Transformación de datos dplyr : : **HOJA DE REFERENCIA**

dplyr r funciona con canalizaciones y requiere **datos ordenados**. En datos ordenados:

Cada **variable** en su propia columna

Cada **observasión**, o **caso**, en su propia **fila**

pipes

x %>% f(y) se convierte en **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).

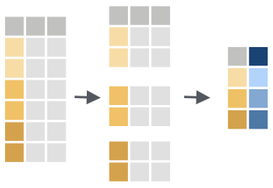
función de resumen

summarise(.data, …**)**
Calcula la tabla de resúmenes.
summarise(mtcars, avg = mean(mpg))

count(.data, …, wt = NULL, sort = FALSE, name = NULL**)** Cuenta el número de filas de cada grupo definido por las variables en … También **tally()**.
count(mtcars, 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 de dplyr calcularán los resultados de cada fila. También aplica funciones a las columnas de listas. Consulte la hoja de referencia rápida de tidyr para el flujo de trabajo de columnas de listas.

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

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

Manipular Casos

EXTRAER CASOS

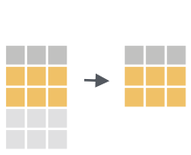
Las funciones de fila devuelven un subconjunto de filas como una nueva tabla.

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

distinct(.data, …, .keep_all = FALSE**)** Elimina las filas con valores duplicados.
distinct(mtcars, gear)

slice(.data, …, .preserve = FALSE**)** Selecciona las filas por posición.
slice(mtcars, 10:15)

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



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

slice_head(.data, …, n, prop**)** y **slice_tail()** Seleccionan las primeras o las últimas filas.
slice_head(mtcars, n = 5)

Operadores lógicos y booleanos para usar con filter()

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

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

ORDENAR CASOS

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

AÑADIR CASOS

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

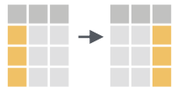
Manipular Variables

EXTRAER VARIABLES

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

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

select(.data, …**)** Extrae columnas como una tabla.
select(mtcars, mpg, wt)



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

Utilice estos ayudantes con select() y across()
e.g. 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

across(.cols, .funs, …, .names = NULL**)** Resume o muta varias columnas de la misma manera.
summarise(mtcars, across(everything(), mean))

c_across(.cols**)** Calcula a través de columnas en datos por filas.
transmute(rowwise(UKgas), total = sum(c_across(1:2)))

CREAR NUEVAS VARIABLES

Aplicación de **funciones vectorizadas** a columnas. Las funciones vectorizadas toman vectores como entrada y devuelven vectores de la misma longitud como salida (Ver atrás).

función vectorizada

mutate(.data, …, .keep = "all", .before = NULL, .after = NULL**)** Calcula nueva(s) columna(s). También **add_column(), add_count(),** y **add_tally().**
mutate(mtcars, gpm = 1 / mpg)

transmute(.data, …**)** Calcula nueva(s) columna(s), elimina otras.
transmute(mtcars, gpm = 1 / mpg)

rename(.data, …**)** Renombra columnas. Use **rename_with()** para renombrar con una función.
rename(cars, distance = dist)

Funciones Vectorizadas

PARA USAR CON MUTATE ()

mutate() y **transmute()** Aplican funciones vectorizadas a columnas para crear nuevas columnas. Las funciones vectorizadas toman vectores como entrada y devuelven vectores de la misma longitud como salida.

función vectorizada

DESFASAR
dplyr::**lag()** - desfasar elementos por 1
dplyr::**lead()** - desfasar 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

RANKING
dplyr::**cume_dist()** - proporción de todos los valores <=
dplyr::**dense_rank()** - rango con empates = mín., sin espacios
dplyr::**min_rank()** - rango con empates = mín.
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() - logaritmos
<, <=, >, >=, !=, == - comparaciones lógicas
dplyr::**between()** - x >= izquierda & x <= derecha
dplyr::**near()** - == seguro para números de punto flotante

MISCELÁNEAS
dplyr::**case_when()** - if_else() multicaso

```
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 único
sum(!is.na()) - # de los que no son NA

POSICIÓN
mean() - promedio, también **mean(!is.na())**
median() - media

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

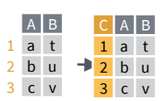
ORDEN
dplyr::**first()** - primer valor
dplyr::**last()** - último valor
dplyr::**nth()** - valor en la enésima ubicación del vector

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

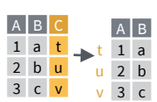
PROPAGACIÓN
IQR() - rango intercuartílico
mad() - desviación absoluta mediana
sd() - desviación estándar
var() - varianza

Nombres de Filas

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()** Mueve los nombres de fila a col.
a <- rownames_to_column(mtcars, var = "C")



tibble::**column_to_rownames()** Mueva col a los nombres de filas.
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 serán combinadas por 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 combinarlas.

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"**)** Combina los valores coincidentes de y a x.

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

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

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

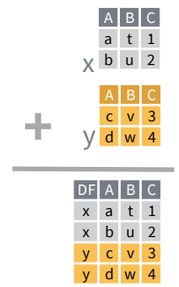
EMPAREJAMIENTO DE COLUMNAS PARA JOINS

Use **by = c("col1", "col2", …)** para especificar una o varias columnas comunes con las que emparejar.
left_join(x, y, by = "A")

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

Use **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"))

COMBINAR CASOS



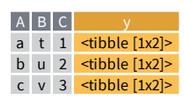
bind_rows(…, .id = NULL**)** Devuelve tablas una encima de la otra como una sola tabla. Establezca .id en un nombre de columna para agregar una columna de los nombres de tabla originales (como se muestra en la imagen).

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 combinación.

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 combinación.

Utilice 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, anidando coincidencias de y en una sola columna de marco 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.

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