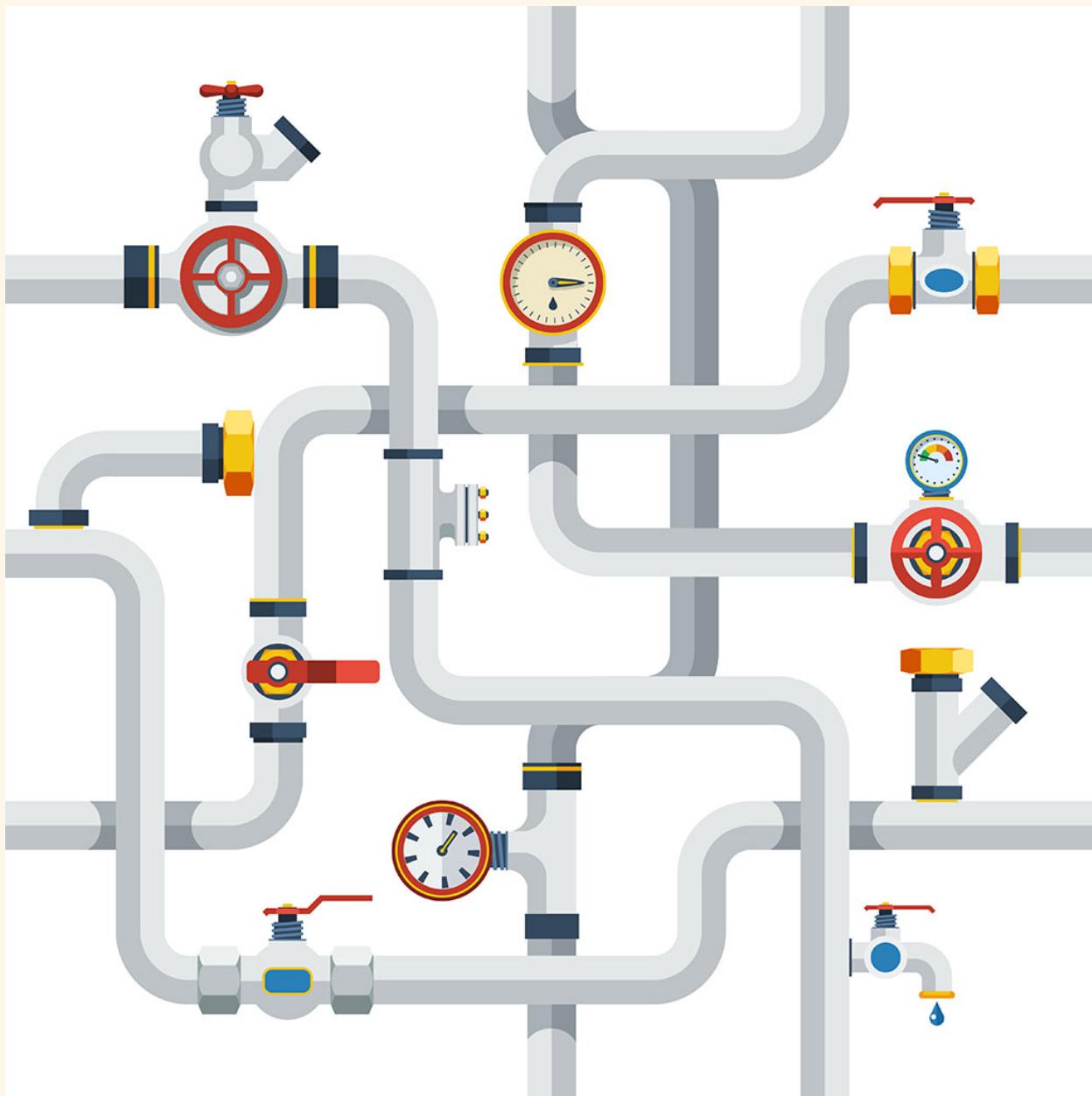
Significa "tubo", "tubería" o "cañería".

Es un operador de magrittr que se combina con los verbos de dplyr.

Se escribe %>% (ctrl + shift + m)



Nos permite concatenar funciones, haciendo más sencilla la lectura del código.



Operador pipe

¿Como saber los nombres de las variables de la base de datos guaguas?

```
names(guaguas)    ## la manera "tradicional" o R base  
  
## [1] "anio"        "nombre"       "sexo"         "n"           "proporcion"  
## [6] "populares"
```

Con pipes sería así:

```
guaguas %>% names()    ## el objeto primero, luego la función  
  
## [1] "anio"        "nombre"       "sexo"         "n"           "proporcion"  
## [6] "populares"
```

Operador pipe

¿Cuál es la importancia relativa del nombre María en 1920, 1950, 1980 y 2020?

```
filter(guaguas,  
       (anio==1920|anio==1950|anio==1980|anio==2020) & nombre=="María")
```

```
## # A tibble: 4 x 6  
##   anio  nombre sexo     n  proporcion  populares  
##   <dbl> <chr>  <chr> <dbl>      <dbl> <chr>  
## 1 1920  María   F     2130      0.104  populares  
## 2 1950  María   F     15023     0.0859  populares  
## 3 1980  María   F     10382     0.0365  populares  
## 4 2020  María   F     1345      0.00690 no populares
```

Con pipes sería así:

```
guaguas %>%  
  filter( (anio==1920|anio==1950|anio==1980|anio==2020) &  
          nombre=="María")
```

Pipes

¿Y cuál es la gracia de la tubería?

Con pipes podemos concatenar funciones.

¿Cuáles son los años en los que el nombre María es más importante?, ¿Que porcentajes representa el nombre María?

Sin pipes tendríamos que hacer algo más o menos así (con el riesgo de confundir paréntesis)

```
select(mutate(head(arrange(filter(guaguas,nombre=="María),-proporcion),  
porcentaje=round(proporcion*100,1)),anio,nombre,porcentaje)
```

Se dificulta bastante la lectura (desde el centro hacia afuera)

Pipes

Con pipes es como leer, de izquierda a derecha:
(y de abajo hacia arriba opcional)

```
guaguas %>%
  filter(nombre=="María") %>%
  arrange(-proporcion) %>%
  head(4) %>%
  mutate(porcentaje=round(proporcion*100,1)) %>%
  select(anio,nombre,porcentaje)

## # A tibble: 4 x 3
##   anio nombre porcentaje
##   <dbl> <chr>     <dbl>
## 1 1920 María      10.4
## 2 1921 María      9.2
## 3 1922 María      9
## 4 1948 María      8.8
```

Tarea práctica N°2

Ocupa las herramientas de `dplyr` vistas en clase sobre la base `guaguas` u otra para responder a las preguntas

Recursos web utilizados

Xaringan: Presentation Ninja, de Yihui Xie. Para generar esta presentación.

Ilustraciones de Allison Horst

Sorting data by Jozef

Para reforzar y seguir aprendiendo

Administración de datos en R. ¿Qué es dplyr?.

Capítulo 5 libro "Ciencia de Datos" de Hadley Wickham

Operadores lógicos en R

Bibliografía utilizada

Wickham, H. (2021). *R Para Ciencia de Datos*. URL: <https://es.r4ds.hadley.nz/>.