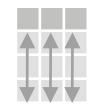


# Transformação de dados com dplyr :: FOLHA DE REFERÊNCIA



Funções dplyr funcionam com canalização (pipes) e esperam dados organizados (tidy). Em dados organizados temos:



Cada variável está em sua própria coluna

&



Cada observação, está em sua própria linha



Pipes

$x \%>\% f(y)$   
É o mesmo que  $f(x, y)$

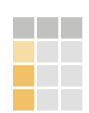
## Resumindo Observações

Aplica funções de resumo em colunas para criar uma nova tabela estatística resumida. Funções de resumo recebem vetores como entrada e retornam um único valor (vide verso).

função de resumo



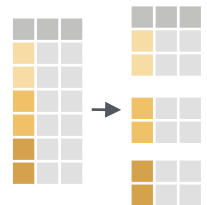
`summarise(.data, ...)`  
Computa tabela de resumo.  
`summarise(mtcars, avg = mean(mpg))`



`count(.data, ..., wt = NULL, sort = FALSE, name = NULL)` Conta número de linhas em cada grupo definidos com suas variáveis... Também tally().  
`count(mtcars, cyl)`

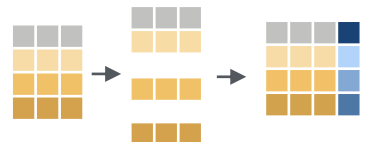
## Agrupando Observações

Use `group_by(.data, ..., .add = FALSE, .drop = TRUE)` para criar uma cópia da tabela agrupada por colunas ... As funções do dplyr irão manipular cada grupo separadamente e combinar os resultados.



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

Use `rowwise(.data, ...)` para agrupar dados em linhas individuais. Funções do dplyr irão computar os resultados para cada linha. Também aplica funções em colunas de listas. Veja a folha de referência do tidyr sobre o fluxo de colunas de listas.



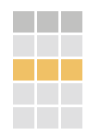
`starwars %>%  
rowwise() %>%  
mutate(film_count = length(films))`

`ungroup(x, ...)` Retorna uma cópia desagrupada da tabela.  
`ungroup(g_mtcars)`

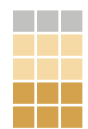
## Manipulando Observações

### EXTRAÇÃO DE OBSERVAÇÕES

Funções de linhas retornam um subconjunto de linhas como uma nova tabela.



`filter(.data, ..., .preserve = FALSE)` Extrai linhas que satisfazem o critério lógico.  
`filter(mtcars, mpg > 20)`



`distinct(.data, ..., .keep_all = FALSE)` Remove linhas com valores duplicados.  
`distinct(mtcars, gear)`



`slice(.data, ..., .preserve = FALSE)` Seleciona linhas pela posição.  
`slice(mtcars, 10:15)`



`slice_sample(.data, ..., n, prop, weight_by = NULL, replace = FALSE)` Randomicamente seleciona linhas. Use `n` para selecionar o número de linhas e `prop` para selecionar um percentual das linhas.  
`slice_sample(mtcars, n = 5, replace = TRUE)`



`slice_min(.data, order_by, ..., n, prop, with_ties = TRUE)` and `slice_max()` Seleciona linhas com valores mínimo e máximo.  
`slice_min(mtcars, mpg, prop = 0.25)`

`slice_head(.data, ..., n, prop)` and `slice_tail()` Seleciona as primeiras or últimas linhas.  
`slice_head(mtcars, n = 5)`

### Operadores Lógicos e Booleanos para usar com filter()

`==` `<` `<=` `is.na()` `%in%` `|` `xor()`

`!=` `>` `>=` `!is.na()` `!` `&`

Veja `?base::Logic` e `?Comparison` para ajuda.

### ARRANJAR OBSERVAÇÕES



`arrange(.data, ..., .by_group = FALSE)` Ordena linhas por valores de uma coluna ou colunas (menor para maior), use com `desc()` para ordenar de maior para menor.  
`arrange(mtcars, mpg)`  
`arrange(mtcars, desc(mpg))`

### ADICIONAR OBSERVAÇÕES



`add_row(.data, ..., .before = NULL, .after = NULL)` Adiciona uma ou mais linhas em uma tabela.  
`add_row(cars, speed = 1, dist = 1)`

## Manipulando Variáveis

### EXTRAÇÃO DE VARIÁVEIS

Funções de colunas retornam um conjunto de colunas como um novo vetor ou tabela.



`pull(.data, var = -1, name = NULL, ...)` Extrai valores da coluna como um vetor, por nome ou índice.  
`pull(mtcars, wt)`



`select(.data, ...)` Extrai colunas como uma tabela.  
`select(mtcars, mpg, wt)`



`relocate(.data, ..., .before = NULL, .after = NULL)` Move colunas para uma nova posição.  
`relocate(mtcars, mpg, cyl, .after = last_col())`

Use estes complementos com `select()` e `across()`

e.g. `select(mtcars, mpg:cyl)`

`contains(match)`

`ends_with(match)`

`starts_with(match)`

`num_range(prefix, range)`

`all_of(x)/any_of(x, ..., vars)`

`matches(match)`

;, e.g. `mpg:cyl`

-, e.g. `-gear`

`everything()`

### MANIPULATE MULTIPLE VARIABLES AT ONCE



`across(.cols, .funs, ..., .names = NULL)` Summarise or mutate multiple columns in the same way.  
`summarise(mtcars, across(everything(), mean))`



`c_across(.cols)` Compute across columns in row-wise data.  
`transmute(rowwise(UKgas), total = sum(c_across(1:2)))`

### CRIANDO NOVAS VARIÁVEIS

Aplica funções vetorizadas em colunas. Funções vetorizadas recebem vetores como entradas e retornam vetores do mesmo tamanho como saída (vide verso).

Função vetorizada



`mutate(.data, ..., .keep = "all", .before = NULL, .after = NULL)` Computa nova(s) coluna(s). Veja também `add_column()`, `add_count()`, e `add_tally()`.  
`mutate(mtcars, gpm = 1 / mpg)`



`transmute(.data, ...)` Computa nova(s) coluna(s) e descarta as demais.  
`transmute(mtcars, gpm = 1 / mpg)`



`rename(.data, ...)` Renomeia colunas. Use `rename_with()` para renomear usando uma função.  
`rename(cars, distance = dist)`



## Funções Vetorizadas

TPARA USAR COM MUTATE ()

mutate() e transmute() aplicam funções vetorizadas em colunas para criar novas colunas. Funções vetorizadas recebem vetores como argumento de entrada e retornar vetores de mesmo tamanho como saída.

Função vetorizada

### DESLOCAMENTO

dplyr::lag() – desloca elementos em 11  
dplyr::lead() – desloca elementos em -1

### AGREGAÇÃO ACUMULADA

dplyr::cumall() – acumulado de all()  
dplyr::cumany() – acumulado de any()  
cummax() – acumulado de max()  
dplyr::cummean() – acumulado de mean()  
cummin() – acumulado de min()  
cumprod() – acumulado de prod()  
cumsum() – acumulado de sum()

### RANQUEAMENTO

dplyr::cume\_dist() – proporção de todos valores <=  
dplyr::dense\_rank() – ranq. sem brechas  
dplyr::min\_rank() – ranq. com empates = min  
dplyr::ntile() – intervalões em n intervalos  
dplyr::percent\_rank() - min\_rank escalado até [0,1]  
dplyr::row\_number() – ranq. empates = “primeiro”

### MATH

+, -, \*, /, ^, %/%, %% - oper. aritméticas  
log(), log2(), log10() - logs  
<, <=, >, >=, !=, == - comparações lógicas  
dplyr::between() - x >= esquerda & x <= direita  
dplyr::near() - == seguro para números com pontos flutuantes

### MISCELÂNEA

dplyr::case\_when() - if\_else() de vários casos  
starwars %>%  
mutate(type = case\_when(  
height > 200 | mass > 200 ~ "large",  
species == "Droid" ~ "robot",  
TRUE ~ "other")  
)

dplyr::coalesce() – primeiro valor não-NA por elemento através de um conjunto de vetores  
dplyr::if\_else() - if() + else() elemento por elemento  
dplyr::na\_if() – altera um valores específico para NA  
pmax() - max() elemento por elemento  
pmin() - min() elemento por elemento

## Funções de Resumo

PARA USAR COM SUMMARISE ()

summarise() aplica funções de resumo em colunas para criar uma nova tabela. Funções de resumo recebem vetores como entrada e retornam um valor único na saída.

Função de resumo

### CONTAGEM

dplyr::n() – número de valores/linhas  
dplyr::n\_distinct() - # de valores únicos  
sum(!is.na()) - # de não-NA's

### POSIÇÃO

mean() - média, também mean(!is.na())  
median() - mediana

### LÓGICA

mean() – proporção de verdadeiros (TRUE)  
sum() - # de verdadeiros (TRUE)

### ORDEM

dplyr::first() – primeiro valor  
dplyr::last() – último valor  
dplyr::nth() – valor na enésima posição do vetor

### RANQUEAMENTO

quantile() - enésimo quartil  
min() – valor mínimo  
max() – valor máximo

### DISPERSÃO

IQR() – distância inter-quartil  
mad() – desvio absoluto médio  
sd() – desvio padrão  
var() - variância

## Nome de Linhas

Dados organizados (tidy) não usam nomes de linhas (que contém uma variável fora das colunas). Para trabalhar com este nomes, mova para uma coluna.

tibble::rownames\_to\_column()  
Move nomes de linhas para coluna.  
a <- rownames\_to\_column(mtcars,  
var = "C")

tibble::column\_to\_rownames()  
Move coluna como nome das linhas.  
column\_to\_rownames(a, var = "C")

Veja também tibble::has\_rownames() e  
tibble::remove\_rownames().

## Combinando Tabelas

JUNTANDO VARIÁVEIS

X	Y	
A B C	E F G	A B C E F G
a t 1	a t 3	a t 1 a t 3
b u 2	b u 2	b u 2 b u 2
c v 3	d w 1	c v 3 d w 1

bind\_cols(..., .name\_repair) Retorna tabelas colocadas lado a lado como um tabela única. Comprimento das colunas devem ser iguais. Colunas não serão combinadas por id (para isso veja Dados Relacionais abaixo), então certifique-se que ambas as tabelas estão ordenadas como você deseja antes de uni-las.

### RELATIONAL DATA

Use uma "União Transformadora" para unir uma tabela com colunas de outra, combinando valores de linhas correspondentes. Cada união (join) retem uma combinação diferente de valores das tabelas.

left\_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ..., keep = FALSE, na\_matches = "na") Une valores iguais de y em x.

right\_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ..., keep = FALSE, na\_matches = "na") Une valores iguais de x em y.

inner\_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ..., keep = FALSE, na\_matches = "na") Une todos os dados. Retem somente linhas em comum.

full\_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ..., keep = FALSE, na\_matches = "na") Ene dados, retem todos os valores e todas as linhas.

COMBINANDO COLUNAS PARA UNIÕES

Use by = c("col1", "col2", ...) para definir uma ou mais colunas de combinação.  
left\_join(x, y, by = "A")

Use um vetor, by = c("col1" = "col2"), para combinar colunas com nomes diferentes em cada tabela.  
left\_join(x, y, by = c("C" = "D"))

Use suffix para definir o sufixo para colunas não combinadas que tem o mesmo nome em tabelas diferentes.  
left\_join(x, y, by = c("C" = "D"), suffix = c("1", "2"))

JUNTANDO OBSERVAÇÕES

X	Y	
A B C	A B C	DF A B C
a t 1	a t 1	x a t 1
b u 2	b u 2	x b u 2
	c v 3	y c v 3
	d w 4	y d w 4

bind\_rows(..., .id = NULL) Retorna tabelas uma em cima da outra como uma tabela única. Defina .id para nome de coluna para incluir uma coluna com a tabela original (conforme figura ao lado).

Use uma "União de Filtro" para filtrar uma tabela conforme linhas de uma outra tabela.

semi\_join(x, y, by = NULL, copy = FALSE, ..., na\_matches = "na") Retorna linhas de x que estão presentes em y. Use para ver o que será incluído em uma união.

anti\_join(x, y, by = NULL, copy = FALSE, ..., na\_matches = "na") Retorna linhas de x que não estão presentes em y. Use para ver o que NÃO será incluído em uma união..

Use uma "União de Aninhamento" para inserir uma tabela em um data frame.

nest\_join(x, y, by = NULL, copy = FALSE, keep = FALSE, name = NULL, ...) Une dados, aninhando combinações de y em uma nova coluna de um data frame.

OPERAÇÕES DE DEFINIÇÃO

intersect(x, y, ...) Linhas que aparecem em x e y.

setdiff(x, y, ...) Linhas que aparecem em x mas não em y.

union(x, y, ...) Linhas que aparecem em x ou y. (Remove duplicadas). union\_all() mantém duplicadas.

Use setequal() para testar se dois conjunto de dados contém as mesmas linhas (em qualquer ordem).