



Trabajo básico con filas

Contenidos

- 01.** Trabajo básico con filas
- 02.** Slicing
- 03.** Paréntesis: operador pipe (`%>%`)
- 04.** Filter



01.

Trabajo básico con filas



Una vez logramos cargar los datos con los que trabajamos, vamos a querer explorarlos, cortarlos, modificarlos. Para estas tareas, utilizaremos el **paquete dplyr**:

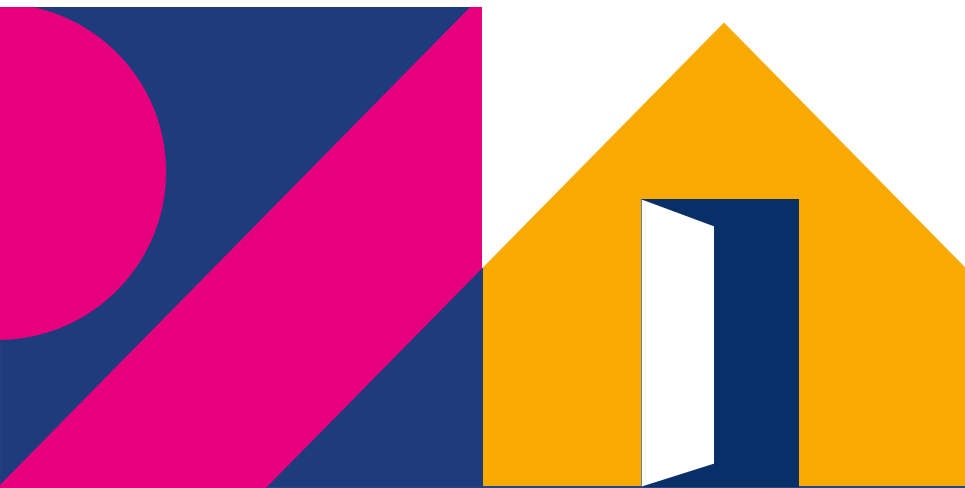


Para esta cápsula nos centraremos en la familia de funciones **slice** y en la función **filter**.



02.

Slicing





El slicing se refiere a seleccionar un subconjunto de los elementos de un objeto

En nuestro caso, queremos seleccionar un número determinado de filas. Para ejemplificar, volvamos a cargar la tabla hospital.csv de la cápsula anterior. Aprovechamos de traducir las columnas al español:

```
library(readr)
```

```
hospital <- read_csv(file = "data/hospital.csv")
```

```
names(hospital) = c('Numero', 'Ciudad', 'Genero', 'Edad', 'Ingreso', 'Enfermedad')
```

Si quisiéramos imprimir la tabla, esto sería demandante para R, pues tiene 150.000 filas.



Una alternativa para explorar las primeras filas de una tabla es usar la función `slice_head()`, que nos entrega los n primeros elementos de un dataframe:

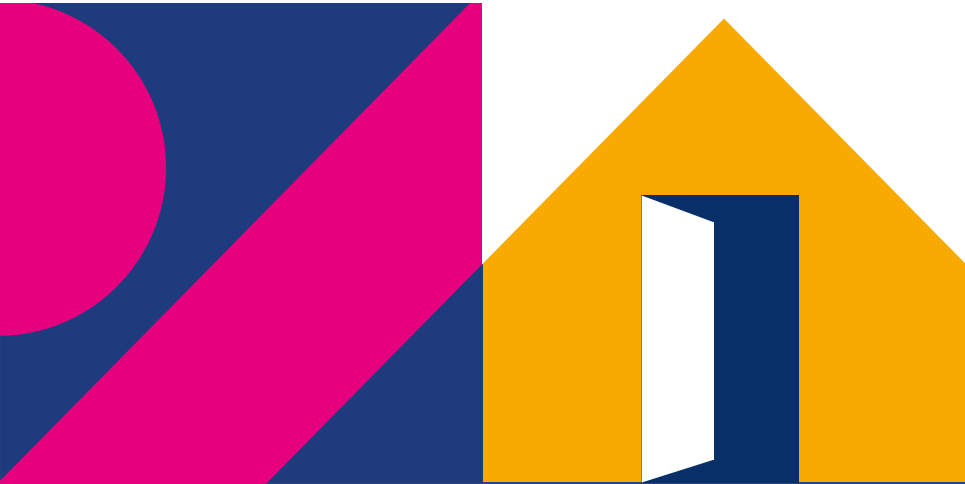
```
hospital %>% slice_head(n = 5)
```

```
## # A tibble: 5 × 6
##   Numero Ciudad Genero   Edad Ingreso Enfermedad
##   <dbl> <chr>   <chr> <dbl> <dbl> <chr>
## 1     1  Dallas Male     41  40367 No
## 2     2  Dallas Male     54  45084 No
## 3     3  Dallas Male     42  52483 No
## 4     4  Dallas Male     40  40941 No
## 5     5  Dallas Male     46  50289 No
```



03.

**Paréntesis: operador
pipe (%>%)**





Este operador es básico para la sintaxis utilizando *tidyverse* y viene del paquete *magrittr*

Permite hacer el código más legible. En la práctica simplemente utiliza el elemento de la izquierda, como el primer argumento de la función de la derecha:

```
x %>% f == f(x)
```

En el ejemplo anterior teníamos `hospital %>% slice_head(n = 5)`, lo que es equivalente a `slice_head(data = hospital, n = 5)`.

La sintaxis usando *pipe* permite leer el código de izquierda a derecha, casi como texto escrito. En este caso podríamos leerlo como: "al dataset `hospital` aplícale la función `slice_head`".



Las funciones del tidyverse están pensadas de forma que el primer argumento de estas es siempre una tabla de datos, lo que implica que podemos encadenar múltiples *pipes*:

```
hospital %>% slice_head(n = 3) %>% select(Numero, Genero) %>% filter(Genero == 'Male')
```

```
## # A tibble: 3 × 2
##   Numero Genero
##   <dbl> <chr>
## 1     1 Male
## 2     2 Male
## 3     3 Male
```



- Aquí nos adelantamos un poco con el contenido, pero el código anterior se puede leer como **"del dataset hospital, obtén las 3 primeras filas, luego selecciona las columnas Numero y Genero y finalmente mantén solo las filas cuyo Genero es 'Male'"**.
- Esto resulta más fácil de leer que la sintaxis alternativa sin uso de pipes:

```
filter(data = select(data = slice_head(data = hospital, n = 3), Numero, Genero), Genero == 'Male')
```



Slicing

Si queremos los últimos n elementos usamos **slice_tail**:

```
hospital %>% slice_tail(n = 3)
```

```
## # A tibble: 3 × 6
##   Numero Ciudad Genero  Edad Ingreso Enfermedad
##   <dbl> <chr>  <chr> <dbl>  <dbl> <chr>
## 1 149998 Austin Male     26  111885 No
## 2 149999 Austin Male     25  111878 No
## 3 150000 Austin Female   37   87251 No
```



Slicing

Si queremos una muestra aleatoria de n elementos usamos **slice_sample**:

```
hospital %>% slice_sample(n = 3)
```

```
## # A tibble: 3 × 6
##   Numero Ciudad      Genero  Edad Ingreso Enfermedad
##   <dbl> <chr>      <chr> <dbl> <dbl> <chr>
## 1  77307 Los Angeles Female   59  81456 Yes
## 2  19903 New York City Male     48  95380 No
## 3  96258 Los Angeles Female   28  87773 No
```

Slicing

Si queremos una selección propia de elementos usamos **slice**:

```
hospital %>% slice(3:7)
```

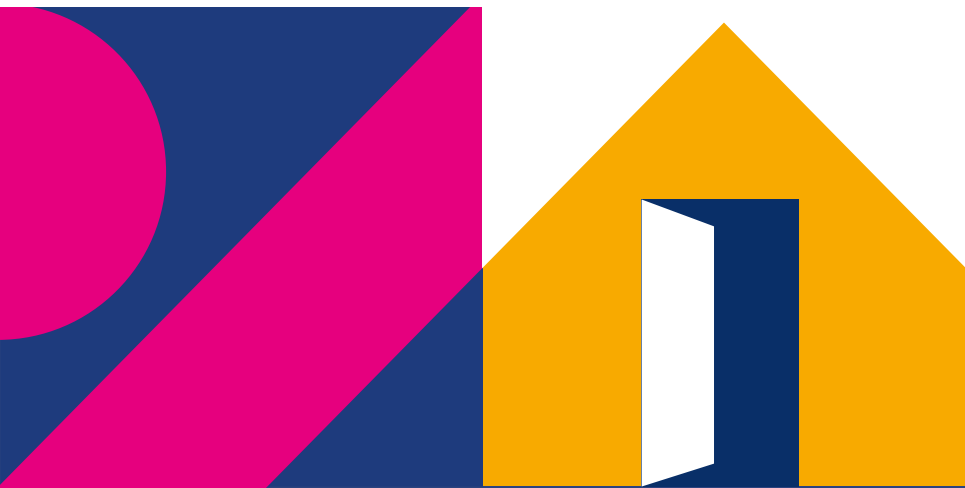
```
## # A tibble: 5 × 6
##   Numero Ciudad Genero   Edad Ingreso Enfermedad
##   <dbl> <chr>   <chr> <dbl>   <dbl> <chr>
## 1     3 Dallas Male     42   52483 No
## 2     4 Dallas Male     40   40941 No
## 3     5 Dallas Male     46   50289 No
## 4     6 Dallas Female   36   50786 No
## 5     7 Dallas Female   32   33155 No
```

En este caso usamos la notación a:b, que en el ejemplo significa "quiero todos los elementos desde el 3 hasta el 7", es decir, los elementos 3, 4, 5, 6 y 7.



04.

Filter





Muchas veces no queremos una selección de filas por su posición, sino que queremos que cumplan cierta condición. Para estos casos utilizaremos la función **filter**.

Por ejemplo: ¿Cómo puedo seleccionar las personas que están enfermas?

```
hospital %>% filter(Enfermedad == 'Yes') %>% slice_head(n = 5)
```

```
## # A tibble: 5 × 6
##   Numero Ciudad Genero   Edad Ingreso Enfermedad
##   <dbl> <chr>   <chr> <dbl> <dbl> <chr>
## 1     11 Dallas Female   48   41524 Yes
## 2     18 Dallas Male     38   46373 Yes
## 3     24 Dallas Female   27   34292 Yes
## 4     30 Dallas Male     45   47421 Yes
## 5     39 Dallas Female   61   39881 Yes
```

Seguimos usando slice_head para controlar el tamaño de la tabla.



Filter y operadores lógicos

En el ejemplo anterior utilizamos el operador lógico `==`, que significa igualdad entre dos valores. Otros operadores lógicos de uso común son:

`!=` distinto a

`>`, `>=` mayor a y mayor o igual a

`<`, `<=` menor y menor o igual a

`%in%` pertenece al conjunto

Numero	Ciudad	Genero	Edad	Ingreso	Enfermedad
1	Dallas	Male	41	40367	No
2	Dallas	Male	54	45084	No
3	Dallas	Male	42	52483	No
5	Dallas	Male	46	50289	No

Probemos algunas de estas condiciones:

- **Personas con 41 o más años:**

```
hospital %>% filter(Edad >= 41) %>% slice_head(n = 4)
```



Probemos algunas de estas condiciones:

- **Personas de Austin o Boston:**

```
hospital %>% filter(Ciudad %in% c('Austin', 'Boston')) %>% slice_head(n = 4)
```

Numero	Ciudad	Genero	Edad	Ingreso	Enfermedad
116407	Boston	Female	57	87004	No
116408	Boston	Male	60	93196	No
116409	Boston	Male	25	112492	No
116410	Boston	Male	28	91910	No



Probemos algunas de estas condiciones:

- **Personas cuyo sexo no es masculino:**

```
hospital %>% filter(Genero != 'Male') %>% slice_head(n = 4)
```

Numero	Ciudad	Genero	Edad	Ingreso	Enfermedad
6	Dallas	Female	36	50786	No
7	Dallas	Female	32	33155	No
10	Dallas	Female	30	50082	No
11	Dallas	Female	48	41524	Yes



Filter y operadores lógicos

También podríamos querer que se cumpla más de una condición o bien alguna condición de varias. Para esto tenemos los operadores lógicos "&" y "|":

condicion1 & condicion2 operador "y", mantenemos filas que cumplen las dos condiciones al mismo tiempo.

condicion1 | condicion2 operador "o", mantenemos filas que cumplen cualquiera de las condiciones.

- **Hombres mayores de 64 años:**

```
hospital %>% filter(Genero == 'Male' & Edad > 64)
```

Numero	Ciudad	Genero	Edad	Ingreso	Enfermedad
405	Dallas	Male	65	54304	No
470	Dallas	Male	65	52691	No
834	Dallas	Male	65	49903	No
1045	Dallas	Male	65	78511	No



Mujeres en Dallas o Boston:

```
hospital %>% filter(Genero == 'Female' & (Ciudad == 'Dallas' | Ciudad == 'Boston'))
```

Numero	Ciudad	Genero	Edad	Ingreso	Enfermedad
6	Dallas	Female	36	50786	No
7	Dallas	Female	32	33155	No
10	Dallas	Female	30	50082	No
11	Dallas	Female	48	41524	Yes



Mujeres en Dallas o Boston:

```
hospital %>% filter((Genero == 'Female' & Ciudad == 'Dallas') | Ciudad == 'Boston')
```

Numero	Ciudad	Genero	Edad	Ingreso	Enfermedad
116408	Boston	Male	60	93196	No
116409	Boston	Male	25	112492	No
116410	Boston	Male	28	91910	No
116412	Boston	Male	41	100492	Yes

¡Muchas gracias!





www.ine.gob.cl