

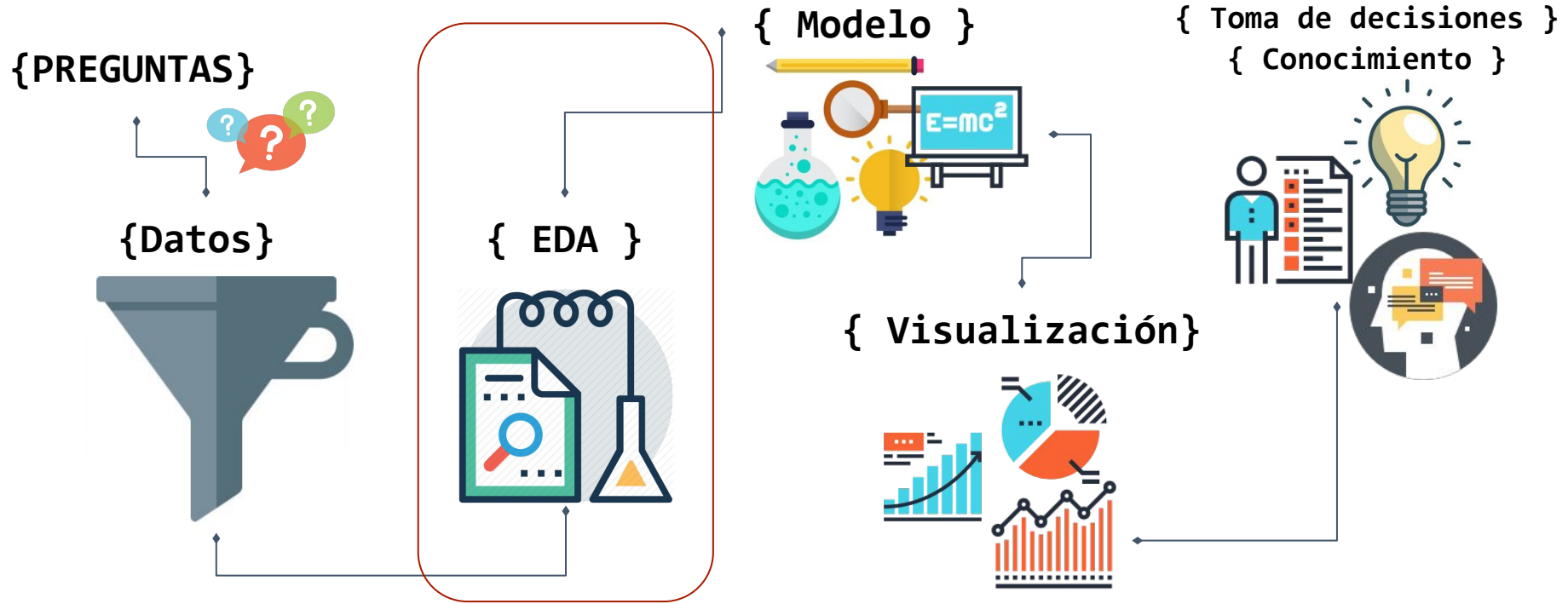
# { TidyVerse I }

# Que vamos a ver hoy?

- Repaso de funciones en R
- Paquetes de Tidyverse
- Manifiesto de Tidyverse
- Funcion Select
- Funcion Filter
- Funcion Count
- Funcion Group by
- Funcion Arrange



# PROCESO DE ANÁLISIS



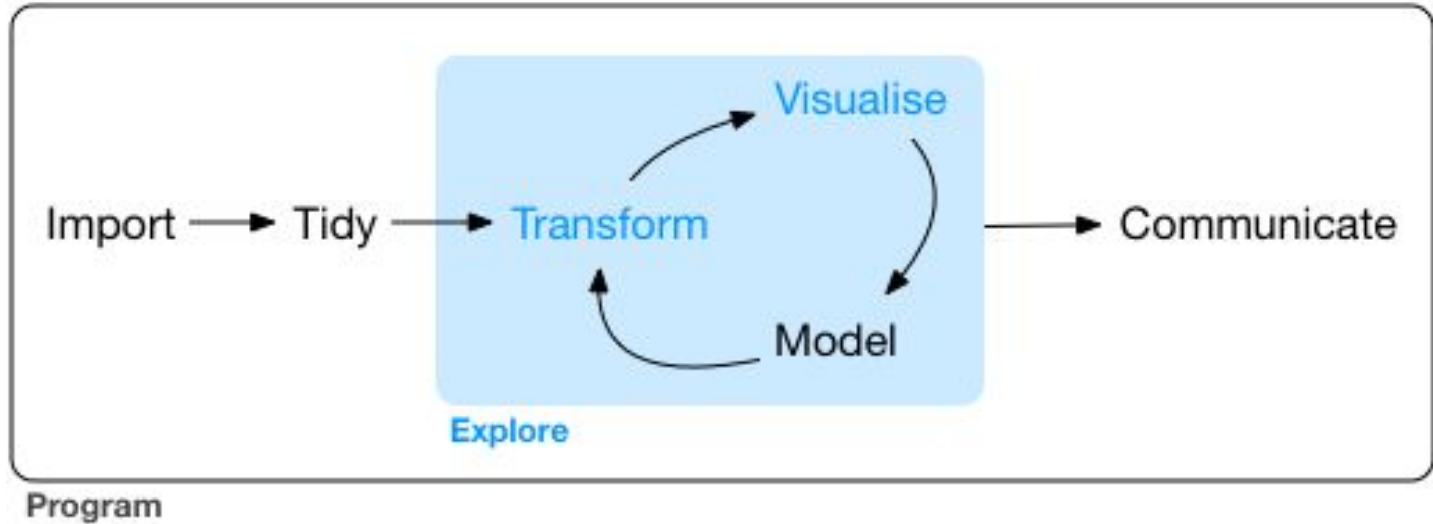
# Instalación de paquetes

Para esta clase necesitamos instalar dos paquetes:

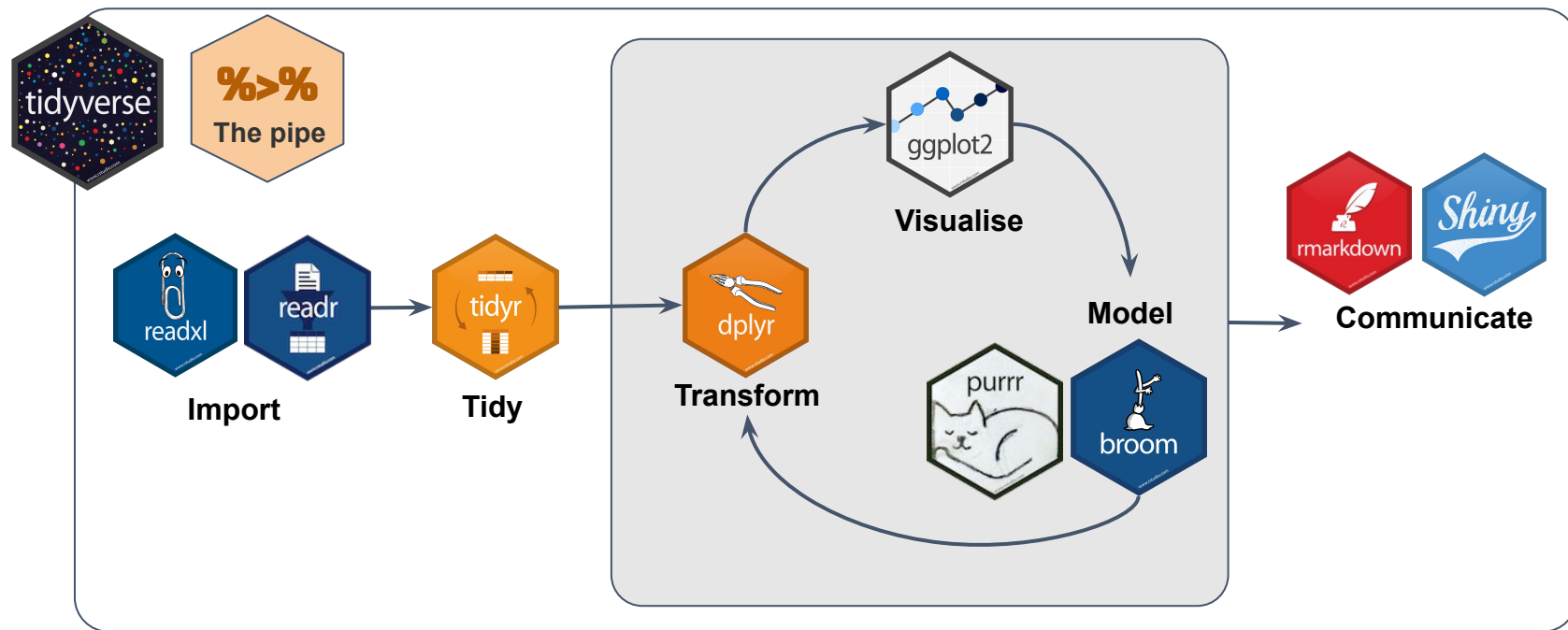
```
install.packages('tidyverse')  
install.packages('reshape2')
```



# QUE PUEDO HACER CON R?

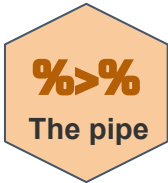


# Tidyverse: introducción y paquetes



# Operador Pipe

El operador pipe es el que permite pasar argumentos o parámetros que están a la izquierda a una función que está a la derecha.

R - BASE		TIDYVERSE
<pre>x=c(7,8,9,10,11,12,13)</pre>		<pre>library(tidyverse) x=c(7,8,9,10,11,12,13)</pre>
<pre>length(x)</pre>		<pre>x%&gt;%length()</pre>



# Tidy Data

Cada conjunto ordenado tiene que cumplir tres reglas:

- Cada variable debe tener una columna.
- Cada observación debe tener una fila.
- Cada valor debe tener su propia celda.

country	year	cases	population
Afghanistan	1999	75	15000071
Afghanistan	2000	866	20500360
Brazil	1999	3737	172006362
Brazil	2000	8488	174504898
China	1999	21258	127291272
China	2000	21676	128042583

variables

country	year	cases	population
Afghanistan	1999	75	15000071
Afghanistan	2000	866	20500360
Brazil	1999	3737	172006362
Brazil	2000	8488	174504898
China	1999	21258	127291272
China	2000	21676	128042583

observations

country	year	cases	population
Afghanistan	1999	75	15000071
Afghanistan	2000	866	20500360
Brazil	1999	3737	172006362
Brazil	2000	8488	174504898
China	1999	21258	127291272
China	2000	21676	128042583

values





# Cuatro principios de Tidyverse

Como se menciona anteriormente se utilizan varios paquetes que se complementan para poder una interfaz uniforme con la funcionalidad en su total.

Existe un [manifiesto](#) que tiene cuatro principios básicos:

1. Reutiliza las estructuras de datos existentes.
2. Combinación de muchas funciones simples por medio de pipes.
3. Adoptar la programación funcional
4. Diseño para humanos



# Funcion Filter

Filter es una de las funciones contenida en el paquete dplyr es la que permite filtrar un dataset.

Una condición sola para filtrar: `tips %>% filter(total_bill>45)`

	total_bill	tip	sex	smoker	day	time	size
1	48.27	6.73	Male	No	Sat	Dinner	4
2	48.17	5.00	Male	No	Sun	Dinner	6
3	48.33	9.00	Male	No	Sat	Dinner	4
4	50.81	10.00	Male	Yes	Sat	Dinner	3
5	45.35	3.50	Male	Yes	Sun	Dinner	3



# Funcion Filter

Filter es una de las funciones contenida en el paquete dplyr es la que permite filtrar un dataset.

Con más de una condición:

```
tips %>% filter(total_bill>45 , sex=='No')
```

	total_bill	tip	sex	smoker	day	time	size
1	48.27	6.73	Male	No	Sat	Dinner	4
2	48.17	5.00	Male	No	Sun	Dinner	6
3	48.33	9.00	Male	No	Sat	Dinner	4



# Función Arrange

Para poder realizar un ordenamiento de una variable se utiliza la función arrange:

- Ascendente : `tips %>% arrange(total_bill)`
- Descendente: `tips %>% arrange(desc(total_bill))`

Si queremos ordenar más de una variable:

- Ascendente : `tips %>% arrange(time,size)`
- Descendente: `tips %>% arrange(desc(time,size))`



# Función Mutate

La función mutate realiza la modificación de una columna:

	total_bill	tip	sex	smoker	day	time	size
1	16.99	1.01	Female	No	Sun	Dinner	2
2	10.34	1.66	Male	No	Sun	Dinner	3
3	21.01	3.50	Male	No	Sun	Dinner	3
4	23.68	3.31	Male	No	Sun	Dinner	2

	total_bill	tip	sex	smoker	day	time	size
1	16.99	1.01	Female	No	Sun	Dinner	4
2	10.34	1.66	Male	No	Sun	Dinner	6
3	21.01	3.50	Male	No	Sun	Dinner	6
4	23.68	3.31	Male	No	Sun	Dinner	4

`tips %>%`

`mutate(size = size*2)`



# Función Mutate

Asimismo esta función también puede agregar una columna:

	total_bill	tip	sex	smoker	day	time	size
1	16.99	1.01	Female	No	Sun	Dinner	2
2	10.34	1.66	Male	No	Sun	Dinner	3
3	21.01	3.50	Male	No	Sun	Dinner	3
4	23.68	3.31	Male	No	Sun	Dinner	2

	total_bill	tip	sex	smoker	day	time	size	tips_porc
1	16.99	1.01	Female	No	Sun	Dinner	2	0.05944673
2	10.34	1.66	Male	No	Sun	Dinner	3	0.16054159
3	21.01	3.50	Male	No	Sun	Dinner	3	0.16658734
4	23.68	3.31	Male	No	Sun	Dinner	2	0.13978041

tips %>%

mutate(tips\_porc = tip/total\_bill)



# Función Select

Select es una función se utiliza para seleccionar columnas del dataset:

	total_bill	tip	sex	smoker	day	time	size
1	16.99	1.01	Female	No	Sun	Dinner	2
2	10.34	1.66	Male	No	Sun	Dinner	3
3	21.01	3.50	Male	No	Sun	Dinner	3
4	23.68	3.31	Male	No	Sun	Dinner	2

	total_bill	tip	sex
1	16.99	1.01	Female
2	10.34	1.66	Male
3	21.01	3.50	Male
4	23.68	3.31	Male

tips %>%

select(total\_bill,tip,sex)



# Función Count

Count es una función que permite agrupar los datos contando los registros en base a las columnas que se seleccionan.

	sex	smoker	n
1	Female	No	54
2	Female	Yes	33
3	Male	No	97
4	Male	Yes	60

```
tips %>%  
  count (sex,smoker)
```





# Ejercicios con Open Data

- 1) Importar el archivo de Sistema Único de Atención Ciudadana 2019  
(Link: <https://bit.ly/2xCjHoL>)
- 2) Formemos equipo en base:
  - TRÁNSITO,
  - LIMPIEZA Y RECOLECCIÓN,
  - ARBOLADO Y ESPACIOS VERDES.
  - CALLES Y VEREDAS



# Ejercicios con Open Data

- 1) Generar un vector con los valores únicos sobre las subcategorías de los reclamos.
- 2) Cual es la comuna con más reclamos?
- 3) Cual es el canal por el que se realizan más reclamos?
- 4) Que tipo de registro se ingresan en mayor cantidad?, tomando como registro: sugerencia, reclamo, etc.
- 5) En base a la categoría genere un dataframe con la cantidad de registros que hay por cada subtema y exprese los valores como porcentuales.



# Ejercicios con Open Data

- 6) Cuáles son los 3 conceptos con menos reclamos? Ordenar en forma creciente.
- 7) Cuántos reclamos hay pendiente todavía? Genere un dataframe con dicho registros
- 8) Cuál es el subtema que tiene más casos que tardan más de 10 días en resolver un incidente?

