Introducción a la Epidemiología en el campo de la Salud Pública

César V. Munayco Escate, MD, MSc, MPH,DrPH
Coordinador del Curso

Libny More Huamán, MD, MSc (Mentor)
Jenny Chirinos Saire, Biol, MSc (Mentor)

Sesión 02: Estructura y limpieza de datos / Procesamiento de datos

- Visualizar o inspeccionar las bases de datos
- Normalizar los nombres de las columnas
- Identificar y eliminar duplicados
- Homogeneizar la escritura
- Indexar filas y columnas
- Renombrar variables
- Modificar valores de variables
- Crear nuevas variables
- Conocer paquetes y funciones para resumir variables numéricas y categóricas.
- Conocer paquetes y funciones para elaborar tablas que puedan ser exportadas como bases de datos.

Paquetes y funciones de la sesión

- Función `c()`
- Función `data.frame()`
- Función `clean_names()`
- Función `distinct()`
- Función `tolower() / toupper()`
- Paquetes **{dplyr}** y **{tidyverse}**
- Función `filter()`
- Función `select()`
- Función `rename()`
- Función `mutate()`

Paquetes y funciones de la sesión

- Función `clean_names()`
- Función `duplicated()`
- Función `distinct()`
- Función `head()`
- Función `str()`
- Función `glimpse()`
- Función `select()` / `filter()`
- Función `rename()`
- Función `arrange()`
- Función `mutate()` / `ifelse()` / `case_when()`
- Función `grup_byb()` / `summarise()` / `count ()`

Paquetes y funciones de la sesión

- Función `summary()`
- Funciones `count()` / `sum()` / `round()`
- Funciones `group_by()`, `summarise()`
- Función `mean()` / `median()`
- Función `max()` / `min()`
- Función `sd()` / `IQR()`
- Función `tbl_summary()`

Tipos de datos

Tipo	**Ejemplo**	**Nombre en inglés**
Entero	1	*integer*
Numérico	1.3	*numeric*
Cadena de texto	“uno”	*character*
Factor	uno	*factor*
Lógico	TRUE	*logical*
Perdido	NA	*NA*
Vacio	NULL	*null*

¿Qué es un dataframe?

- Estructuras de datos de **dos dimensiones** (como una tabla)
- Constan de columnas que corresponden a variables o atributos, y de filas que corresponden a los casos o registros individuales
- Las columnas de un dataframe tienen las propiedades de los vectores, y la condición de que dichos vectores tienen la misma longitud
- Muy útil para realizar análisis de datos

¿Cómo se crea un dataframe?

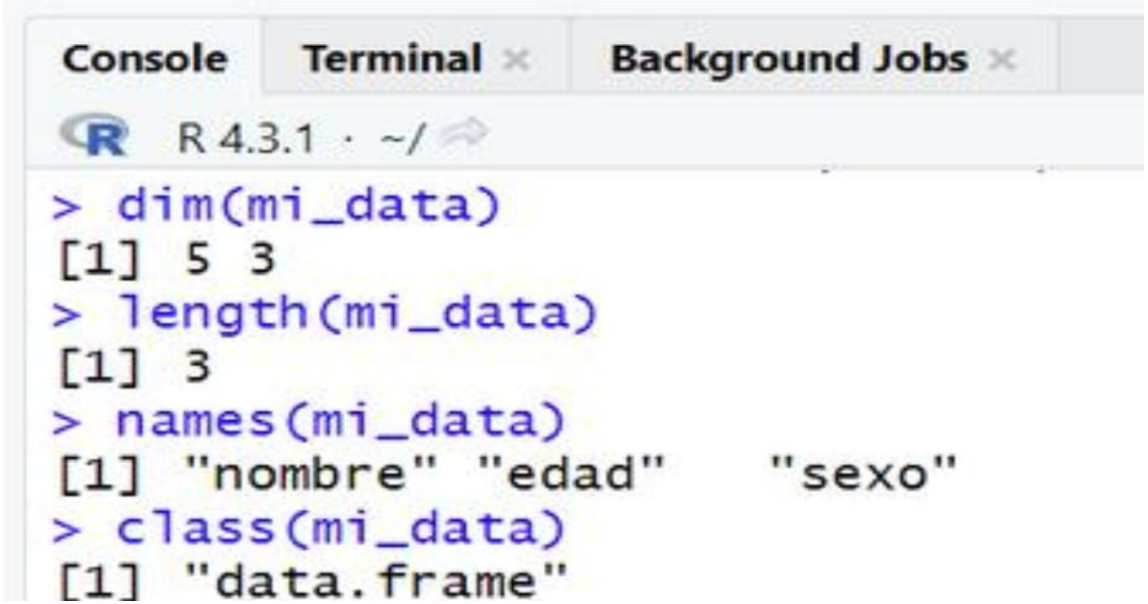
Se crean dataframes usando la función data.frame()

```
nombre <- c('Jose', 'Maria', 'Luis', 'Martha')
edad <- c(22, 25, 28, 25)
sexo <- c('masculino', 'femenino', 'masculino', 'femenino')
mi_data <- data.frame(nombre, edad, sexo)
```

	nombre	edad	sexo
1	Jose	22	masculino
2	Maria	25	femenino
3	Luis	28	masculino
4	Martha	25	femenino

Funciones para manejar dataframes

- `dim()`, informa las dimensiones del dataframe
- `length()`, informa del número de variables que conforma el dataframe
- `names()`, permite ver los nombres de las variables
- `class()`, informa el tipo de estructura



Visualización de datos

Examinar el conjunto de datos:

- **Función `head()` [de R base]**
- **Función `str()` [de R base]**
- **Función `glimpse()` [del paquete `tidyverse`]**

str()

Muestra la estructura interna de cualquier objeto R de forma compacta.

```
> str(eda)
'data.frame':   100000 obs. of  25 variables:
 $ ano        : num  2021 2016 2021 2023 2019 ...
 $ semana     : num  23 51 41 16 1 14 25 18 40 44 ...
 $ sub_reg_nt: chr  "46" "08" "18" "02" ...
 $ red        : chr  "14" "07" "03" "01" ...
 $ microred   : chr  "01" "01" "03" "06" ...
 $ e_salud    : chr  "060614A301" "080106A101" "180205A301"
 ...
 $ ubigeo     : chr  "060614" "081301" "180205" "020506" ...
 $ daa_c1     : num  1 0 0 0 0 10 0 2 0 0 ...
 $ daa_c1_4   : num  0 0 1 0 0 31 1 1 0 0 ...
 $ daa_c5     : num  0 1 0 2 1 65 2 0 0 1 ...
 $ daa_d1     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ daa_d1_4   : num  0 0 0 0 0 0 0 0 0 0 ...
```

glimpse()

Describe el número de observaciones y variables.

Muestra la designación del tipo de datos de las variables.

```
> glimpse(eda)
Rows: 100,000
Columns: 25
$ ano        <dbl> 2021, 2016, 2021, 2023, 2019,
$ semana     <dbl> 23, 51, 41, 16, 1, 14, 25, 18
$ sub_reg_nt <chr> "46", "08", "18", "02", "16",
$ red        <chr> "14", "07", "03", "01", "08",
$ microred   <chr> "01", "01", "03", "06", "33",
$ e_salud    <chr> "060614A301", "080106A101", "
$ ubigeo     <chr> "060614", "081301", "180205",
$ daa_c1     <dbl> 1, 0, 0, 0, 0, 10, 0, 2, 0, 0
$ daa_c1_4   <dbl> 0, 0, 1, 0, 0, 31, 1, 1, 0, 0
$ daa_c5     <dbl> 0, 1, 0, 2, 1, 65, 2, 0, 0, 1
```

<fct>: variable factorial, también conocida como variable categórica.

<int>: variable cuantitativa que toma solo valores enteros.

<dbl>: doble precisión, variable cuantitativa continua (toma valores decimales).

<chr>: variable categórica, texto.

<dttm>: fecha

Normalizar los nombres de las columnas

Función clean_names()

Convierte nombres a minúsculas, reemplaza espacios con guiones bajos, y elimina caracteres especiales, hace nombres únicos.

Paquete: janitor

```
# Instalar y cargar el paquete janitor
install.packages("janitor")
library(janitor)

# Normalizar nombres de columnas
df <- df %>% clean_names()
```

Ejemplo:
Nombres como "**First Name**" pasan a ser "**first_name**".

Identificar y eliminar duplicados

Función distinct ( )

Elimina filas duplicadas en un conjunto de datos.

Uso típico: Elimina filas completamente duplicadas o duplicados en una columna específica.

```
# Eliminar filas duplicadas
df <- df %>% distinct()
```

Homogenizar escritura

A veces, los datos de texto (nombres, categorías, etc.) vienen en diferentes formatos (mayúsculas, minúsculas, combinación). Es importante homogenizar la escritura para evitar problemas durante el análisis (por ejemplo, evitar que "Juan" y "juan" se consideren distintos).

Función tolower ( ): Convertir a minúscula

```
df$nombre <- tolower(df$nombre)
```

Función toupper ( ): Convertir a mayúscula

```
df$nombre <- toupper(df$nombre)
```

Paquete tidyverse

tidyverse es un conjunto de paquetes de R que permite que el análisis y la manipulación de los datos sean más eficientes, legibles y eficaces.

Algunos de los paquetes más destacados de tidyverse incluyen:

- **ggplot2** -> gráficos y visualizaciones de datos.
- **dplyr** -> manipulación y transformación de datos.
- **tidyr** -> datos en formato "tidy" (ordenados y estructurados).
- **readr** -> leer y escribir datos de manera eficiente.
- **purrr** -> funciones y su aplicación de manera consistente.
- **tibble** -> crear data frames modernos.

Funciones de tidyverse (paquete dplyr)

- filter ( ) → Filtra datos de acuerdo a condiciones específicas.
- select ( ) → Selecciona columnas específicas del conjunto de datos.
- mutate ( ) → Crea nuevas columnas o modifica las existentes.
- arrange ( ) → Ordena datos según los valores de una columna.
- count ( ) → Cuenta la frecuencia de observaciones en un conjunto de datos
- summarize ( ) → Resumen estadístico de los datos.
- group_by( ) → Agrupa datos por columnas para realizar operaciones por grupo.
- left_join ( ) → Realiza una unión izquierda entre dos conjuntos de datos
- inner_join ( ) → Realiza una unión interna entre dos conjuntos de datos

Indexar filas y columnas

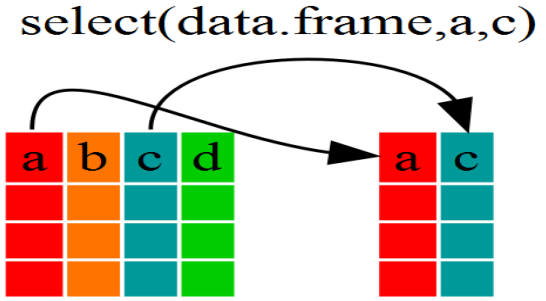
Permite seleccionar columnas y filas específicas para un análisis más eficiente.

Función select ( )

Extrae una o más variables (columnas) de una base de datos(data frame).

Se recomienda usar esta función al inicio para seleccionar solo las columnas necesarias para el análisis.

Sintaxis



Función select ( )

Existen algunos símbolos y funciones que se pueden usar dentro de la función select( ) y que ayudan a hacer una selección más eficiente.

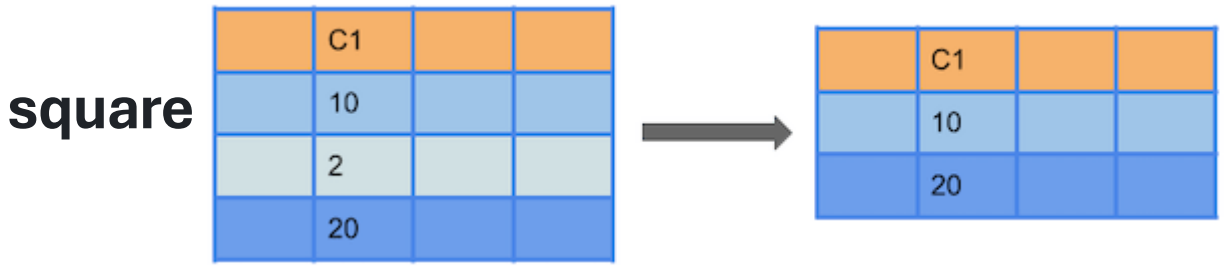
Funciones de ayuda	Uso
-	Excepto la columna
:	Selecciona un rango de variables
last_col ( )	Selecciona la ultima variable
contains( )	Selecciona variables que contienen un valor cadena
start_with	Selecciona variables que empiezan con un prefijo
ends_with	Selecciona variables que finalizan con un sufijo
one_of()	Selecciona columnas cuyos nombres están dentro del vector

Función filter ( )

Permite seleccionar observaciones (filas) de una base de datos (data frame) a partir de una o varias condiciones

Sintaxis

```
data_frame %>%
        filter(condición_1, condición_2, ...condición_n)
```




```
square%>%
    filter(c1 %in% c(10, 20))
```

Función filter ( )

Existen algunos operadores relacionales y lógicos útiles para esta función:

Funciones de ayuda	Uso
== , !=	Igual a..., diferente de...
> , <	Mayor, menor
>=, <=	Mayor o igual, menor o igual
is.na( )	El valor de un dato faltante (NA)
!is.na( )	El valor de un dato NO faltante (no es NA)
%in%	Incluye a...
&	Y (adicion)
$, |	

Renombrar variables

Función rename ( )

Permite **cambiar** el nombre de una variable

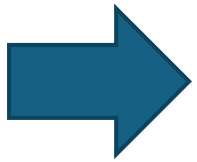
```
depresion %>%
        rename(nombre=paciente)
```

Nombre nuevo

Nombre que quieres cambiar

depresion

paciente	edad	sexo	puntaje
Maria	24	femenino	15
Juanito	18	masculino	8
Adriana	13	femenino	4
Gabriel	31	masculino	21
Jorge	28	masculino	10
Andre	35	masculino	16



nombre	edad	sexo	puntaje
Andre	35	masculino	16
Adriana	13	femenino	4
Gabriel	31	masculino	21
Jorge	28	masculino	10
Juanito	18	masculino	8
Maria	24	femenino	15

Ordenar variables

Función arrange ( ) :

Ordena las filas de un data frame según una o mas columnas:

Sintaxis

```
data_frame %>%
        arrange(columna_1, columna_2, ...columna_n) → orden ascendente por defecto
```

```
data_frame %>%
        arrange(desc(columna_1, columna_2, ...columna_n)) → orden desccendente
```

Modificar valores de variables

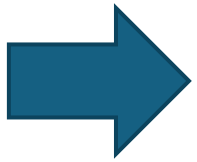
Función mutate ( )

Dentro de mutate se pueden aplicar funciones ya conocidas

```
residencia%>%
    mutate(DEPARTAMENTO = tolower(DEPARTAMENTO))
```

residencia

	DEPARTAMENTO	FEMENINO	MASCULINO
1	AMAZONAS	1831	1415
2	ANCASH	4264	3351
3	APURIMAC	2252	1723
4	AREQUIPA	10485	10356
5	AYACUCHO	1311	941
6	CAJAMARCA	2533	2441
7	CALLAO	3140	2378
8	CUSCO	2876	2467
9	HUANCAVELICA	1483	1216
10	HUANUCO	1695	1216



	DEPARTAMENTO	EMENINO	MASCULINO
1	amazonas	1831	1415
2	ancash	4264	3351
3	apurimac	2252	1723
4	arequipa	10485	10356
5	ayacucho	1311	941
6	cajamarca	2533	2441
7	callao	3140	2378
8	cusco	2876	2467
9	huancavelica	1483	1216
10	huanuco	1695	1216

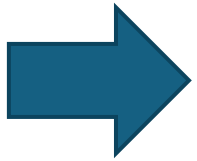
Función mutate ( )

Modificar una variable existente

```
peso%>%
    mutate(PESO = PESO * 1000)
```

Peso (kg.)

	nombre	peso	talla	edad
1	MARIA	45	1.53	25
2	JAIME	68	1.70	16
3	ANITA	53	1.58	31
4	ROBERTO	72	1.72	32
5	ARMANDO	80	1.78	27
6	JAVIER	110	1.80	37
7	LUCIA	64	1.65	14
8	ISABELA	62	1.61	26
9	RENZO	76	1.76	36
10	CARLOTA	58	1.59	22



Peso (gr.)

	nombre	peso	talla	edad
1	MARIA	45000	1.53	25
2	JAIME	68000	1.70	16
3	ANITA	53000	1.58	31
4	ROBERTO	72000	1.72	32
5	ARMANDO	80000	1.78	27
6	JAVIER	110000	1.80	37
7	LUCIA	64000	1.65	14
8	ISABELA	62000	1.61	26
9	RENZO	76000	1.76	36
10	CARLOTA	58000	1.59	22

mutate()

Se utiliza para crear nuevas variables o modificar las existentes dentro de un ***dataframe*** (base de datos). Es esencial para la manipulación de datos, ya que permite transformar y generar columnas de manera eficiente. Permite:

- **Añadir nuevas columnas sin alterar las existentes**.
- **Modificar el contenido de columnas actuales**.
- **Incluir cálculos, transformaciones y funciones complejas**, en la creación de variables.

	DEPARTAMENTO	FEMENINO	MASCULINO
1	AMAZONAS	1831	1415
2	ANCASH	4264	3351
3	APURIMAC	2252	1723
4	AREQUIPA	10485	10356
5	AYACUCHO		
6	CAJAMARCA		

	DEPARTAMENTO	FEMENINO	MASCULINO	TOTAL
1	AMAZONAS	1831	1415	3246
2	ANCASH	4264	3351	7615
3	APURIMAC	2252	1723	3975
4	AREQUIPA	10485	10356	20841
			941	2252
			2441	4974

```
residencia %>%
        mutate(TOTAL = FEMENINO + MASCULINO)
```

mutate()

Dentro de **mutate** se pueden usar funciones ya conocidas para modificar variables

```
dengue_03 <- dengue_02 %>%
      mutate(
      departamento = toupper(departamento)
      )
```

```
> dengue_02
  edad sexo departamento tipo_dx
1   31    F       loreto       S
2   24    M      ucayali       C
3   52    M         lima       S
4   41    M       loreto       C
5   18    F      huanuco       C
6   30    F      ucayali       C
```

```
> dengue_03
  edad sexo departamento tipo_dx
1   31    F       LORETO       S
2   24    M      UCAYALI       C
3   52    M         LIMA       S
4   41    M       LORETO       C
5   18    F      HUANUCO       C
6   30    F      UCAYALI       C
```

mutate()

También se pueden crear nuevas variables, empleando funciones adicionales

```
dengue_02 <- dengue_01 %>%
        mutate(
        id = row_number()
        )
```

edad	sexo	departamento	tipo_dx	fecha_notificacion	id
31	F	Loreto	S	2025-01-10	1
24	M	Ucayali	C	2025-01-11	2
52	M	Lima	S	2025-01-10	3
41	M	Loreto	C	2025-01-09	4
18	F	Huanuco	C	2025-01-10	5
30	F	Ucayali	C	2025-02-13	6

Función mutate ( ) : Otros usos importantes

Cambiar el tipo de variable:

1. **`as.numeric()`:** Cambia a formato ***numeric***
2. **`as.character()`:** Cambia a formato ***character***
3. **`as.Date()`:** Cambia a formato ***Date (fecha)***
4. **`as.logical()`:** Cambia a formato ***logical***

```
dengue_03 <- dengue_02 %>%
     mutate(
     fecha_notificacion= as.Date(fecha_notificacion)
     )
```

mutate()

Existen algunos operadores relacionales y lógicos útiles para esta función:

Funciones de ayuda	Uso
== , !=	Igual a..., diferente de...
> , <	Mayor, menor
>=, <=	Mayor o igual, menor o igual
is.na( )	El valor de un dato faltante (NA)
!is.na( )	El valor de un dato NO faltante (no es NA)
%in%	Incluye a...
&	Y (adición)

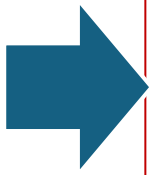
if_else()

Permite crear nuevas columnas basadas en condiciones específicas.

Se emplea para variables categóricas dicotómicas.

```
dengue_04 <- dengue_03 %>%
        mutate(
        sexo = if_else(sexo=="F", "femenino", "masculino"))
```

```
> dengue_03
  edad sexo departamento tipo_dx
1   31    F       LORETO       S
2   24    M     UCAYALI        C
3   52    M        LIMA        S
4   41    M       LORETO       C
5   18    F      HUANUCO       C
6   30    F      UCAYALI       C
```



```
> dengue_04
  edad      sexo departamento tipo_dx
1   31  femenino       LORETO       S
2   24 masculino     UCAYALI        C
3   52 masculino        LIMA        S
4   41 masculino       LORETO       C
5   18  femenino      HUANUCO       C
6   30  femenino      UCAYALI       C
```

Función mutate ( ) – if_else( )

Genera una variable binaria de acuerdo a una condición

```
depresion%>%
        mutate(edad_cat = if_else(edad > 17, “mayor”, “menor”))
```

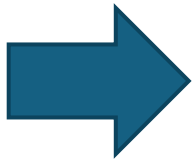
Condición

Valor si es que **SI** se cumple la condición

Valor si es que **NO** se cumple la condición

depresion

paciente	edad	sexo	puntaje
Maria	24	femenino	15
Juanito	18	masculino	8
Adriana	13	femenino	4
Gabriel	31	masculino	21
Jorge	28	masculino	10
Andre	35	masculino	16



paciente	edad	sexo	puntaje	edad_cat
Maria	24	femenino	15	mayor
Juanito	18	masculino	8	mayor
Adriana	13	femenino	4	menor
Gabriel	31	masculino	21	mayor
Jorge	28	masculino	10	mayor
Andre	35	masculino	16	mayor

case_when()

Se utiliza para crear nuevas variables condicionadas con base en múltiples condiciones.

Permite asignar valores a una variable en función de diferentes criterios, facilitando la categorización.

```
## case_when()
dengue_06 ← dengue_05 %>%
  mutate(tipo_dx = case_when(
    tipo_dx=="C" ~ "confirmado",
    tipo_dx=="S" ~ "sospechoso",
    tipo_dx=="P" ~ "probable"
    ))
```

dengue_05

edad	sexo	departamento	tipo_dx	fec
31	femenino	LORETO	S	
24	masculino	UCAYALI	C	
52	masculino	LIMA	S	
41	masculino	LORETO	C	
18	femenino	HUANUCO	C	
30	femenino	UCAYALI	P	

dengue_06

edad	sexo	departamento	tipo_dx
31	femenino	LORETO	sospechoso
24	masculino	UCAYALI	confirmado
52	masculino	LIMA	sospechoso
41	masculino	LORETO	confirmado
18	femenino	HUANUCO	confirmado
30	femenino	UCAYALI	probable

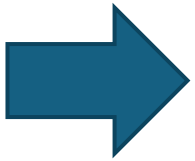
Función mutate ( ) – case_when( )

Permite categorizar variables

```
depresion %>%
mutate(gravedad= case_when( puntaje <= 7   ~ “normal”,
                            puntaje <= 12  ~ “depresión menor”,
                            puntaje <= 17  ~ “depresión
moderada”,
                            puntaje <= 29  ~ “depresión mayor))
```

depresion

paciente	edad	sexo	puntaje
Maria	24	femenino	15
Juanito	18	masculino	8
Adriana	13	femenino	4
Gabriel	31	masculino	21
Jorge	28	masculino	10
Andre	35	masculino	16



paciente	edad	sexo	puntaje	gravedad
Maria	24	femenino	15	depresión moderada
Juanito	18	masculino	8	depresión menor
Adriana	13	femenino	4	normal
Gabriel	31	masculino	21	depresión mayor
Jorge	28	masculino	10	depresión menor
Andre	35	masculino	16	depresión moderada

group_by()

Agrupa un conjunto de filas seleccionado en un conjunto de filas de resumen de acuerdo con los valores de una o más columnas o expresiones.

```
data_frame %>%
  group_by(city)
```

city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

group_by() y summarise()

Las estadísticas se calculan para cada grupo de outcome. Observa cómo se trasladan las columnas de agrupación al nuevo dataframe.

Código

```
# estadísticas resumidas de linelist agrupados
linelist %>%
  group_by(outcome) %>%
  summarise(
    n_cases  = n(),
    mean_age = mean(age_years, na.rm=T),
    max_age  = max(age_years, na.rm=T),
    min_age  = min(age_years, na.rm=T),
    n_males    = sum(gender == "m", na.rm=T))
```

Resultados

```
# A tibble: 3 × 6
  outcome n_cases mean_age max_age min_age n_males
  <chr>     <int>    <dbl>   <dbl>   <dbl>   <int>
1 Death      2582     15.9      76       0    1228
2 Recover    1983     16.1      84       0     950
3 <NA>       1323     16.2      69       0     625
```

group_by() y summarise()

```
data_frame %>%
        group_by(city) %>%
        summarise(mean = mean(amount),
                  sum = sum(amount),
                  n = n())
```

city	particle size	amount (μg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	mean	sum	n
New York	18.5	37	2
London	19.0	38	2
Beijing	88.5	177	2

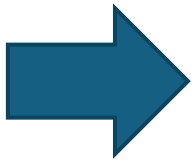
count()

Devuelve un nuevo data frame con dos columnas: las variables de agrupación y una columna llamada n que contiene el recuento de observaciones para cada grupo.

Ejemplo 1:

```
data_frame %>%
        count(METODODX)
```

	METODODX	EDAD	SEXO
1	PCR	19	FEMENINO
3	AG	27	FEMENINO
4	AG	27	FEMENINO
5	AG	25	FEMENINO
7	PCR	16	FEMENINO
2	PCR	15	MASCULINO
6	AG	16	MASCULINO
8	AG	23	MASCULINO
9	AG	28	MASCULINO
10	AG	20	MASCULINO



	METODODX	n
1	AG	7
2	PCR	3

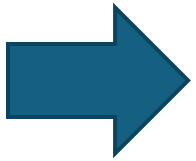
count()

Devuelve un nuevo data frame con dos columnas: las variables de agrupación y una columna llamada n que contiene el recuento de observaciones para cada grupo.

Ejemplo 2:

```
data_frame %>%
        count(METODODX, SEXO)
```

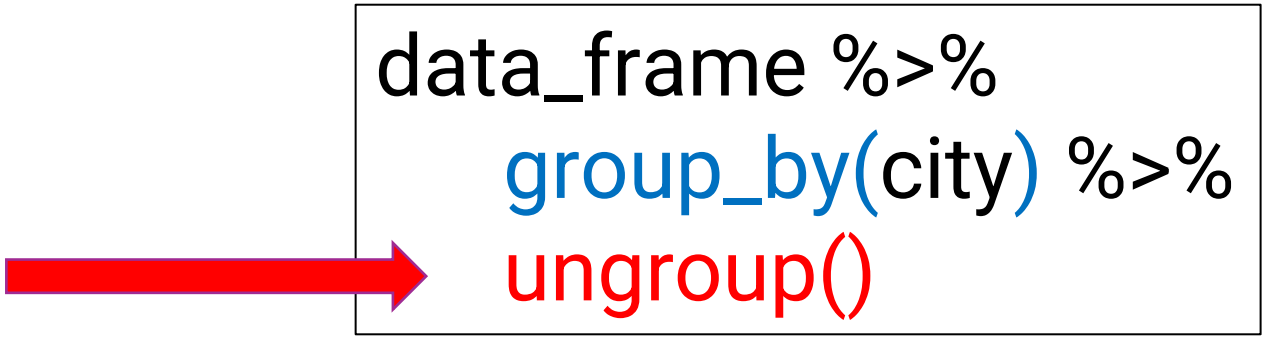
	METODODX	EDAD	SEXO
1	PCR	19	FEMENINO
3	AG	27	FEMENINO
4	AG	27	FEMENINO
5	AG	25	FEMENINO
7	PCR	16	FEMENINO
2	PCR	15	MASCULINO
6	AG	16	MASCULINO
8	AG	23	MASCULINO
9	AG	28	MASCULINO
10	AG	20	MASCULINO



	METODODX	SEXO	n
1	AG	FEMENINO	3
2	AG	MASCULINO	4
3	PCR	FEMENINO	2
4	PCR	MASCULINO	1

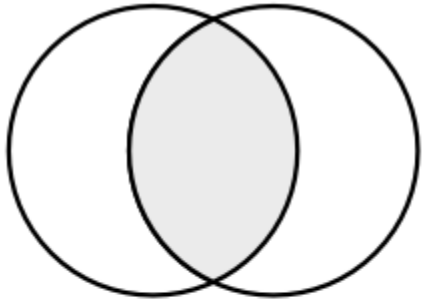
ungroup()

Se utiliza para eliminar el agrupamiento de un data frame. Cuando trabajas con datos agrupados usando group_by(), ungroup() revierte este comportamiento, permitiendo operaciones que se aplican al data frame completo en lugar de por grupos

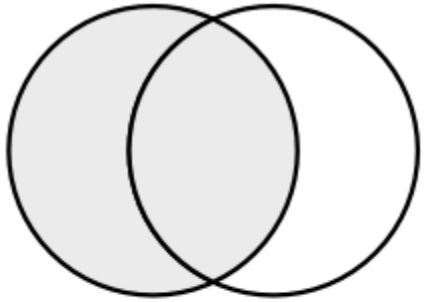


Funciones para unir bases de datos

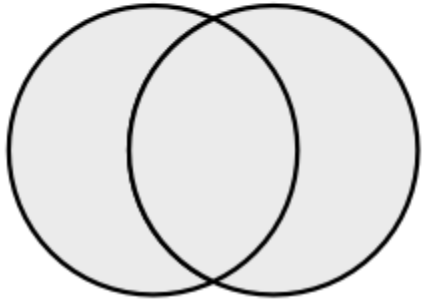
Combinar / unir bases de datos



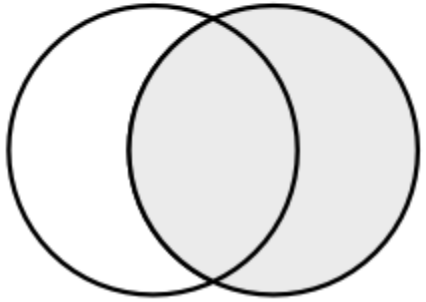
inner_join(x, y)



left_join(x, y)



full_join(x, y)

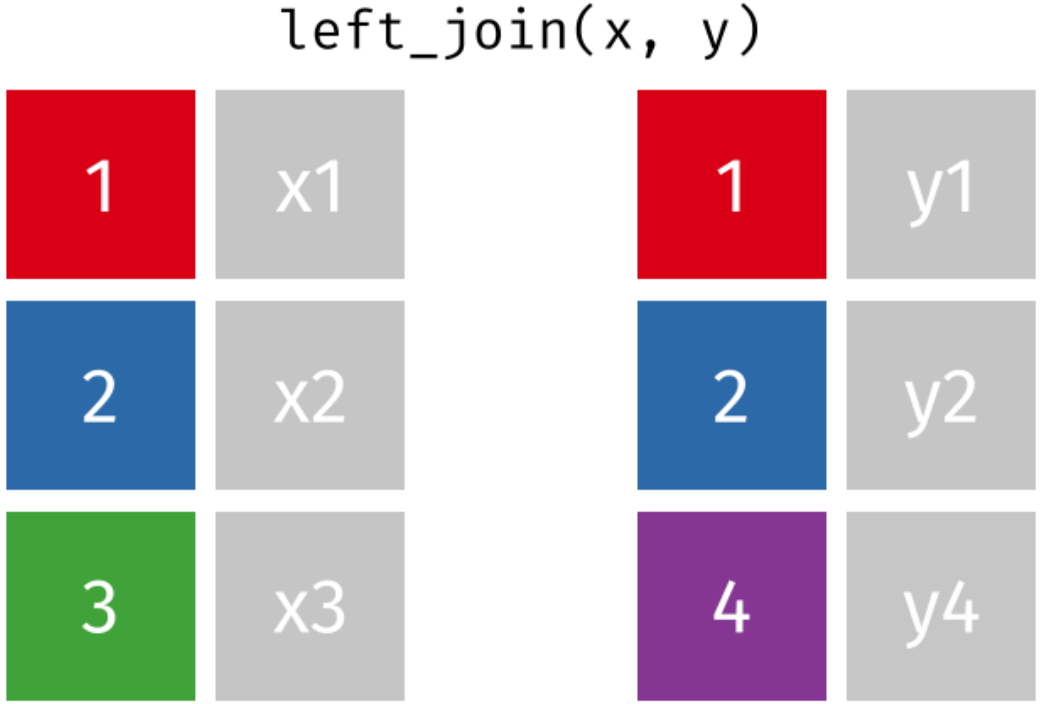


right_join(x, y)

Uniones dplyr

`left_join()`

Conserva la primera base mencionada.

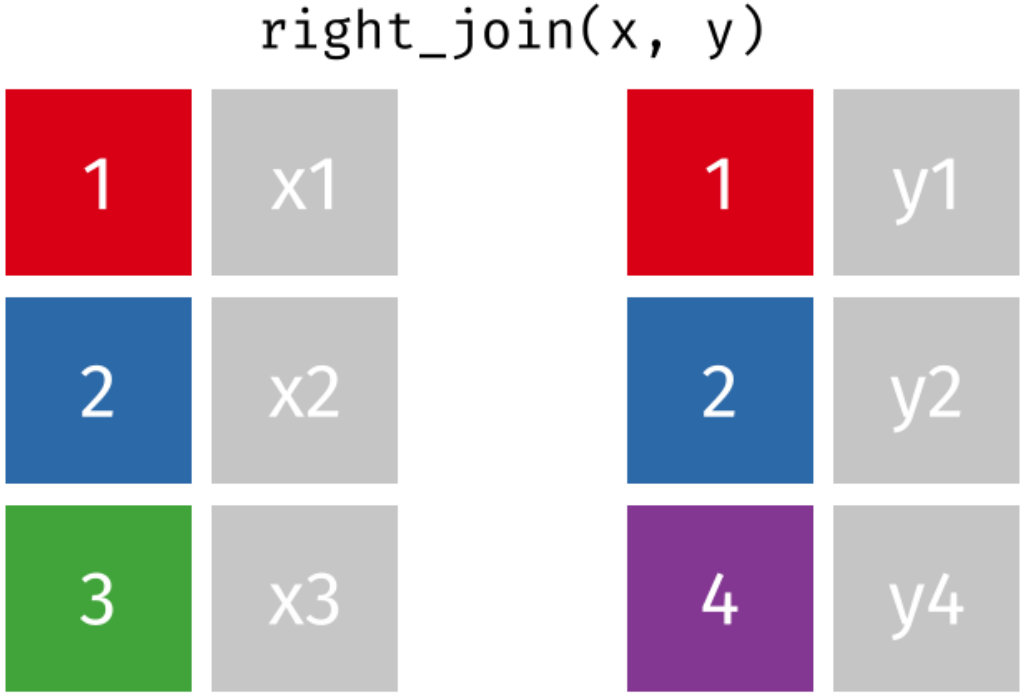


Fuente: https://github.com/gadenbuie.

Uniones dplyr

`right_join()`

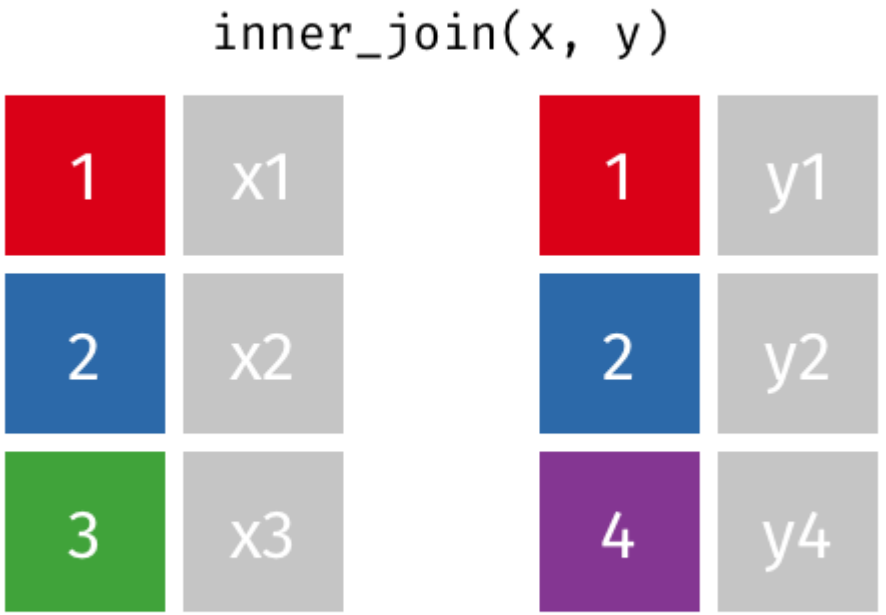
Conserva la segunda base mencionada



Fuente: https://github.com/gadenbuie.

Uniones dplyr

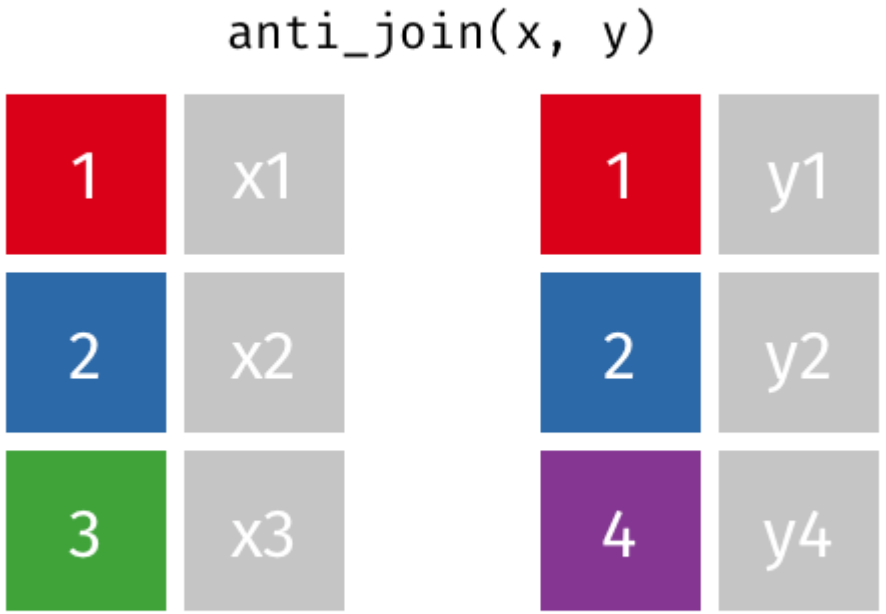
`inner_join()`



Fuente: https://github.com/gadenbuie.

Uniones dplyr

`anti_join()`

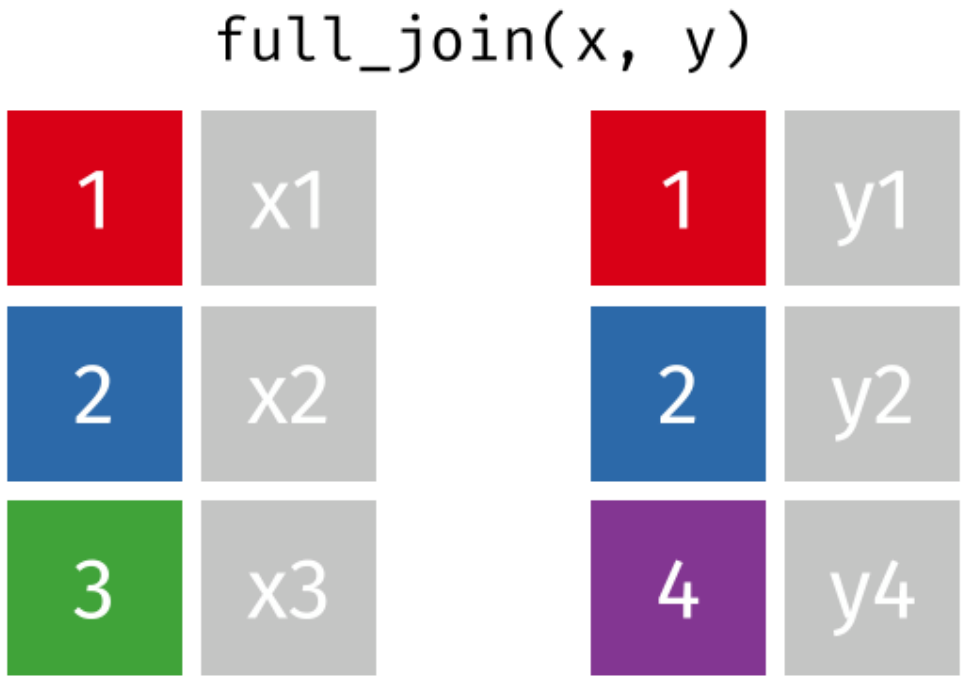


Fuente: https://github.com/gadenbuie.

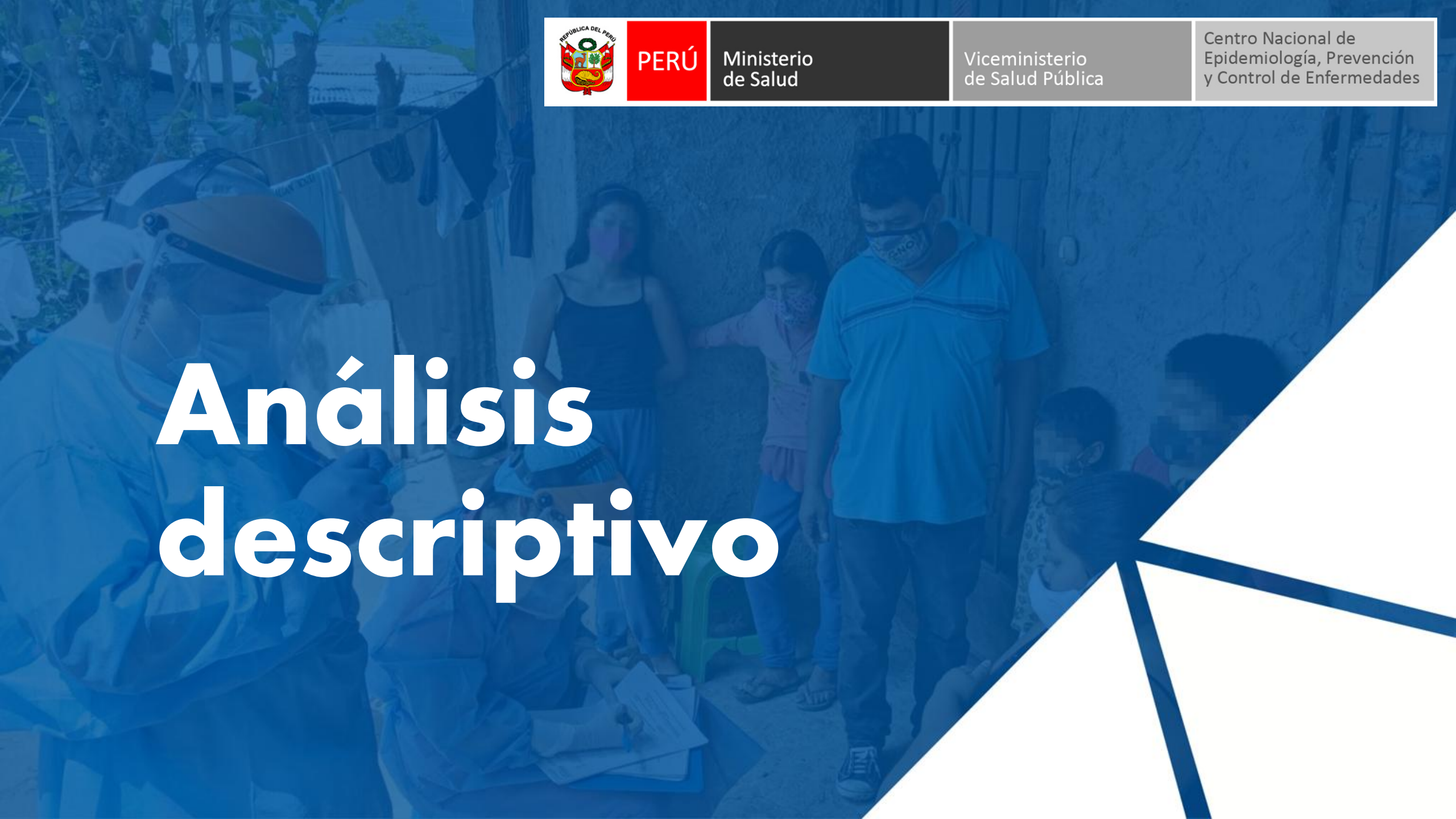
Uniones dplyr

`full_join()`

Conserva todas las filas de las bases.



Fuente: https://github.com/gadenbuie.



Análisis descriptivo

Reconocimiento del tipo de variable en una BD

not_anio	not_semana	tipo_dx	not_diresa_nom	not_eess_cat	edad	sexo	resid_dpto
2021	36	C	DIRIS LIMA CENTRO	III-2	4	M	APURIMAC
2021	21	C	CAJAMARCA	II-2	5	F	CAJAMARCA
2021	29	C	JUNIN	III-E	12	M	JUNIN
2021	23	C	PIURA	III-1	16	F	PIURA
2021	22	C	DIRIS LIMA NORTE	III-1	28	F	ANCASH
2021	46	C	LA LIBERTAD	III-1	28	M	LA LIBERTAD
2021	33	C	PIURA	II-2	53	M	PIURA
2021	38	S	JUNIN	III-E	53	F	JUNIN
2021	40	S	DIRIS LIMA CENTRO	III-1	55	F	HUANUCO
2021	45	C	DIRIS LIMA CENTRO	III-1	56	M	LIMA
2021	10	C	DIRIS LIMA CENTRO	III-2	57	M	LIMA
2021	20	C	LA LIBERTAD	III-1	60	F	LA LIBERTAD
2021	7	C	DIRIS LIMA CENTRO	III-2	61	M	LIMA
2021	29	S	DIRIS LIMA CENTRO	III-2	62	M	LIMA
2021	32	C	LUCIANO CASTILLO	II-2	64	F	PIURA
2021	30	C	JUNIN	III-E	70	M	JUNIN
2021	16	C	DIRIS LIMA CENTRO	III-2	71	F	LIMA
2022	1	C	PIURA	II-2	7	M	PIURA
2021	26	C	DIRIS LIMA CENTRO	III-1	18	F	LIMA
2021	48	C	DIRIS LIMA CENTRO	III-2	26	M	ANCASH
2021	29	C	PIURA	II-2	35	M	PIURA

Variables categóricas / cualitativas

Tipo de dato	• Descriptivos, cualidades • Datos no numéricos • **Ejemplos:** ¿Hospitalizado? (si/no), sexo, departamento
Resumir con	• Medidas de frecuencia • **Medidas:** Recuento, Razón, Proporción, Tasas

Variables numéricas / cuantitativas

Tipo de dato	• Datos numéricos • Continuos / Discretos • **Ejemplos:** imc, nivel de glucosa, edad
Resumir con	• Medidas de tendencia central • Medidas de dispersión

Análisis descriptivo por tipo de variable

- ***Para variables categóricas:***
  - Frecuencias absolutas
  - Frecuencias relativas (proporciones/porcentajes)
- ***Para variables numéricas:***
  - **Medidas de tendencia central:** media, mediana (percentil 50)
  - **Medidas de dispersión:** desviación estándar, rango, mínimo, máximo, rango intercuartil (percentil 25 percentil 75)

Recordando las medidas de resumen para variables categóricas

Medidas de resumen para variables categóricas en epidemiología

Conteo o Recuento
Número de casos

Razón
Comparación de dos números cualesquiera

Proporción / Porcentaje
Parte del todo

Tasa de Incidencia
Casos nuevos, cualquier intervalo de tiempo, deben tener en cuenta el tiempo

Tasa de Prevalencia
Casos actuales en la población independientemente del tiempo de inicio

Tasa de Ataque
Casos nuevos, intervalo de tiempo corto

Tasa de Mortalidad
Cantidad de muertes en durante un período de tiempo determinado

Tasa de Letalidad
Proporción de casos que murieron

Tablas

Tablas para variables numéricas

Medidas de tendencia central:

- Media (promedio)
- Mediana (percentil 50)

Medidas de dispersión:

- Desviación estándar
- Rango
- Rango intercuartil (percentil 25 percentil 75)

`summarize()` / `summarise()`

Resume el marco de datos en un solo valor o vector.

```
> sgb %>%
+   summarise(
+     mediana = median(edad),
+     minimo = min(edad),
+     maximo = max(edad)
+     )
  mediana minimo maximo
1    44.5      2     76
```

summarize() / summarise()

Funciones	Paquete	Descripción
`min(), max()`	base	Valores max y min
`mean()`		media
`median()`		mediana
`sum()`		suma de los valores
`var, sd()`		varianza y desviación típica
`first()`	dplyr (tidyverse)	primer valor en un vector
`last()`		el último valor en un vector
`n()`		el número de valores en un vector
`n_distinct()`		el número de valores distintos en un vector

Tablas para variables categóricas

Conteo

- Medida descriptiva común. Primer paso en el cálculo de tasas

Proporción y porcentaje

- Comparación de una parte con el todo.
- Útil para describir la distribución de características dentro de una población.

Proporción = x / y

Porcentaje = x / y * 100

TABLE. Characteristics of reported confirmed and probable travel-associated dengue cases (N = 7,528) — National Arbovirus Surveillance System, United States, 2010–2018, 2019, and 2020–2021.

Characteristic	No. (%)		
	2010–2018	2019	2020–2021
Total	**5,495 (100)**	**1,474 (100)**	**559 (100)**
Case status			
Probable	1,708 (31)	793 (46)	229 (41)
Confirmed	3,787 (69)	681 (54)	330 (59)
Sex			
Female	2,748 (50)	743 (50)	275 (49)
Male	2,746 (50)	729 (49)	284 (51)
Age group, yrs			
0–9	143 (3)	68 (5)	21 (4)
10–19	557 (10)	178 (12)	60 (11)
20–29	962 (17)	196 (13)	80 (14)
30–39	936 (17)	238 (16)	88 (16)
40–49	991 (18)	254 (17)	107 (19)
50–59	986 (18)	256 (18)	94 (17)
≥60	910 (17)	284 (19)	108 (19)
Dengue clinical syndrome			
Dengue-like illness	297 (5)	56 (4)	18 (3)
Dengue	5,030 (92)	1,388 (94)	532 (95)
Severe dengue	55 (1)	29 (2)	4 (1)
Unknown	113 (2)	1 (<1)	5 (1)
Hospitalized			
No	2,912 (53)	820 (56)	251 (45)
Yes	2,325 (42)	625 (42)	185 (33)
Unknown	258 (5)	29 (2)	123 (22)

Wong JM et al. Travel-Associated Dengue Cases - United States, 2010-2021. MMWR Morb Mortal Wkly Rep. 2023 Jul 28;72(30):821-826.

Cálculo de frecuencias absolutas y relativas

1. Inspección del tipo de variables → `str()` / `glimpse()`
2. Limpieza de datos → `clean_names()` / `distinct()` / `rename()`
3. Manipulación de datos → `select()` / `filter()` / `mutate()` / `recode()`
4. Funciones para proporciones y porcentajes:

   → `count()`, `mutate()`, `sum()`, `round()`

   → `group_by()`, `summarise()`, `mutate()`

Cálculo de frecuencias absolutas y relativas

1

```
$ pulso      <dbl> 88, 60, 62, 58, 80, 73, 75, 76, 65, 85, 71, 75, 86, 69, 7…
$ glucosa    <dbl> 150, 69, 90, 64, 60, 87, 98, 78, 78, 73, 66, 75, 78, 57, …
$ colesterol <dbl> 180, 194, 295, 200, 212, 260, 215, 298, 217, 260, 234, 18…
$ hdl        <dbl> NA, 37, 55, 50, 54, 78, 44, 35, NA, NA, NA, 56, NA, NA, N…
$ ldl        <dbl> NA, 181, 197, 150, 212, 234, 155, 209, NA, NA, NA, 153, N…
$ muerte     <dbl> 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, …
$ t_angina   <dbl> 18.439425, 7.737166, 18.565366, 21.284052, 1.180382, 2.25…
$ t_infarto  <dbl> 18.4394251, 19.5345654, 18.5653662, 21.2813142, 17.980666…
$ t_acv      <dbl> 14.989733, 19.529090, 18.565366, 21.284052, 2.464433, 1.6…
```

2-3

```r
# Manipulación de datos
framingham_1 <- framingham %>%
  distinct() %>%
  select(id, sexo, fumador, diabetes, hta, acv, muerte) %>%
  mutate(sexo = if_else(sexo == 1, "masculino", "femenino"),
         fumador=recode(fumador, "0"="no fumador", "1"="fumador", .default = NA_character_),
         diabetes=recode(diabetes, "0"="no diabetico", "1"="diabetico", .default = NA_character_),
         hta=recode(hta, "no"="no hipertenso", "si"="hipertenso", .default = NA_character_),
         acv=recode(acv, "0"="no acv", "1"="acv", .default = NA_character_),
         muerte=recode(muerte, "0"="vivo", "1"="muerto", .default = NA_character_))
```

id	sexo	fumador	diabetes	hta	acv	muerte
610021	femenino	fumador	diabetico	hipertenso	no acv	muerto
7547192	femenino	no fumador	no diabetico	hipertenso	no acv	muerto
6961522	femenino	no fumador	no diabetico	hipertenso	no acv	muerto
3719680	femenino	fumador	no diabetico	hipertenso	no acv	muerto
1579538	femenino	fumador	no diabetico	hipertenso	no acv	vivo
9216480	femenino	no fumador	no diabetico	hipertenso	no acv	vivo
491826	femenino	fumador	no diabetico	hipertenso	no acv	vivo
5023667	masculino	no fumador	no diabetico	hipertenso	no acv	muerto
8465853	femenino	fumador	no diabetico	hipertenso	no acv	muerto
749515	femenino	fumador	no diabetico	hipertenso	no acv	vivo
1520409	masculino	fumador	no diabetico	hipertenso	no acv	vivo
5633361	femenino	fumador	no diabetico	hipertenso	no acv	vivo
6478737	masculino	fumador	no diabetico	hipertenso	no acv	muerto
1267186	masculino	no fumador	no diabetico	hipertenso	no acv	muerto

```
> # Frecuencias absolutas:
> framingham_1 %>%
+   count(sexo)
      sexo    n
1  femenino 2490
2 masculino 1944
> framingham_1 %>%
+   count(sexo, acv, muerte)
      sexo    acv muerte    n
1  femenino    acv muerto   13
2  femenino    acv   vivo    5
3  femenino no acv muerto  694
4  femenino no acv   vivo 1778
5 masculino    acv muerto   12
6 masculino    acv   vivo    2
7 masculino no acv muerto  831
8 masculino no acv   vivo 1099
```

4

Cálculo de frecuencias relativas

Proporciones

```
> # Cálculo de proporción por sexo
> framingham_1 %>%
+   count(sexo, name = "casos") %>%
+   mutate(proporcion = casos/sum(casos))
      sexo casos proporcion
1  femenino  2490  0.5615697
2 masculino  1944  0.4384303
>
```

Porcentajes

```
> # Cálculo de porcentaje por sexo
> framingham_1 %>%
+   count(sexo, name = "casos") %>%
+   mutate(porcentaje = casos/sum(casos) * 100)
      sexo casos porcentaje
1  femenino  2490   56.15697
2 masculino  1944   43.84303
```

Redondeo de frecuencias relativas

```
tabla1 <- framingham_1 %>%
       count(sexo, name = "casos") %>%
       mutate(porcentaje = round(casos/sum(casos)*100, 2))


tabla2 <- framingham_1 %>%
       count(fumador, sexo, name = "casos") %>%
       mutate(porcentaje = round(casos/sum(casos)*100, 2))
```

round()

sexo	casos	porcentaje
femenino	2490	56.16
masculino	1944	43.84

fumador	sexo	casos	porcentaje
fumador	femenino	1006	22.69
fumador	masculino	1175	26.50
no fumador	femenino	1484	33.47
no fumador	masculino	769	17.34

Una variable: group_by(), summarise(), mutate()

```
tabla1 <- framingham_1 %>%
       count(sexo, name = "casos") %>%
       mutate(porcentaje = round(casos/sum(casos)*100, 2))


tabla3 <- framingham_1 %>%
       group_by(sexo) %>%
       summarise(casos=n()) %>%
       mutate(porcentaje = round(casos/sum(casos) * 100, 2))
```

sexo	casos	porcentaje
femenino	2490	56.16
masculino	1944	43.84

sexo	casos	porcentaje
femenino	2490	56.16
masculino	1944	43.84

Dos variables: group_by(), summarise(), mutate()

OPCIÓN 1:

```
tabla4 <- framingham_1 %>%
       group_by(muerte, sexo) %>%
       summarise(casos=n()) %>%
       mutate(proporcion = round(casos/sum(casos),2),
              porcentaje = round(casos/sum(casos) * 100, 2)) %>%
       ungroup()
```

	muerte	sexo	casos	proporcion	porcentaje
1	muerto	femenino	707	0.46	45.61
2	muerto	masculino	843	0.54	54.39
3	vivo	femenino	1783	0.62	61.82
4	vivo	masculino	1101	0.38	38.18

Tasas relacionadas con epidemiología

$$\frac{\text{Número de muertes debido a una enfermedad}}{\text{Número de casos de la enfermedad}} \times \text{Constante (usualmente 100)}$$

Letalidad

$$\frac{\text{Número de casos nuevos durante un periodo específico}}{\text{Tamaño de la población} \times \text{Tiempo}} \times \text{Constante}$$

Tasa de Incidencia

$$\frac{\text{Número de muertes durante un periodo específico}}{\text{Tamaño de la población}} \times \text{Constante (usualmente 1000)}$$

Tasa de Mortalidad

$$\frac{\text{Número actual de casos o personas con el atributo}}{\text{Tamaño de la población}} \times \text{Constante}$$

Tasa de Prevalencia

Paquete {gtsummary} : Función `tbl_summary()`

- **{gtsummary}** permite crear tablas descriptivas o de resumen con formato de publicación sencillo.
- La función principal de este paquete es **`tbl_summary()`**: Muestra estadísticas de resumen según el tipo de variable (categórica o continua).
- Para guardar las tablas en un formato editable se complementa con el paquete **{gt}** .

Paquete {gt} : Función `gtsave()`

Paquete {gtsummary} : Función `tbl_summary()`

- **Variables categóricas:** calcula el *número de observaciones* por categoría y el *porcentaje de las observaciones* sobre el total.
- **Variables continuas:** calcula la mediana y el rango intercuartílico; es decir, el intervalo comprendido entre el percentil 25 y el percentil 75.

```
tab_fram_01 <- fram_1 %>%
   tbl_summary() %>%
   as_gt()

gtsave(tab_fram_01, "tabla01_fram.docx")
```

Characteristic	N = 4,434[1]
sexo	
femenino	2,490 (56%)
masculino	1,944 (44%)
edad	49 (42, 57)
imc	25.5 (23.1, 28.1)
diabetes	
diabetico	121 (2.7%)
no diabetico	4,313 (97%)
hta	
hipertenso	1,619 (37%)
no hipertenso	2,815 (63%)
acv	
acv	32 (0.7%)
no acv	4,402 (99%)
glucosa	78 (72, 87)
colesterol	234 (206, 264)
hdl	48 (39, 58)
ldl	173 (145, 205)

[1]n (%); Median (Q1, Q3)

Paquete {gtsummary} : Función `tbl_summary()`

- **Tablas descriptivas**
- **Tablas cruzadas**

```
tab05 <- fram_1 %>%
    tbl_summary(include =
          c(sexo,
            grupo_edad,
            glucosa,
            colesterol),
            by=muerte) %>%
    as_gt()

gtsave(tab05, "tabla05_fram.docx")
```

Características	muerto N = 1,550[1]	vivo N = 2,884[1]
sexo		
femenino	707 (46%)	1,783 (62%)
masculino	843 (54%)	1,101 (38%)
Grupo de edad		
Adulto	998 (64%)	2,652 (92%)
Adulto mayor	552 (36%)	232 (8.0%)
Glucosa	79 (72, 90)	77 (71, 85)
Colesterol	240 (212, 270)	230 (202, 260)

[1]n (%); Median (Q1, Q3)

Gracias