# Transformación de datos con dplyr:: guía rápida



**dplyr** funciona con pipes y asume que los datos están ordenados. En los datos ordenados:



Cada **variable** está en su propia columna



Cada **observacíon** o caso está en su propia



x % > % f(y)se convierte en f(x, y)

## **Resumir Casos**

Las **funciones de resumen** se aplican a columnas para crear una tabla nueva con los estadísticos de resumen. Las funciones (funs) de resumen toman vectores como entrada y devuelven un solo valor (ver en el reverso).

## función de resumen



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



count(x, ..., wt = NULL, sort = FALSE)Cuenta los números de filas de cada grupo definido por las variables en ... También puede usarse tally(). count(iris, Species)

## **VARIANTES**

**summarise\_all()** - Aplica funs a cada columna. **summarise\_at()** - Aplica funs a columnas específicas. **summarise\_if()** - Aplica funs a todas las columnas de un tipo.

## **Agrupar Casos**

Usa **group\_by()** para crear una copia "agrupada" de una tabla. Las funciones de dplyr manipulan cada "grupo" por separado y luego combinan los resultados.



mtcars %>% group\_by(cyl) %>% summarise(avg = mean(mpg))

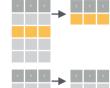
group\_by(.data, ..., add = FALSE) devuelve una copia de la tabla agrupada por .... g\_iris <- group\_by(iris, Species)</pre>

ungroup(x,...)devuelve una copia desagrupada de la tabla ungroup(g\_iris)

## **Manipular Casos**

#### **EXTRAER CASOS**

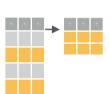
Las funciones de fila devuelven un subconjunto de filas como una tabla nueva.



**filter**(.data, ...) extrae filas que cumplen con un criterio lógico. filter(iris, Sepal.Length > 7)



distinct(.data, ..., .keep\_all = FALSE) remueve filas con valores duplicados. distinct(iris, Species)



sample\_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = parent.frame()) Selecciona al azar una fracción de filas. sample\_frac(iris, 0.5, replace = TRUE)

sample\_n(tbl, size, replace = FALSE, weight = NULL, .env = parent.frame()) selecciona al azar un numero fijo de filas. sample\_n(iris, 10, replace = TRUE)

**slice(**.data, ...**)** Selecciona filas por posición. *slice(iris*, *10*:*15*)



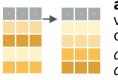
top\_n(x, n, wt) Selecciona y ordena las n entradas más altas (por grupo si los datos están agrupados). top\_n(iris, 5, Sepal.Width)

## Operadores lógicos y booleanos para usar con filter()

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

Para más ayuda busca ?base::Logic y ?Comparison

#### **ORDENAR CASOS**



**arrange**(.data, ...) ordena las filas por los valores crecientes de una o más columnas. Para ordenar de mayor a menor, agrega **desc()**. 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(faithful, eruptions = 1, waiting = 1)

## Manipular Variables

#### **EXTRAER VARIABLES**

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



pull(.data, var = -1) Extrae valores de columnas como vectores. Elige las columnas por nombre o pull(iris, Sepal.Length)



select(.data,...) Extrae columnas como tabla. También puedes usar select if(). select(iris, Sepal.Length, Species)

## Usa estos ayudantes con select ():

e.j. select(iris, starts with("Sepal"))

contains(match) ends\_with(match) matches(match)

num range(prefix, range) one\_of(...)

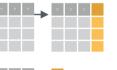
: e.j. mpg:cyl • e.j. -Species

#### **CREAR NUEVAS VARIABLES**

Se hace aplicando **funciones vectorizadas** a columnas. Las funciones vectorizadas (funs) toman vectores como entradas y devuelven vectores del mismo tamaño como salida (ver en el reverso).

starts\_with(match)

## función vectorizada



mutate(.data, ...) Calcula columna(s) nueva(s). mutate(mtcars, gpm = 1/mpg)



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



**mutate\_all(**.tbl, .funs, ...) Aplica funs a cada columna. Usa con **funs()**, o también con **mutate\_if()**. mutate\_all(faithful, funs(log(.), log2(.))) mutate\_if(iris, is.numeric, funs(log(.)))



**mutate\_at(**.tbl, .cols, .funs, ...**)** Aplica funs a columnas especificas. Usa con funs(), vars() y las funciones ayudantes de select(). mutate\_at(iris, vars(-Species), funs(log(.)))



add\_column(.data, ..., .before = NULL, .after = NULL) Agrega nueva(s) columna(s). También con add\_count(), add\_tally(). add\_column(mtcars, new = 1:32)



rename(.data, ...) Renombra columnas. rename(iris, Length = Sepal.Length)



## Funciones Vectorizadas Funciones de Resumen

### PARA USAR CON MUTATE ()

mutate() y transmute() aplican funciones vectorizadas a columnas para crear columnas nuevas. Las funciones vectorizadas toman vectores como entradas y como salida devuelven vectores de la misma longitúd.

## función vectorizada



### **CORRIMIENTOS (OFFSETS)**

Busca los valores siguientes o anteriores a un elemento dentro de un vector dplyr::lead() - avanzando dplyr::lag() - retrocediendo

## **AGREGADOS ACUMULADOS**

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

#### **RANKINGS**

dplyr::cume\_dist() - Proporcion de todos los valores <= dplyr::dense\_rank() - rank con ties = min, sin huecos dplyr::min\_rank() - rank con ties = min dplyr::ntile() - bins en n bins dplyr::percent\_rank() - min\_rank escalado a [0,1]
dplyr::row\_number() - rank donde ties = "first"

## **OPERACIONES MATEMÁTICAS**

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

## **MISCELÁNEA**

## dplyr::case\_when() - if\_else() multi caso iris %>% mutate(Species = case when(

Species == "versicolor" ~ "versi". Species == "virginica" ~ "virgi", TRUE ~ Species))

dplyr::coalesce() -primeros valores no NA por elemento a lo largo de un conjunto de vectores dplyr::**if\_else()** - if() + else() por elemento dplyr::na\_if() - reemplaza valores con NA pmax() - max() por elemento **pmin()** - element-wise min() dplyr::**recode()** - Vectorized switch() dplyr::recode\_factor() - Vectorized switch() for factors

### PARA USAR CON SUMMARISE ()

**summarise()** aplica funciones de resumen a columnas para crear una tabla nueva. Las funciones de resumen toman vectores de n valores como entrada v devuelven un valor único como salida.

## función de resumen

## **CONTEOS**

dplyr::n() - número de valores/filas dplyr::**n\_distinct()** - # de valores únicos sum(!is.na()) - # de valores no NA

## LOCACIÓN

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

### LÓGICAS

mean() – proporción de verdaderos (TRUE) sum() - # de TRUE's

## POSICIÓN/ORDEN

dplyr::first() - primer valor
dplyr::last() - último valor

dplyr::**nth()** – valor en la n-ésima posición de un dplyr es el 0.8.3 y el tibble es el 2.1.3 vector

#### **RANK**

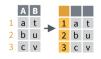
quantile() – quantil n-ésimo min() – välor mínimo max() – valor máximo

#### DISPERSIÓN

**IQR()** – rango inter quartil mad() – desviación absoluta mediana **sd()** – desvío estándar var() - varianza

## Nombres de Filas

Los datos ordenados en tibbles no usan nombres de filas, que implica valor es fuera de las columnas. Para trabajar con nombres de las filas es necesario primero moverlos a una columna.



3 c v 3 c v

A B C

rownames\_to\_column() 1 a t Mueve los nombre de filas a una columna. a <- rownames\_to\_column(iris, var = "C")

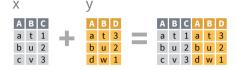


column\_to\_rownames() Mueve una col a nombre de filas. column\_to\_rownames(a, var = "C")

## También has rownames(), remove rownames()

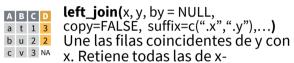
## Combinar Tablas

#### **COMBINA VARIABLES**

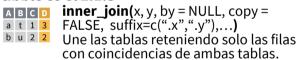


bind\_cols() sirve para adosar tablas una al lado de la otra, tal cual están. Asegúrate de que las filas estén alineadas.

Usa una Unión de Transformación (mutating join) para combinar una tabla con columnas de otra tabla, de acuerdo con las coincidencias de los valores de sus claves. Cada unión retiene una combinación diferente de los valores de cada tabla.



right\_join(x, y, by = NULL, copy =
FALSE, suffix=c(".x",".y"),...) A B C D a t 1 3 b u 2 2 Une las filas coincidentes de x con y. Retiene todas las de y.





full\_join(x, y, by = NULL, copy=FALSE, suffix=c(".x",".y"),...) Une las tablas reteniendo todos los d w NA 1 valores de todas las filas.

## A B.x C B.y D a t 1 t 3 b u 2 u 2 c v 3 NA NA

Usa by = c("col1", "col2", ...) especificar una o más columnas a usar para determinar coincidencias.  $left_{join}(x, y, by = "A")$ 



A.xB.x C A.yB.y Usa un vector con nombres, by = c("col1" = "col2"), para determinar coincidencias de columnas que tienen nombres diferentes.  $left\_join(x, y, by = c("C" = "D"))$ 



A1 B1 C A2 B2 Para columnas no coincidentes que presentan el mismo nombre, usa c v 3 a t suffix para indicar el sufijo a darle a cada columna.

 $left\_join(x, y, by = c("C" = "D"), suffix =$ c("1", "2"))

### **COMBINA CASOS**



Usa **bind rows()** para adosar tablas una debajo

АВС

a t 1

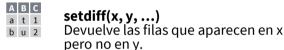
b u 2

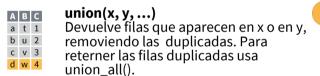
c v 3

de la otra tal cual están.

DF	Α	В	С	<pre>bind_rows(,.id = NULL)</pre>
х	а	t	1	Devuelve las tablas una debajo de la
		u		otra como una tabla única. Usa .id
			3	16
Z	С	٧	3	para demini el nombre de una columna
Z	d	w	4	nueva donde se guarde el nombre de
				las tablas originales (ver figura).

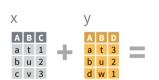
A B C intersect(x, y, ...) Devuelve las filas que aparecen tanto en x como en y.



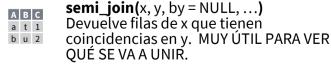


Usa **setequal()** para probar si dos datasets contienen exactamente las mismas filas (en cualquier orden).

### **EXTRAER FILAS**



Usa una "**Unión de filtro**" para filtrar una tabla usando las filas de otra tabla.



A B C anti join(x, y, by = NULL, ...) Devuelve filas de x que no tienen coincidencias en v. MUY ÚTIL PARA VER OUÉ NO SE VA A ÚNIR.

