

Procesando datos con **{tidyverse}**

Principales herramientas (funciones) para el tratamiento de datos

 Estación R

2024-02-21

Bienvenidos y bienvenidas a Estación R

 Slack

 Web

 Mastodon

 X

 Correo

Instagram

LinkedIn

¿Qué vimos?

- ✓ Conceptos básicos de R (valores, vectores, data.frames, funciones, objetos)
- ✓ Cómo armar y organizar un proyecto de trabajo
- ✓ Qué son los paquetes (o librerías)

Hoja de Ruta

📌 ¿Qué es la Ciencia de Datos?

📦 Paquete {dplyr}

```
🔧 `select()` `🔧 `filter()`  
🔧 `mutate()` `🔧 `rename()`  
🔧 `arrange()`  
🔧 `group_by()` `🔧 `summarise()`  
🔧 `joins`
```

📌 La pipa (|> o %>%)

📦 Paquete {tidyverse}

```
🔧 `pivot_longer()` `🔧 `pivot_wider()`
```

Configuración para esta clase

- Armar un proyecto de trabajo nuevo
- Crear una carpeta en el llamada **datos**
- Descargar la base del Padrón Único Nacional de Alojamientos (Argentina) y ubicarla en la carpeta **datos**
- Crear un script de trabajo

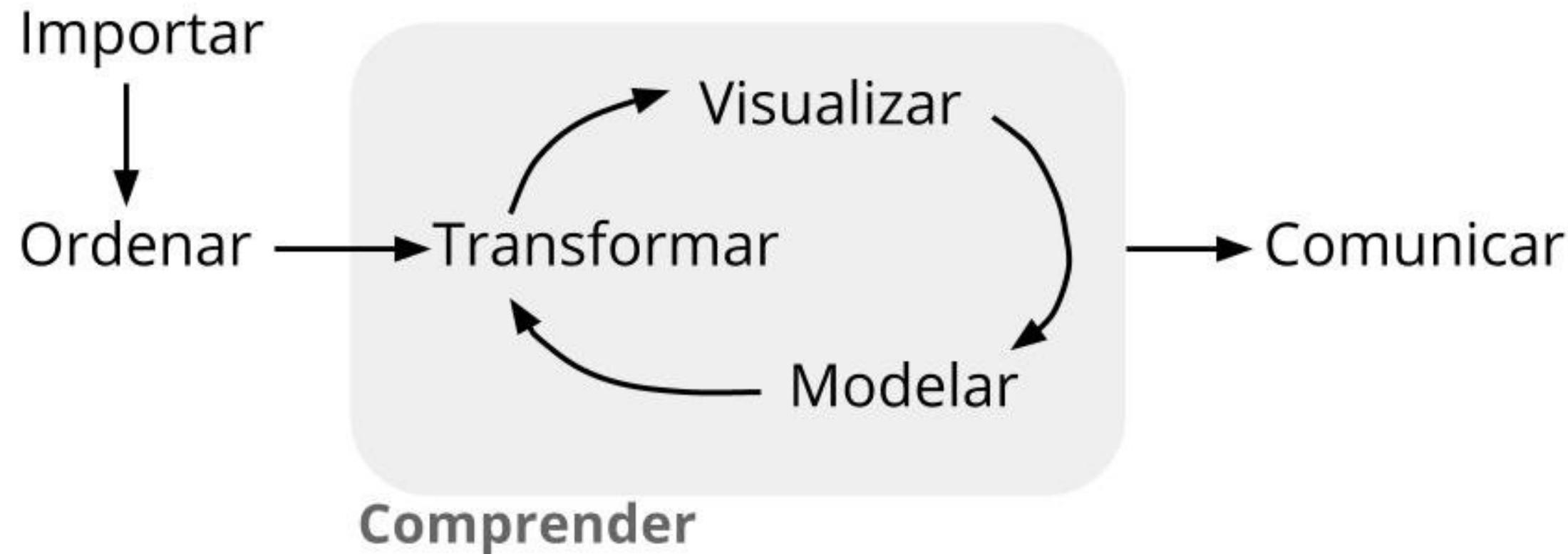


¿Qué es la Ciencia de Datos?



¿Qué es la Ciencia de Datos?

El proceso del análisis de datos



{tidyverse}

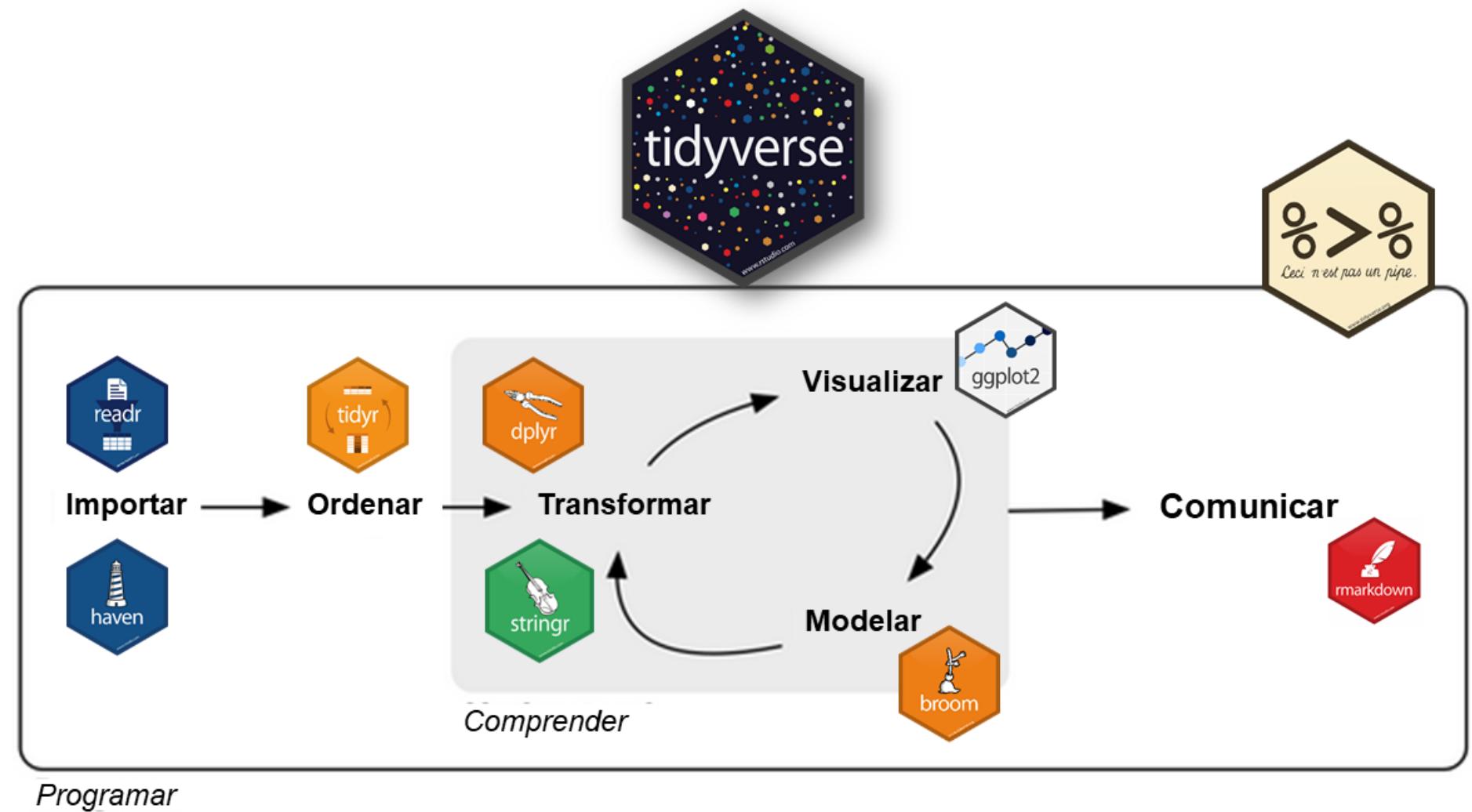


¿Qué es {tidyverse}?

- Una colección de paquetes.
- Comparten una filosofía acerca de los datos y la programación en R (“*tidy* -ordenado-).
- Tienen una coherencia para ser utilizados en conjunto.
- Orientado a ser leído y escrito por y para seres humanos.
- Una comunidad, basada en los principios del código abierto y trabajo colaborativo.



¿Qué es {tidyverse}?



{tidyverse}

- Instalación:

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

{tidyverse}

- Cargo el paquete:

```
library(tidyverse)
-- Attaching core tidyverse packages ━━━━━━━━ tidyverse 2.0.0 ━━━━━━
✓ dplyr     1.1.4      ✓ readr     2.1.5
✓forcats    1.0.0      ✓ stringr   1.5.1
✓ ggplot2   3.4.4      ✓ tibble    3.2.1
✓ lubridate 1.9.3      ✓ tidyrr    1.3.0
✓ purrr    1.0.2
-- Conflicts ━━━━━━━━ tidyverse_conflicts() ━━━━━━
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

{tidyverse}

- Nos evita tener que instalar uno por uno a cada paquete:

```
install.packages("dplyr")
install.packages("tidyr")
install.packages("ggplot2")
```

- Como también tener que convocarlos de a uno:

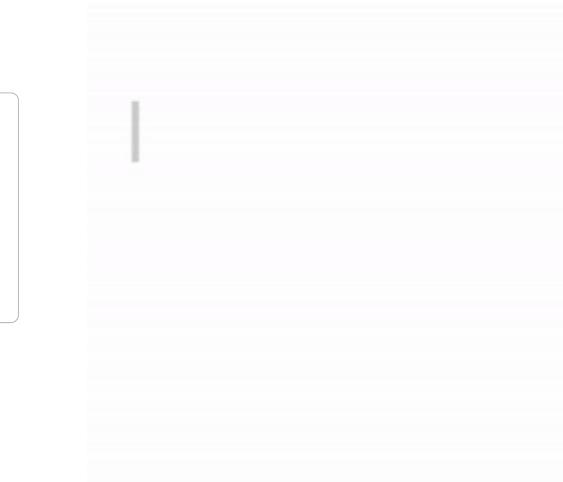
```
library(dplyr)
library(tidyr)
library(ggplot2)
```



La pipa

Un operador llamado pipa

```
base_de_datos |>  
  funcion1 |>  
  funcion2 |>  
  funcion3
```



Un operador llamado **pipa**

- Pipa de R base: `|>`
- Pipa de {magritr}: `%>%`



Ejemplo:

```
datos <- data.frame(nombre = c("Pirulanzo", "Rodogovia", "Rodogovia", "Rodogovia"),  
                     edad = c(23, 12, 87, 32))
```

```
datos
```

	nombre	edad
1	Pirulanzo	23
2	Rodogovia	12
3	Rodogovia	87
4	Rodogovia	32

Ejemplo:

- Quiero calcular qué proporción de personas se llaman *Rodogovia*
- Antes (*sin el pipe*):

```
round(prop.table(table(datos$nombre)), digits = 1)
```

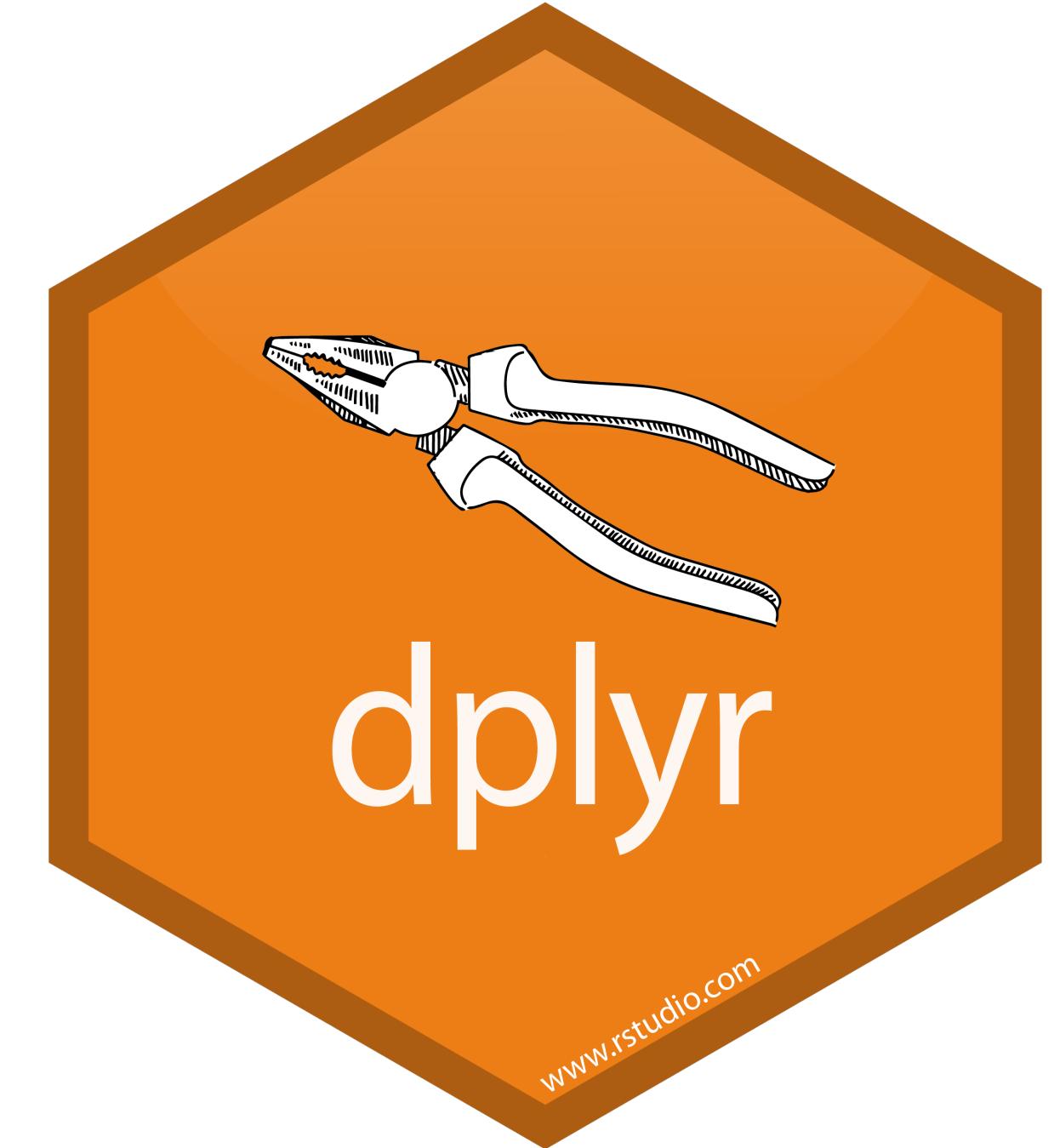
Pirulanzo	Rodogovia
0.2	0.8

Ejemplo:

- Después (*con el pipe*):

```
datos$nombre |>  
  table() |>  
  prop.table() |>  
  round(digits = 1)
```

Pirulanzo Rodogovia
0.2 0.8



Funciones del paquete dplyr:

Función	Acción
<code>select()</code>	<i>selecciona o descarta variables</i>
<code>filter()</code>	<i>selecciona filas</i>
<code>mutate()</code>	<i>crea / edita variables</i>
<code>rename()</code>	<i>renombrá variables</i>
<code>group_by()</code>	<i>segmenta en función de una variable</i>
<code>summarize()</code>	<i>genera una tabla de resumen</i>

select()

Elige o descarta columnas de una base de datos



select()

- La función tiene la siguiente estructura:

```
base_de_datos |>  
  select(id, nombre) #<<
```

Diagrama que ilustra la ejecución de la función `select()`. Un cuadro azul titulado "Columnas seleccionadas" tiene una flecha apuntando hacia el cuadro verde que rodea las columnas `id` y `nombre` de la tabla.

Id	nombre	Edad	localidad	tipo_alojamiento
1	Pepa	25	Jujuy	Casa
2	Juana	64	Jujuy	Casa
3	Rigoberta	13	La Pampa	Deptó
4	Anastacio	87	Córdoba	Deptó
5	Luguercio	68	Jujuy	Deptó
6	Lolo	5	Chubut	Casa

Caso práctico

```
# Cargo paquete  
library(readr)  
  
# Importo datos  
df_puna <- read_csv("datos/puna_base_agregada.csv")
```

```
# Exploro la base  
colnames(df_puna)  
  
[1] "indice_tiempo"          "region"  
[3] "ruta_natural"           "provincia_codigo"  
[5] "provincia_nombre"       "departamento_partido"  
[7] "localidad"              "clasificacion_minturdep"  
[9] "tipo"                   "establecimientos"  
[11] "unidades"               "habitaciones"  
[13] "plazas"
```

Caso práctico

- Pedido: La coordinadora me ha solicitado conocer la cantidad de plazas que hay por localidad y, si es posible, saber de qué tipo son los alojamientos
- Variables de trabajo:
 - *localidad*
 - *plazas*
 - *tipo*

Caso práctico

- Seleccióno las 3 columnas de interés

```
1 library(tidyverse)
2
3 df_puna |>
4   select(localidad, tipo, plazas)
```

Caso práctico

- Seleccióno las 3 columnas de interés

```
1 library(tidyverse)
2
3 df_puna_sel <- df_puna |>
4   select(localidad, tipo, plazas)
```

- Chequeo las columnas del nuevo objeto

```
1 colnames(df_puna_sel)
[1] "localidad" "tipo"      "plazas"
```

Otras formas de seleccionar...



select() - por posición

1. Chequeo la posición:

```
colnames(df_puna)
```

```
[1] "indice_tiempo"           "region"  
[3] "ruta_natural"           "provincia_codigo"  
[5] "provincia_nombre"        "departamento_partido"  
[7] "localidad"               "clasificacion_minturdep"  
[9] "tipo"                    "establecimientos"  
[11] "unidades"                "habitaciones"  
[13] "plazas"
```



select() - por posición

2. Selección

```
1 df_puna_sel_posicion <- df_puna |>  
2   select(7, 9, 13)
```



select() - *por posición*

3. Chequeo

```
colnames(df_puna_sel_posicion)  
[1] "localidad" "tipo"      "plazas"
```

select() - por posición (*columnas consecutivas*)

```
1 df_puna_sel_posicion2 <- df_puna |>  
2   select(1:3)
```



select() - por posición (*columnas consecutivas*)

```
1 df_puna_sel_posicion2 <- df_puna |>  
2   select(1:3)
```

```
colnames(df_puna_sel_posicion2)  
[1] "indice_tiempo" "region"          "ruta_natural"
```



select() - por nombre (consecutiva)

```
1 df_puna_sel_posicion3 <- df_puna |>  
2   select(establecimientos:plazas)
```

select() - por nombre (consecutiva)

```
1 df_puna_sel_posicion3 <- df_puna |>  
2   select(establecimientos:plazas)
```

```
colnames(df_puna_sel_posicion3)  
[1] "establecimientos" "unidades"          "habitaciones"      "plazas"
```



select() - *Por patrones de texto*

Trío:

- `starts_with()` -> *empieza con...*
- `ends_with()` -> *termina con...*
- `contains()` -> *contiene...*



select() + starts_with()

```
1 df_puna_sel_patron1 <- df_puna |>  
2   select(starts_with("provincia"))
```



select() + starts_with()

```
1 df_puna_sel_patron1 <- df_puna |>  
2   select(starts_with("provincia"))
```

```
colnames(df_puna_sel_patron1)  
[1] "provincia_codigo" "provincia_nombre"
```

select() + ends_with()

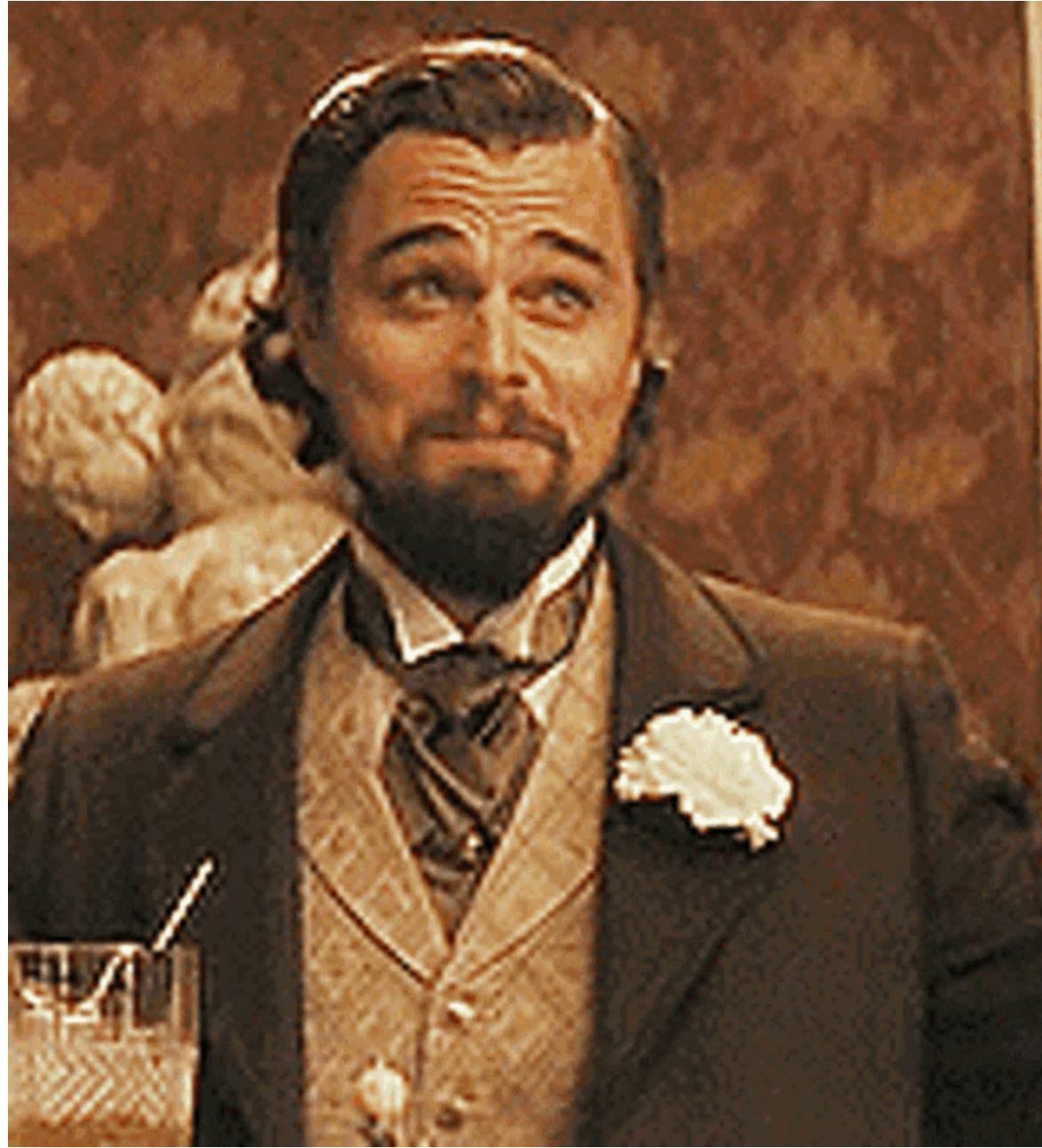
```
1 df_puna_sel_patron2 <- df_puna |>  
2   select(ends_with("o"))
```



select() + starts_with()

```
1 df_puna_sel_patron2 <- df_puna |>  
2   select(ends_with("o"))
```

```
colnames(df_puna_sel_patron2)  
[1] "indice_tiempo"      "provincia_codigo"    "departamento_partido"  
[4] "tipo"
```



select() + contains()

```
1 df_puna_sel_patron3 <- df_puna |>  
2   select(contains("_"))
```



select() + contains()

```
1 df_puna_sel_patron3 <- df_puna |>  
2   select(contains("_"))
```

```
colnames(df_puna_sel_patron3)  
[1] "indice_tiempo"          "ruta_natural"  
[3] "provincia_codigo"       "provincia_nombre"  
[5] "departamento_partido"   "clasificacion_minturdep"
```



LA COMBINACIÓN FINAL



select()

```
1 df_puna_select_tuto <- df_puna |>  
2   select(localidad, 2, starts_with("provincia"), 9:11)
```



select()

```
1 df_puna_select_tuto <- df_puna |>  
2   select(localidad, 2, starts_with("provincia"), 9:11)
```

```
colnames(df_puna_select_tuto)  
[1] "localidad"          "region"           "provincia_codigo" "provincia_nombre"  
[5] "tipo"               "establecimientos" "unidades"
```



Ejercitación grupal



Ejercitación

- Crear un objeto en donde importamos la base de datos de Alojamientos.
- Seleccionar 3 variables de la base según el nombre de las mismas y guardar en otro objeto.
- Seleccionar 3 variables de la base según la posición de las mismas y guardar en otro objeto.
- Seleccionar todas las variables que empiecen con un patrón de texto (a elegir).



filter()

Define los casos (filas) en base a una condición



filter()

- La función tiene la siguiente estructura:

```
1 base_de_datos |>  
2   filter(condicion)
```

Condición $\text{edad} > 65$

Id	nombre	Edad	localidad	tipo_alojamiento
1	Pepa	25	Jujuy	Casa
2	Juana	64	Jujuy	Casa
3	Rigoberta	13	La Pampa	Depto
4	Anastacio	87	Córdoba	Depto
5	Luguercio	68	Jujuy	Depto
6	Lolo	5	Chubut	Casa

Filas
seleccionadas

filter()

- La función tiene la siguiente estructura:

```
1 base_de_datos |>  
2   filter(Edad > 65)
```

Condición $\text{edad} > 65$

Id	nombre	Edad	localidad	tipo_alojamiento
1	Pepa	25	Jujuy	Casa
2	Juana	64	Jujuy	Casa
3	Rigoberta	13	La Pampa	Depto
4	Anastacio	87	Córdoba	Depto
5	Luguercio	68	Jujuy	Depto
6	Lolo	5	Chubut	Casa

Filas
seleccionadas

Caso práctico

- La directora de tesis me pidió que estudie los alojamientos de tipo Camping.
- Universo de análisis / Población de estudio:
 - Alojamientos tipo *Camping*



Caso práctico

- Chequeo con qué tipos de alojamientouento en la base:



Caso práctico

- Chequeo con qué tipos de alojamiento cuenta en la base:

```
unique(df_puna$clasificacion_minturdep)  
[1] "Albergue municipal / complejo deportivo"  
[2] "Caba\xf1as / bungalows"  
[3] "Conjunto de unidades turisticas"  
[4] "Establecimiento rural"  
[5] "Hosteria"  
[6] "Hotel 1 estrella"  
[7] "Hotel 2 estrellas"  
[8] "Hotel boutique"  
[9] "Hotel sin categorizar"  
[10] "Residencial"  
[11] "Camping"  
[12] "Hotel 3 estrellas"  
[13] "Sin clasificar"  
[14] "Motel"  
[15] "Hospedaje"  
[16] "Posada"  
[17] "Apart. hotel"
```

Caso práctico

- Aplico filtro

```
library(tidyverse)

df_filtrada <- df_puna |>
  filter(clasificacion_minturdep == "Camping")
```

Caso práctico

- Chequeo filtro

```
unique(df_filtrada$clasificacion_minturdep)  
[1] "Camping"
```

filter()

Condición Acción

`==` *igual*

`%in%` *incluye*

`!=` *distinto*

`>` *mayor que*

`<` *menor que*

`>=` *mayor o igual que*

`<=` *menor o igual que*

Operador Descripción

`&` *y* - Cuando se cumplen ambas condiciones

`|` *o* - Cuando se cumple una u otra condición



Caso práctico

- La encargada de la oficina de turismo de Buenos Aires quiere que le arme una base sólo con alojamientos de tipo *Camping* y *Hoteles 3 estrellas*.

Caso práctico

- Chequeo las categorías de la variable `clasificacion_minturdep`

```
unique(df_puna$clasificacion_minturdep)  
[1] "Albergue municipal / complejo deportivo"  
[2] "Caba\xf1as / bungalows"  
[3] "Conjunto de unidades turisticas"  
[4] "Establecimiento rural"  
[5] "Hosteria"  
[6] "Hotel 1 estrella"  
[7] "Hotel 2 estrellas"  
[8] "Hotel boutique"  
[9] "Hotel sin categorizar"  
[10] "Residencial"  
[11] "Camping"  
[12] "Hotel 3 estrellas"  
[13] "Sin clasificar"  
[14] "Motel"  
[15] "Hospedaje"  
[16] "Posada"  
[17] "Apart. hotel"
```

Caso práctico

- Filtro (opción 1):

```
df_camping_y_hoteles3estrellas <- df_puna |>  
  filter(clasificacion_minturdep == "Camping" | clasificacion_minturdep == "Hotel 3 estrellas")
```



Caso práctico

- Chequeo filtro

```
unique(df_camping_y_hoteles3estrellas$clasificacion_minturdep)  
[1] "Camping"           "Hotel 3 estrellas"
```

Caso práctico

- Filtro (opción 2, operador: `%in%`):

```
df_camping_y_hoteles3estrellas <- df_puna |>  
  filter(clasificacion_minturdep %in% c("Camping", "Hotel 3 estrellas"))
```

Caso práctico

- Chequeo filtro

```
unique(df_camping_y_hoteles3estrellas$clasificacion_minturdep)  
[1] "Camping"           "Hotel 3 estrellas"
```

select() + filter()

```
1 df_puna |>  
2   select(localidad,  
3           clasificacion_minturdep)
```

```
# A tibble: 14,717 × 2  
  localidad clasificacion_minturdep  
  <chr>     <chr>  
1 Baradero  "Albergue municipal / complejo  
deportivo"  
2 Baradero  "Caba\xf1as / bungalows"  
3 Baradero  "Conjunto de unidades turisticas"  
4 Baradero  "Establecimiento rural"  
5 Baradero  "Hosteria"  
6 Baradero  "Hotel 1 estrella"  
7 Baradero  "Hotel 2 estrellas"  
8 Baradero  "Hotel boutique"  
9 Baradero  "Hotel sin categorizar"  
10 Baradero "Residencial"  
# i 14,707 more rows
```

select() + filter()

```
1 df_puna |>  
2   select(localidad,  
3           clasificacion_minturdep) |>  
4   filter(clasificacion_minturdep %in% c("Camping", "Hotel 3 estrellas"))
```

```
# A tibble: 1,438 × 2  
  localidad    clasificacion_minturdep  
  <chr>        <chr>  
1 Belen de Escobar Camping  
2 Belen de Escobar Hotel 3 estrellas  
3 Fatima       Hotel 3 estrellas  
4 Ramallo      Hotel 3 estrellas  
5 Villa Ramallo Hotel 3 estrellas  
6 San Nicolas de Los Arroyos Hotel 3 estrellas  
7 San Pedro     Camping  
8 San Pedro     Hotel 3 estrellas  
9 Zarate        Hotel 3 estrellas  
10 9 de Julio   Hotel 3 estrellas  
# i 1,428 more rows
```

Ejercitación grupal



Ejercitación

- Crear un objeto que contenga la base de PUNA sólo con las variables `localidad`, `ruta_natural` y `plazas`
- En ese mismo objeto, quedarse sólo con las filas de la ruta natural `Delta`.
- Calcular cuántas plazas hay en total para la ruta natural `Delta` (*tip: la función `sum()` puede ser de ayuda*)



mutate()

Crea / edita variables (columnas)



mutate()

- La función tiene la siguiente estructura:

```
1 base_de_datos %>%
2     mutate(var_nueva = var_1 + var_2)
```



mutate() - Caso práctico

- Llega a la oficina una persona interesada en saber cuál es el valor total disponible para dormir en los establecimientos. Quiere, entonces, conocer el resultado de la suma entre `habitaciones` y `plazas`.
- Para ello, podemos crear una variable que contenga este resultado:



mutate() - Caso práctico

```
1 df_lugar_tot <- df_puna |>  
2   select(localidad, habitaciones, plazas)
```

```
# A tibble: 14,717 × 3  
  localidad habitaciones plazas  
  <chr>        <dbl>      <dbl>  
1 Baradero          6       30  
2 Baradero          0      277  
3 Baradero          0       50  
4 Baradero         57      131  
5 Baradero         68      164  
6 Baradero         15       34  
7 Baradero         49      135  
8 Baradero         27       78  
9 Baradero         28       60  
10 Baradero        5        15  
# i 14,707 more rows
```

mutate() - Caso práctico

```
1 df_lugar_tot <- df_puna |>  
2   select(localidad, habitaciones, plazas) |>  
3   mutate(lugar_disponible = habitaciones + plazas)
```

```
# A tibble: 14,717 × 4  
  localidad habitaciones plazas lugar_disponible  
  <chr>        <dbl>    <dbl>            <dbl>  
1 Baradero          6      30              36  
2 Baradero          0     277             277  
3 Baradero          0      50              50  
4 Baradero         57     131             188  
5 Baradero         68     164             232  
6 Baradero         15      34              49  
7 Baradero         49     135             184  
8 Baradero         27      78              105  
9 Baradero         28      60              88  
10 Baradero        5       15              20  
# i 14,707 more rows
```

mutate() + case_when() = Recodificación

- Necesito agregarle una etiqueta a la variable `indice_tiempo` (pasar de `2020` a "Año 2020")



mutate() + case_when() = Recodificación

```
1 df_lugar_tot <- df_puna |>  
2   select(indice_tiempo, localidad, plazas)
```

```
# A tibble: 14,717 × 3  
  indice_tiempo localidad plazas  
  <dbl> <chr>     <dbl>  
1 2020 Baradero 30  
2 2020 Baradero 277  
3 2020 Baradero 50  
4 2020 Baradero 131  
5 2020 Baradero 164  
6 2020 Baradero 34  
7 2020 Baradero 135  
8 2020 Baradero 78  
9 2020 Baradero 60  
10 2020 Baradero 15  
# i 14,707 more rows
```

mutate() + case_when() = Recodificación

```
1 df_puna_recod <- df_puna |>
2   select(indice_tiempo, localidad, plazas) |>
3   mutate(anio_etiqueta = case_when(indice_tiempo == 2020 ~ "Año 2020",
4                                     indice_tiempo == 2021 ~ "Año 2021",
5                                     indice_tiempo == 2022 ~ "Año 2022")
```

```
# A tibble: 14,717 × 4
  indice_tiempo localidad plazas anio_etiqueta
  <dbl> <chr>     <dbl> <chr>
1 2020 Baradero 30 Año 2020
2 2020 Baradero 277 Año 2020
3 2020 Baradero 50 Año 2020
4 2020 Baradero 131 Año 2020
5 2020 Baradero 164 Año 2020
6 2020 Baradero 34 Año 2020
7 2020 Baradero 135 Año 2020
8 2020 Baradero 78 Año 2020
9 2020 Baradero 60 Año 2020
10 2020 Baradero 15 Año 2020
# i 14,707 more rows
```

mutate() + case_when() = Recodificación

- Necesito caracterizar al sector hotelero y compararlo con el resto de los alojamientos



mutate() + case_when() = Recodificación

```
1 df_puna_agrup <- df_puna |>  
2   select(tipo, plazas) |>  
3   mutate(tipo_agrupado = case_when(tipo == "Hoteleros" ~ "Hoteleros",  
4                                     .default = "Otros"))
```

```
# A tibble: 14,717 × 3  
  tipo           plazas tipo_agrupado  
  <chr>          <dbl> <chr>  
1 Otros colectivos     30 Otros  
2 Parahoteleros        277 Otros  
3 Otros colectivos      50 Otros  
4 Parahoteleros         131 Otros  
5 Parahoteleros         164 Otros  
6 Hoteleros             34 Hoteleros  
7 Hoteleros             135 Hoteleros  
8 Hoteleros              78 Hoteleros  
9 Hoteleros              60 Hoteleros  
10 Parahoteleros        15 Otros  
# i 14,707 more rows
```

mutate() + case_when() = Recodificación

- Necesito Reagrupar únicamente a los hoteles de 1, 2 y 3 estrellas, para compararlos frente al resto:



mutate() + case_when() = Recodificación

```
1 df_puna_agrup_hotel <- df_puna |>  
2   select(clasificacion_minturdep, plazas) |>  
3   mutate(clasif_agrupado = case_when(  
4     clasificacion_minturdep %in% c("Hotel 1 estrella",  
5       "Hotel 2 estrellas",  
6       "Hotel 3 estrellas") ~ "Hotel hasta 3  
7     .default = "Otros"))
```

```
# A tibble: 14,717 × 3  
  clasificacion_minturdep    plazas  
  clasif_agrupado  
  <chr>  
  <dbl> <chr>  
  1 "Albergue municipal / complejo deportivo"  
  30 Otros  
  2 "Caba\xf1as / bungalows"  
  277 Otros  
  3 "Conjunto de unidades turisticas"  
  50 Otros  
  4 "Establecimiento rural"  
 131 Otros  
  5 "Hosteria"  
 164 Otros  
  6 "Hotel 1 estrella"  
 34 Hotel hasta 3 estrellas
```

Ejercitación grupal



Ejercitación

- Necesito reagrupar la variable `clasificacion_minturdep` para quedarme con 2 categorías:
 - `Camping`
 - `Otros`

Dada la consigna anterior, llenar en el campo marcado con _____ con el código necesario para ejecutar la sentencia:

```
df_puna |>  
  select(localidad, _____) |>  
  mutate(nueva_clasificacion = case_when(_____ == "Camping" ~ "Camping",  
                                         .default = "Otros"))
```

summarise()

Resume información y realiza cálculos



summarise()

- Antes:

```
sum(df_puna$plazas)
```

```
[1] 2346290
```

summarise()

- Ahora:

```
1 df_puna |>
2   summarise(cant_plazas = sum(plazas))

# A tibble: 1 × 1
  cant_plazas
  <dbl>
1     2346290
```

summarise()

- Ahora:

```
1 df_puna |>
2   summarise(cant_plazas = sum(plazas),
3             prom_plazas = mean(plazas))
# A tibble: 1 × 2
  cant_plazas prom_plazas
     <dbl>        <dbl>
1     2346290      159.
```

summarise()

- Ahora:

```
1 df_puna |>
2   summarise(cant_plazas = sum(plazas),
3             prom_plazas = mean(plazas),
4             min_plazas = min(plazas),
5             max_plazas = max(plazas))
# A tibble: 1 × 4
  cant_plazas prom_plazas min_plazas max_plazas
        <dbl>       <dbl>      <dbl>      <dbl>
1     2346290       159.        0       20001
```

group_by()

Ayuda a ejecutar una función de forma agrupada



group_by()

- Antes:

```
1 df_puna |>
2   summarise(cant_plazas = sum(plazas),
3             prom_plazas = mean(plazas),
4             min_plazas = min(plazas),
5             max_plazas = max(plazas))

# A tibble: 1 × 4
  cant_plazas prom_plazas min_plazas max_plazas
  <dbl>        <dbl>      <dbl>      <dbl>
1     2346290       159.        0       20001
```

group_by()

- Después:

```
1 df_puna |>
2   group_by(indice_tiempo) |>
3   summarise(cant_plazas = sum(plazas),
4             prom_plazas = mean(plazas),
5             min_plazas = min(plazas),
6             max_plazas = max(plazas))

# A tibble: 3 × 5
  indice_tiempo cant_plazas prom_plazas min_plazas max_plazas
  <dbl>        <dbl>       <dbl>      <dbl>       <dbl>
1 2020         785963     161.        0       20001
2 2021         780050     159.        0       20001
3 2022         780277     158.        0       19964
```

input / output



- Toda función tiene un **input** (ingredientes 🧂🌿🧅) y un **output** (la torta 🎂)

input / output

- ¿Cuál es el output del `summarise()`?

input / output

```
# Creo objeto
tabla_puna <- df_puna |>
  group_by(indice_tiempo) |>
  summarise(cant_plazas = sum(plazas),
            prom_plazas = mean(plazas),
            min_plazas = min(plazas),
            max_plazas = max(plazas))

# Chequeo tipo de objeto
class(tabla_puna)

[1] "tbl_df"     "tbl"        "data.frame"
```

input / output

- ¿Y qué puedo hacer sobre un data frame?

input / output

```
1 tabla_puna |>  
2   filter(indice_tiempo == 2022)
```

```
# A tibble: 1 × 5  
  indice_tiempo cant_plazas prom_plazas min_plazas max_plazas  
  <dbl>        <dbl>        <dbl>        <dbl>        <dbl>  
1         2022     780277     158.          0       19964
```

input / output

```
1 tabla_puna |>  
2   filter(indice_tiempo == 2022) |>  
3   select(indice_tiempo, cant_plazas)
```

```
# A tibble: 1 × 2  
  indice_tiempo cant_plazas  
  <dbl>          <dbl>  
1       2022      780277
```

tidyverse en acción



tidyverse en acción

```
1 df_puna |>  
2   select(indice_tiempo, tipo, plazas)
```

```
# A tibble: 14,717 × 3  
  indice_tiempo tipo      plazas  
  <dbl> <chr>     <dbl>  
1 2020 Otros colectivos 30  
2 2020 Parahoteleros 277  
3 2020 Otros colectivos 50  
4 2020 Parahoteleros 131  
5 2020 Parahoteleros 164  
6 2020 Hoteleros 34  
7 2020 Hoteleros 135  
8 2020 Hoteleros 78  
9 2020 Hoteleros 60  
10 2020 Parahoteleros 15  
# i 14,707 more rows
```

tidyverse en acción

```
1 df_puna |>  
2   select(indice_tiempo, tipo, plazas) |>  
3   filter(indice_tiempo %in% c(2021, 2022))
```

```
# A tibble: 9,843 × 3  
  indice_tiempo tipo           plazas  
          <dbl> <chr>        <dbl>  
1         2021 Otros colectivos     30  
2         2021 Parahoteleros      277  
3         2021 Otros colectivos     50  
4         2021 Parahoteleros      131  
5         2021 Parahoteleros      164  
6         2021 Hoteleros          34  
7         2021 Hoteleros          135  
8         2021 Hoteleros          78  
9         2021 Hoteleros          60  
10        2021 Parahoteleros      15  
# i 9,833 more rows
```

tidyverse en acción

```
1 df_puna |>  
2   select(indice_tiempo, tipo, plazas) |>  
3   filter(indice_tiempo %in% c(2021, 2022)) |>  
4   mutate(tipo_recod = case_when(tipo == "Hoteleros" ~ "Hote"  
5                     .default = "Otros"))
```

```
# A tibble: 9,843 × 4  
  indice_tiempo tipo          plazas tipo_recod  
  <dbl> <chr>        <dbl> <chr>  
1 2021 Otros colectivos     30 Otros  
2 2021 Parahoteleros       277 Otros  
3 2021 Otros colectivos     50 Otros  
4 2021 Parahoteleros       131 Otros  
5 2021 Parahoteleros       164 Otros  
6 2021 Hoteleros           34 Hoteleros  
7 2021 Hoteleros           135 Hoteleros  
8 2021 Hoteleros           78 Hoteleros  
9 2021 Hoteleros           60 Hoteleros  
10 2021 Parahoteleros      15 Otros  
# i 9,833 more rows
```

tidyverse en acción

```
1 df_puna |>  
2   select(indice_tiempo, tipo, plazas) |>  
3   filter(indice_tiempo %in% c(2021, 2022)) |>  
4   mutate(tipo_recod = case_when(tipo == "Hoteleros" ~ "Hote")  
5           .default = "Otros")) |>  
6   group_by(indice_tiempo, tipo_recod) |>  
7   summarise(cant_plazas = sum(plazas),  
8             prom_plazas = mean(plazas))
```

```
# A tibble: 4 × 4  
# Groups:   indice_tiempo [2]  
  indice_tiempo tipo_recod cant_plazas prom_plazas  
  <dbl> <chr>          <dbl>        <dbl>  
1     2021 Hoteleros      454631       264.  
2     2021 Otros           325419       102.  
3     2022 Hoteleros      455702       264.  
4     2022 Otros           324575       101.
```

tidyverse en acción

