

Procesando datos con **{tidyverse}**

Principales herramientas (funciones) para el
tratamiento de datos



Estación R

2024-02-21



Introducción al Procesamiento de Datos con R

ESTACIÓN

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

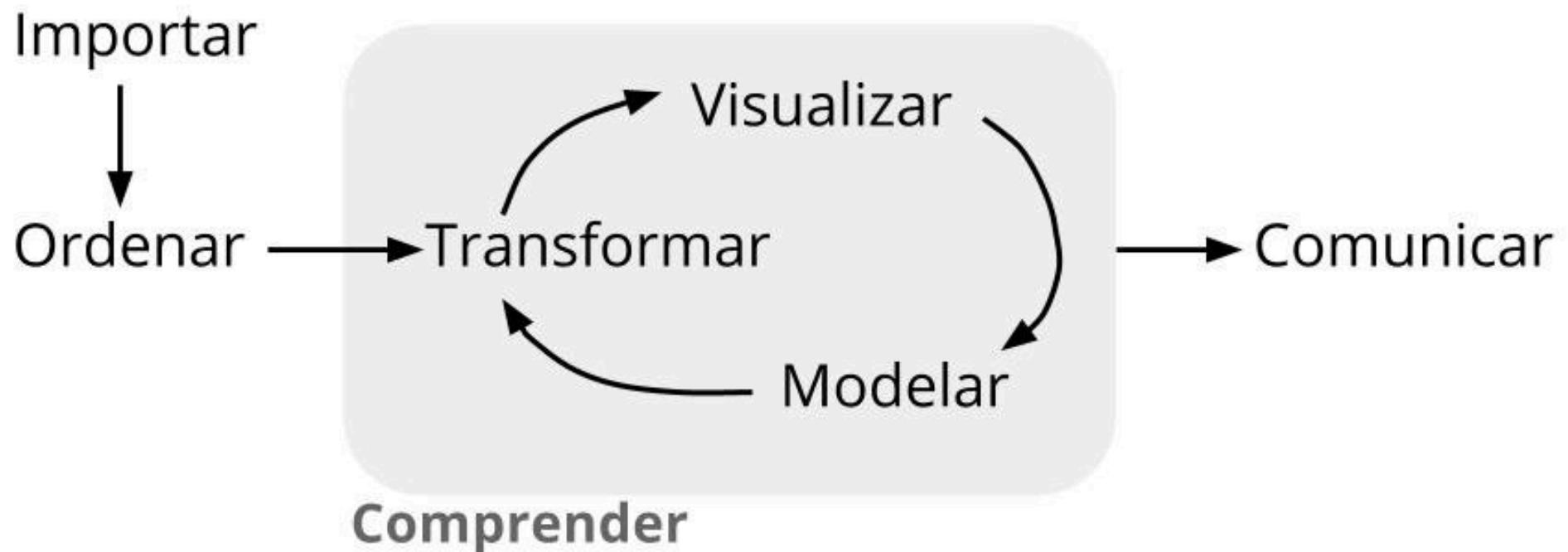


¿Qué es la Ciencia de Datos?



¿Qué es la Ciencia de Datos?

El proceso del análisis de datos



{tidyverse}

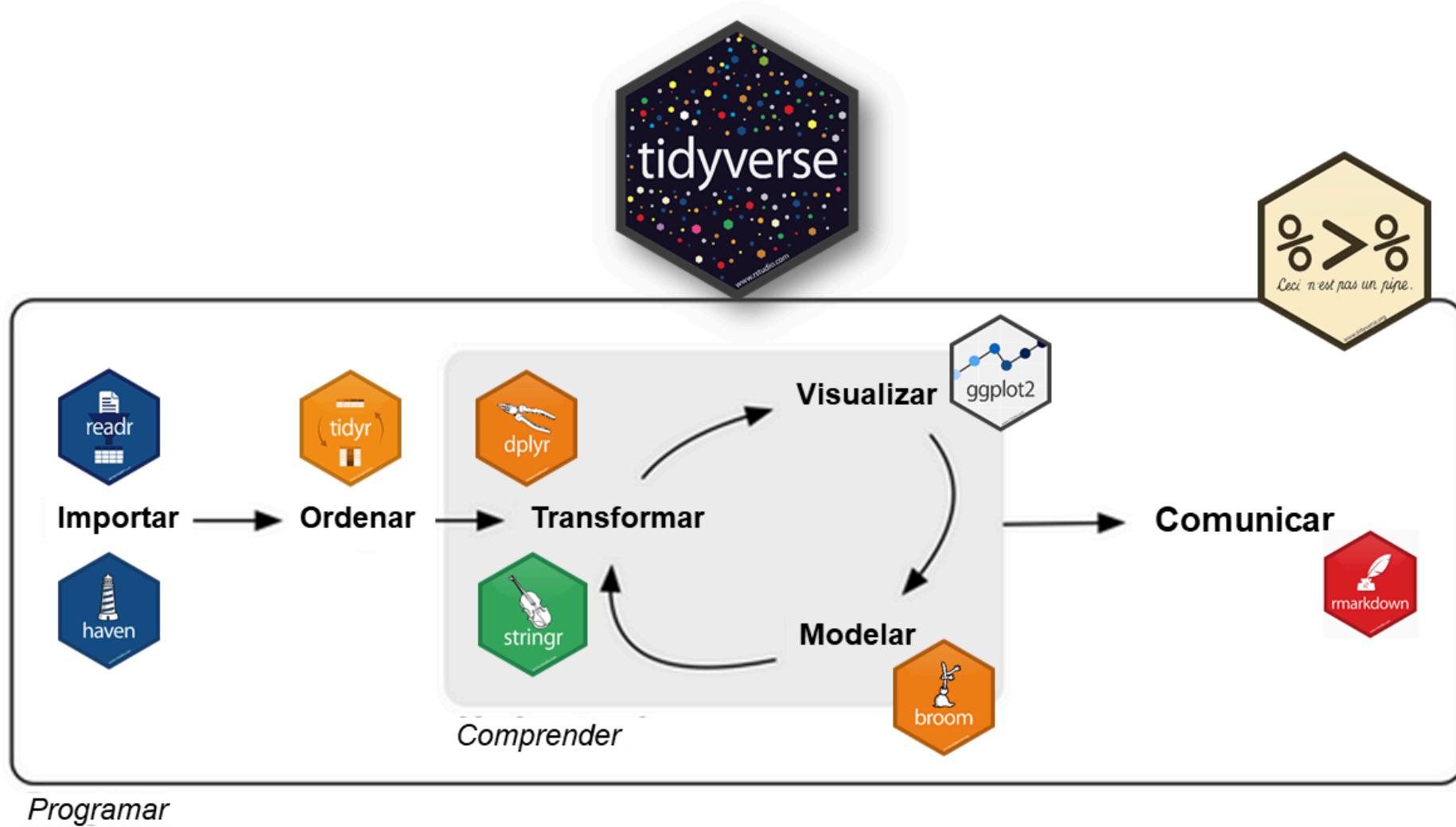


¿Qué es {tidyverse}?

- 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	✓ tidyr	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",  
                           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



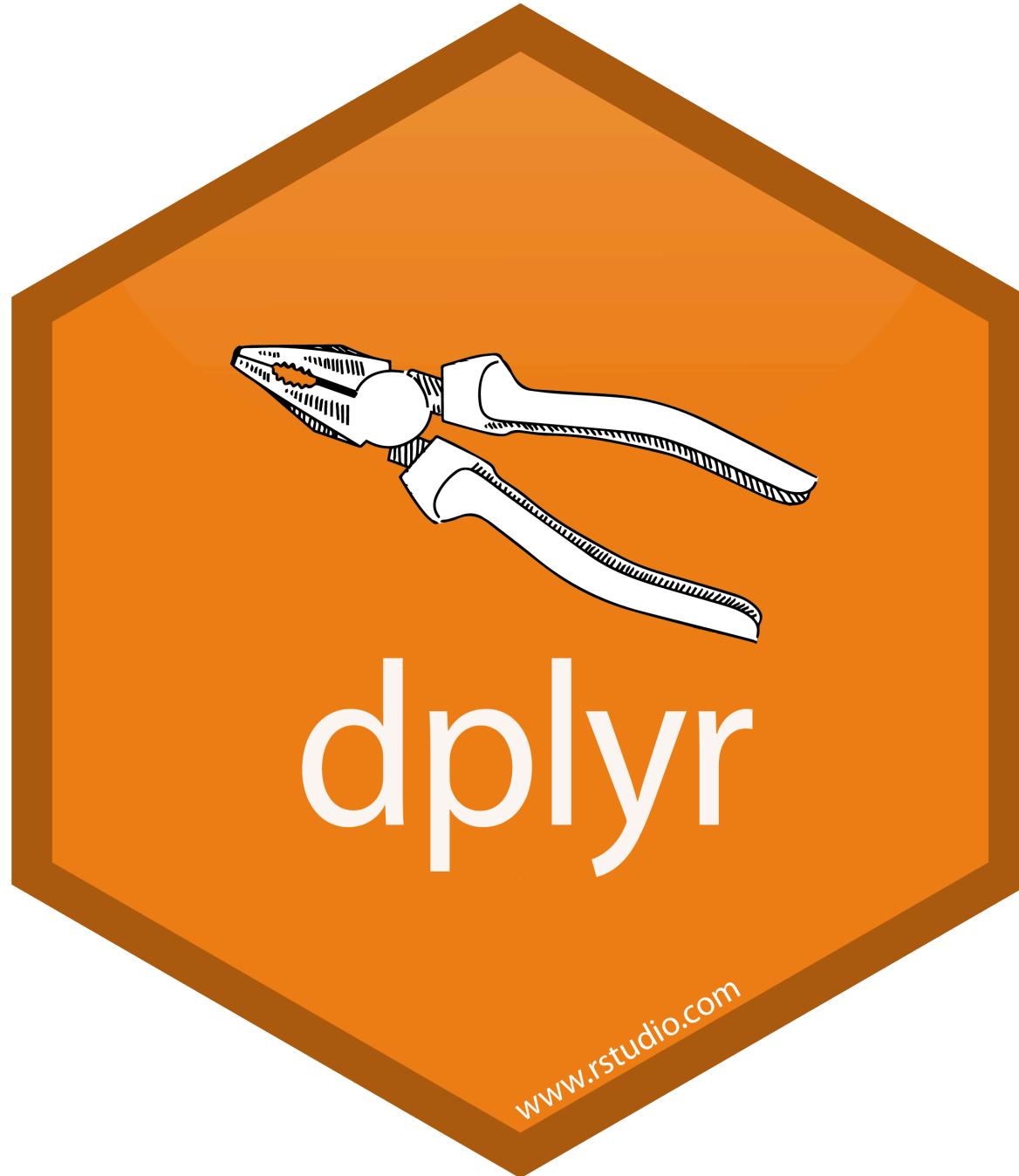
Ejemplo:

- 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>renombra 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 el resultado de la ejecución de la función `select()`. Se muestra una lista de columnas seleccionadas y un cuadro que contiene los datos resultantes.

Las columnas seleccionadas son: `id`, `nombre`, `Edad`, `localidad` y `tipo_alojamiento`.

<code>Id</code>	<code>nombre</code>	<code>Edad</code>	<code>localidad</code>	<code>tipo_alojamiento</code>
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



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

- Selecciono las 3 columnas de interés

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



Caso práctico

- Selecciono 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

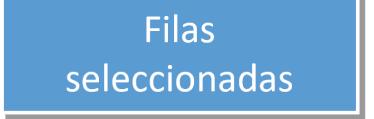


filter()

- 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



filter()

- La función tiene la siguiente estructura:

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

Condición `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 cuento 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"
```



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	Operador	Descripción
<code>==</code>	<i>igual</i>	<code>&</code>	<i>y</i> - Cuando se cumplen ambas condiciones
<code>%in%</code>	<i>incluye</i>	<code> </code>	<i>o</i> - Cuando se cumple una u otra condición
<code>!=</code>	<i>distinto</i>		
<code>></code>	<i>mayor que</i>		
<code><</code>	<i>menor que</i>		
<code>>=</code>	<i>mayor o igual que</i>		
<code><=</code>	<i>menor o igual que</i>		



