Transformación de datos con dplyr : : **GUÍA RÁPIDA**

Las funciones de **dplyr** se usan con canalizaciones (pipe) y **datos ordenados**. Datos ordenados:

Cada **variable** es su propia **columna**

Cada **observación**, o **caso**, es una **fila**

pipe

x |> f(y)
es **f(x, y)**

Resumir Casos

Aplique **funciones de resumen** a las columnas para crear una nueva tabla de estadísticas de resumen. Las funciones de resumen toman vectores como entrada y devuelven un valor (ver atrás).

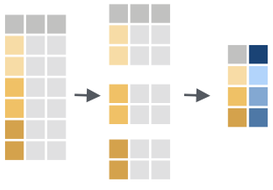
summary function

summarise(.data, …**)**
Calcular tabla de resúmenes.
mtcars |> summarise(avg = mean(mpg))

count(.data, …, wt = NULL, sort = FALSE, name = NULL**)** Cuente el número de filas de cada grupo definido por las variables de ... Además **tally()**.
Mtcars |> count(cyl)

Agrupar Casos

Utilice **group_by(**.data, ..., .add = FALSE, .drop = TRUE**)** para crear una copia "agrupada" de una tabla agrupada por columnas en ... Las funciones de dplyr manipularán cada "grupo" por separado y combinarán los resultados.



```
mtcars |>
  group_by(cyl) |>
  summarise(avg = mean(mpg))
```

Utilice **rowwise(**.data, ...**)** para agrupar los datos en filas individuales. Las funciones dplyr calcularán los resultados de cada fila. También aplique funciones a las columnas de lista. Consulte la hoja de referencia rápida de tidyr para el flujo de trabajo de la lista de columnas.

```
starwars |>
  rowwise() |>
  mutate(film_count = length(films))
```

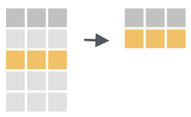
ungroup(x, …**)** Devuelve una copia desagrupada de la tabla.

```
g_mtcars <- mtcars |> group_by(cyl)
ungroup(g_mtcars)
```

Manipular Casos

EXTRAER CASOS

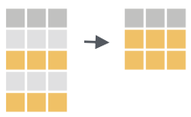
Las funciones de fila devuelven un subconjunto de filas como una nueva tabla.



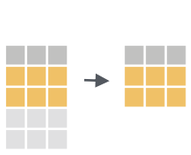
filter(.data, …, .preserve = FALSE**)** Extraer filas que cumplan criterios lógicos.
mtcars |> filter(mpg > 20)

distinct(.data, …, .keep_all = FALSE**)** Eliminar filas con valores duplicados.
mtcars |> distinct(gear)

slice(.data, …, .preserve = FALSE**)** Seleccionar filas por posición.
mtcars |> slice(10:15)



slice_sample(.data, …, n, prop, weight_by = NULL, replace = FALSE**)** Seleccione filas al azar. Use n para seleccionar un número de filas y prop para seleccionar una fracción de filas.
mtcars |> slice_sample(n = 5, replace = TRUE)



slice_min(.data, order_by, …, n, prop, with_ties = TRUE**)** y **slice_max()** Seleccione las filas con los valores más bajos y más altos.
mtcars |> slice_min(mpg, prop = 0.25)

slice_head(.data, …, n, prop**)** y **slice_tail()** Seleccione la primera o la última fila.
mtcars |> slice_head(n = 5)

Operadores lógicos y booleanos que se pueden usar con filter()

==	<	<=	is.na()	%in%	|	xor()
!=	>	>=	!is.na()	!	&	

Vea **?base::Logic** y **?Comparison** para obtener ayuda.

ORGANIZAR CASOS

arrange(.data, …, .by_group = FALSE**)** Ordene las filas por valores de una columna o columnas (de menor a mayor), utilícela con **desc()** para ordenar de mayor a menor.
mtcars |> arrange(mpg)
mtcars |> arrange(desc(mpg))

AÑADIR CASOS

add_row(.data, …, .before = NULL, .after = NULL**)** Agregar una o más filas a una tabla.
cars |> add_row(speed = 1, dist = 1)

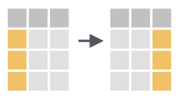
Manipular variables

EXTRAER VARIABLES

Las funciones de columna devuelven un conjunto de columnas como un nuevo vector o tabla.

pull(.data, var = -1, name = NULL, …**)** Extraer valores de columna como un vector, por nombre o índice.
mtcars |> pull(wt)

select(.data, …**)** Extraer columnas como una tabla.
mtcars |> select(mpg, wt)



relocate(.data, …, .before = NULL, .after = NULL**)** Mover columnas a una nueva posición.
mtcars |> relocate(mpg, cyl, .after = last_col())

Utilice estos ayudantes con select() y across()
e.j. select(mtcars, mpg:cyl)

contains(match**)**	**num_range(**prefix, range**)**	**:**, e.g. mpg:cyl
ends_with(match**)**	**all_of(**x**)/any_of(**x, …, vars**)**	-, e.g, -gear
starts_with(match**)**	**matches(**match**)**	**everything()**

MANIPULAR VARIAS VARIABLES A LA VEZ

```
df <- tibble(x_1 = c(1, 2), x_2 = c(3, 4), y = c(4, 5))
```

across(.cols, .funs, …, .names = NULL**)** Resumir o mutar varias columnas de la misma manera.
df |> summarise(across(everything(), mean))

c_across(.cols**)** Calcule a través de columnas en datos agrupados por filas**.**

```
df |>
  rowwise() |>
  mutate(x_total = sum(c_across(1:2)))
```

MAKE NEW VARIABLES

Apply **vectorized functions** to columns. Vectorized functions take vectors as input and return vectors of the same length as output (see back).

vectorized function

mutate(.data, …, .keep = "all", .before = NULL, .after = NULL**)** compute new column(s). Also **add_column().**
mtcars |> mutate(gpm = 1 / mpg)
mtcars |> mutate(gpm = 1 / mpg, keep = “none”)

rename(.data, …**)** Rename columns. Use **rename_with()** to rename with a function.
mtcars |> rename(miles_per_gallon = mpg)

Funciones Vectorizadas

USAR CON MUTATE ()

mutate() aplica funciones vectorizadas a las columnas para crear nuevas columnas. Las funciones vectorizadas toman vectores como vectores de entrada y de retorno de la misma longitud que salida.

función vectorizada

DESPLAZAR

dplyr::**lag()** – desplaza elementos por 1
dplyr::**lead()** - desplaza elementos por -1

AGREGADO ACUMULADO

dplyr::**cumall()** - all() acumulativo
dplyr::**cumany()** - any() acumulativo
cummax() - max() acumulativo
dplyr::**cummean()** - mean() acumulativo
cummin() - min() acumulativo
cumprod() - prod() acumulativo
cumsum() - sum() acumulativo

CLASIFICACIÓN

dplyr::**cume_dist()** - proporción de todos los valores <=
dplyr::**dense_rank()** - Rangos w empates = mín., sin huecos
dplyr::**min_rank()** - Rango con empates = Min
dplyr::**ntile()** - Contenedores en N contenedores
dplyr::**percent_rank()** - min_rank escalado a [0,1]
dplyr::**row_number()** - Rango con empates= "first"

MATEMÁTICA

+, - , *, /, ^, %/%, %% - operaciones aritméticas
log(), log2(), log10() - logs
<, <=, >, >=, !=, == - comparaciones lógicas
dplyr::**between()** - x >= left & x <= right
dplyr::**near()** - seguro == para números de coma flotante

MISCELÁNEO

dplyr::**case_when()** - multi-caso if_else()

```
starwars |>
  mutate(type = case_when(
    height > 200 | mass > 200 ~ "large",
    species == "Droid"        ~ "robot",
    TRUE                      ~ "other")
    )
```

dplyr::**coalesce()** - primeros valores no NA por elemento en un conjunto de vectores
dplyr::**if_else()** - if() + else() por elemento
dplyr::**na_if()** - reemplace valores específicos con NA
pmax() - max() por elemento
pmin() - min() por elemento

Funciones De Resumen

PARA USAR CON SUMMARISE ()

summarise() aplica funciones de resumen a las columnas para crear una nueva tabla. Las funciones de resumen toman vectores como entrada y devuelven valores individuales como salida.

función de resumen

CONTAR

dplyr::**n()** - número de valores/filas
dplyr::**n_distinct()** - # de carácter único
sum(!is.na()) - # de los que no son NA

POSICIÓN

mean() - media, además **mean(!is.na())**
median() - mediana

LÓGICO

mean() - proporción de TRUE's
sum() - # of TRUE's

ORDEN

dplyr::**first()** - primer valor
dplyr::**last()** - último valor
dplyr::**nth()** - valor en la enésima ubicación del vector

RANGO

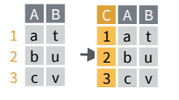
quantile() - enésimo cuantil
min() - valor mínimo
max() - valor máximo

PROPAGACIÓN

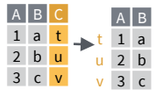
IQR() - Rango intercuartílico
mad() - desviación absoluta mediana
sd() - desviación estándar
var() - varianza

Nombres De Fila

Los datos ordenados no utilizan nombres de fila, que almacenan una variable fuera de las columnas. Para trabajar con los nombres de fila, primero muévalos a una columna.



tibble::**rownames_to_column()**
Mover nombres de fila a col.
a <- rownames_to_column(mtcars, var = "C")

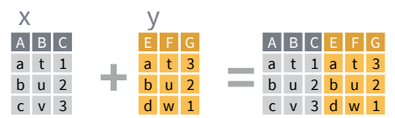


tibble::**column_to_rownames()**
Mover col a los nombres de fila.
column_to_rownames(a, var = "C")

También tibble::**has_rownames()** y tibble::**remove_rownames()**.

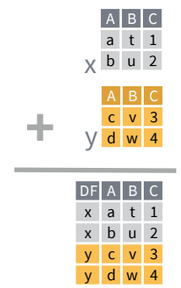
Combinar Tablas

COMBINAR VARIABLES



bind_cols(…, .name_repair**)** Devuelve tablas colocadas una al lado de la otra como una sola tabla. Las longitudes de las columnas deben ser iguales. Las columnas NO coincidirán con id (para ello, consulte Datos relacionales a continuación), así que asegúrese de comprobar que ambas tablas están ordenadas de la forma que desee antes de enlazar.

COMBINAR CASOS



bind_rows(…, .id = NULL**)** Devuelve tablas una encima de la otra como una sola tabla. Establezca .id en un nombre de columna para agregar una columna de los nombres de tabla originales (como se muestra en la imagen).

DATOS RELACIONALES

Utilice una **"Unión mutante"** para unir una tabla a columnas de otra, haciendo coincidir los valores con las filas a las que corresponden. Cada combinación conserva una combinación diferente de valores de las tablas.

left_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), …, keep = FALSE, na_matched = "na"**)** Unir valores coincidentes de y a x.

right_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), …, keep = FALSE, na_matches = "na"**)** Unir valores coincidentes de x a y.

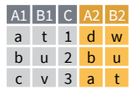
inner_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), …, keep = FALSE, na_matches = "na"**)** Unir datos. Conservar solo las filas con coincidencias.

full_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), …, keep = FALSE, na_matches = "na"**)** Unir datos. Conservar todos los valores, todas las filas.

COINCIDENCIA DE COLUMNAS PARA UNIONES

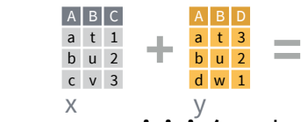
Utilice **by = c("col1", "col2", ...)** para especificar una o varias columnas comunes para que coincidan en.
left_join(x, y, by = "A")

Usar un vector con nombre, **by = c("col1" = "col2")**, para que coincida en columnas que tienen nombres diferentes en cada tabla.
left_join(x, y, by = c("C" = "D"))



Usar **suffix** para especificar el sufijo que se va a dar a las columnas no coincidentes que tienen el mismo nombre en ambas tablas.
left_join(x, y, by = c("C" = "D"), suffix = c("1", "2"))

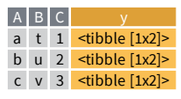
Utilice una **"Unión de filtrado"** para filtrar una tabla con respecto a las filas de otra.



semi_join(x, y, by = NULL, copy = FALSE, …, na_matches = "na"**)** Devuelve filas de x que tienen una coincidencia en y. Utilícelo para ver lo que se incluirá en una unión.

anti_join(x, y, by = NULL, copy = FALSE, …, na_matches = "na"**)** Devuelve filas de x que no tienen una coincidencia en y. Utilícelo para ver lo que no se incluirá en una unión.

Usar una **"unión anidada"** para unir internamente una tabla con otra en un marco de datos anidado.



nest_join(x, y, by = NULL, copy = FALSE, keep = FALSE, name = NULL, …**)** Unir datos, anidar coincidencias de y en una sola columna de trama de datos nueva.

OPERACIONES DE CONJUNTO

intersect(x, y, ...)
Filas que aparecen tanto en x como en y.

setdiff(x, y, ...)
Filas que aparecen en x pero no en y.

union(x, y, ...)
Filas que aparecen en x o y. (Duplicados eliminados). **union_all()** conserva los duplicados.

Utilice **setequal()** para comprobar si dos conjuntos de datos contienen exactamente las mismas filas (en cualquier orden).