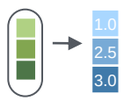
Aplicando funções com purrr : : FOLHA DE RESUMO

Funções Map

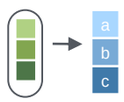
UMA LISTA

map(.x, .f, …) Aplica uma função em cada elemento da lista ou vetor e retorna uma lista.
x <- list(1:10, 11:20, 21:30)
l1 <- list(x = c("a", "b"), y = c("c", "d"))
map(l1, sort, decreasing = TRUE)



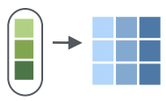
map_dbl(.x, .f, …)
Retorna um vetor de double.
map_dbl(x, mean)

map_int(.x, .f, ...)
Retorna um vetor de inteiro.
map_int(x, length)



map_chr(.x, .f, …)
Retorna um vetor de caractere.
map_chr(l1, paste, collapse = "")

map_lgl(.x, .f, …)
Retorna um vetor de lógico.
map_lgl(x, is.integer)



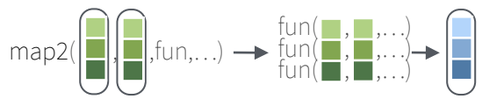
map_dfc(.x, .f, ...)
Retorna um data frame criado com junção de colunas.
map_dfc(l1, rep, 3)

map_dfr(.x, .f, ..., .id = NULL)
Retorna um data frame criado com junção de linhas.
map_dfr(x, summary)

walk(.x, .f, ...) Executa o map silencioso.
walk(x, print)

DUAS LISTAS

map2(.x, .y, .f, …) Aplica uma função a um par de listas ou vetores e retorna uma lista.
y <- list(1, 2, 3); z <- list(4, 5, 6); l2 <- list(x = "a", y = "z")
map2(x, y, ~ .x * .y)



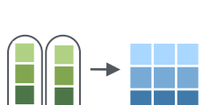
map2_dbl(.x, .y, .f, …)
Retorna um vetor de double.
map2_dbl(y, z, ~ .x / .y)

map2_int(.x, .y, .f, …)
Retorna um vetor de inteiro.
map2_int(y, z, `+`)

map2_chr(.x, .y, .f, …)
Retorna um vetor de caractere.
map2_chr(l1, l2, paste, collapse = ",", sep = ":")

map2_lgl(.x, .y, .f, …)
Retorna um vetor lógico..
map2_lgl(l2, l1, `%in%`)

map2_dfc(.x, .y, .f, ...)
Retorna um data frame com junção de colunas.
map2_dfc(l1, l2,
~ as.data.frame(c(.x, .y)))



map2_dfr(.x, .y, .f, ..., .id = NULL) Retorna um data frame criado com junção de linhas.
map2_dfr(l1, l2,
~ as.data.frame(c(.x, .y)))

walk2(.x, .y, .f, ...) Executa o map2 silencioso.
walk2(objs, paths, save)

VÁRIAS LISTAS

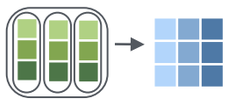
pmap(.l, .f, …) Aplica uma função à grupos de elementos de uma lista ou listas de vetores e retorna uma lista.
pmap(list(x, y, z), ~ ..1 * (..2 + ..3))

pmap_dbl(.l, .f, …)
Retorna um vetor de double.
pmap_dbl(list(y, z), ~ .x / .y)

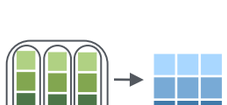
pmap_int(.l, .f, …)
Retorna um vetor de inteiro.
pmap_int(list(y, z), `+`)

pmap_chr(.l, .f, …)
Retorna um vetor de caractere.
pmap_chr(list(l1, l2), paste, collapse = ",", sep = ":")

pmap_lgl(.l, .f, …)
Retorna um logical vector.
pmap_lgl(list(l2, l1), `%in%`)



pmap_dfc(.l, .f, ...) Retorna um data frame com junção de colunas.
pmap_dfc(list(l1, l2),
~ as.data.frame(c(.x, .y)))

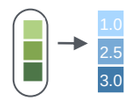


pmap_dfr(.l, .f, ..., .id = NULL) Retorna um data frame criado com junção de linhas.
pmap_dfr(list(l1, l2),
~ as.data.frame(c(.x, .y)))

pwalk(.l, .f, ...) Executa o pmap silencioso.
pwalk(list(objs, paths), save)

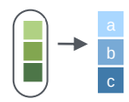
LISTAS E ÍNDICES

imap(.x, .f, ...) Aplica uma função a cada elemento de uma lista e seu índice e retorna uma lista.
imap(y, ~ paste0(.y, ": ", .x))



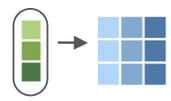
imap_dbl(.x, .f, …)
Retorna um vetor de double.
imap_dbl(y, ~ .y)

imap_int(.x, .f, ...)
Retorna um vetor de inteiro.
imap_int(y, ~ .y)



imap_chr(.x, .f, …)
Retorna um vetor de caractere.
imap_chr(y, ~ paste0(.y, ": ", .x))

imap_lgl(.x, .f, …)
Retorna um logical vector.
imap_lgl(l1, ~ is.character(.y))



imap_dfc(.x, .f, ...)
Retorna um data frame criado com junção de colunas.
imap_dfc(l2,
~ as.data.frame(c(.x, .y)))

imap_dfr(.x, .f, ..., .id = NULL) Retorna um data frame criado com junção de linhas.
imap_dfr(l2,
~ as.data.frame(c(.x, .y)))

iwalk(.x, .f, ...) Executa o imap silencioso.
iwalk(z, ~ print(paste0(.y, ": ", .x)))

Atalhos de Funções

Use ~ . Com função como map() que tem um único argumento.

map(l, ~ . + 2)
é o mesmo que
map(l, function(x) x + 2))

Use ~ .x .y com função como o map2() que tem dois argumentos.

map2(l, p, ~ .x +.y)
é o mesmo que
map2(l, p, function(l, p) l + p)

Use ~ ..1 ..2 ..3 etc com função como o pmap() que tem vários argumentos.

pmap(list(a, b, c), ~ ..3 + ..1 - ..2)
é o mesmo que
pmap(list(a, b, c), function(a, b, c) c + a -b)

Use ~ .x .y com função como o imap() .x recebe o valor da lista e .y recebe os índices da lista.

imap(list(a, b, c), ~ paste0(.y, ": ", .x)
retorna "index: value" para cada item

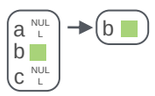
Use uma string ou um inteiro com qualquer função map para indexar os elementos por nome ou posição. map(l, "name") se torna map(l, function(x) x[["name"]])

Traduzido por: Eric Scopinho • linkedin.com/in/scopinho

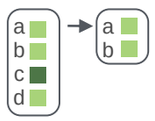
Work with Lists

Filtrar

keep(.x, .p, …)
Seleciona elementos que passam no teste lógico. Para oposto, discard().
keep(x, is.na)



compact(.x, .p = identity)
Elimina elementos vazios.
compact(x)



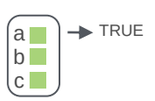
head_while(.x, .p, …)
Retorna elementos do topo até um não passar no teste lógico. Ver também tail_while().
head_while(x, is.character)

detect(.x, .f, ..., dir = c("forward", "backward"), .right = NULL, .default = NULL) Retorna o primeiro elemento que passa no teste. detect(x, is.character)

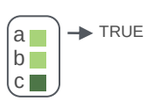
detect_index(.x, .f, ..., dir = c("forward", "backward"), .right = NULL) Retorna o índice do primeiro elemento que passa no teste.
detect_index(x, is.character)

every(.x, .p, …)
Todos elementos passam no teste?
every(x, is.character)

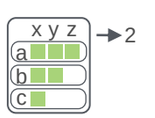
some(.x, .p, …)
Algum elementos passa no teste?
some(x, is.character)



none(.x, .p, …)
Nenhum elemento passa no teste?
none(x, is.character)

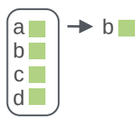


has_element(.x, .y)
Does a list contain an element?
has_element(x, "foo")

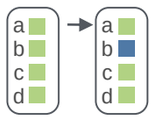


vec_depth(x)
Return depth (number of levels of indexes).
vec_depth(x)

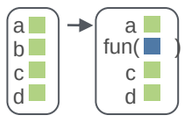
Indexar



pluck(.x, ..., .default=NULL)
Seleciona um elemento por nome ou posição. Ver attr_getter() e chuck().
pluck(x, "b")
x %>% pluck("b")

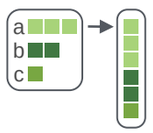


assign_in(x, where, value)
Define um valor para uma posição usando seleção do pluck.
assign_in(x, "b", 5)
x %>% assign_in("b", 5)



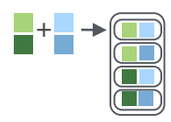
modify_in(.x, .where, .f)
Aplica uma função para o valor em certa posição.
modify_in(x, "b", abs)
x %>% modify_in("b", abs)

Remodelar

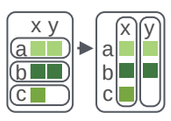


flatten(.x) Remove a level of indexes from a list.
Also flatten_chr() etc.
flatten(x)

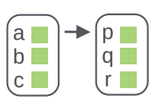
array_tree(array, margin = NULL) Turn array into list.
Also array_branch().
array_tree(x, margin = 3)



cross2(.x, .y, .filter = NULL)
All combinations of .x and .y.
Also cross(), cross3(), and cross_df().
cross2(1:3, 4:6)



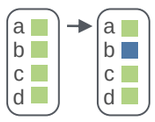
transpose(.l, .names = NULL)
Transposes the index order in a multi-level list.
transpose(x)



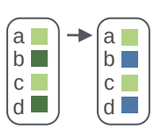
set_names(x, nm = x)
Set the names of a vector/list directly or with a function.
set_names(x, c("p", "q", "r"))
set_names(x, tolower)

Modificar

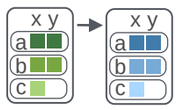
modify(.x, .f, ...) Aplica uma função em cada elemento. Ver modify2() e imodify().
modify(x, ~.+ 2)



modify_at(.x, .at, .f, ...) Aplica uma função aos elementos selecionados. Ver map_at().
modify_at(x, "b", ~.+ 2)



modify_if(.x, .p, .f, ...) Aplica uma função aos elementos que passam no teste. Ver map_if().
modify_if(x, is.numeric,~.+2)

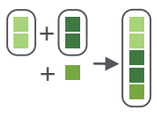


modify_depth(.x, .depth, .f, ...) Aplica uma função a cada elemento dado o nível da lista. Ver map_depth().
modify_depth(x, 2, ~.+ 2)

Combinar

append(x, values, after = length(x)) Adiciona valores ao final da lista.
append(x, list(d = 1))

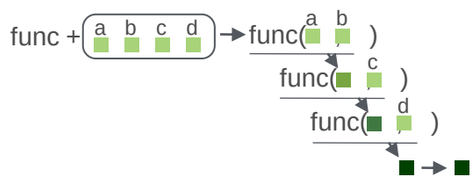
prepend(x, values, before = 1) Adiciona valores no início da lista.
prepend(x, list(d = 1))



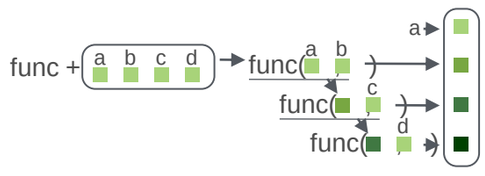
splice(…) Combine objetos em uma lista, armazenando objetos S3 como sub-listas.
splice(x, y, "foo")

Reduzir

reduce(.x, .f, ..., .init, .dir = c("forward", "backward")) Aplica uma função recursivamente em cada elemento de uma lista ou vetor. Ver também reduce2().
reduce(x, sum)



accumulate(.x, .f, ..., .init) Reduz uma lista, mas retornando os resultados intermediários. Ver também accumulate2().
accumulate(x, sum)



Colunas de Listas

max	seq
3	<int [3]>
4	<int [4]>
5	<int [5]>

Colunas de listas são colunas em um data frame onde cada elemento é uma lista ou vetor ao invés de uma valor atômico. Colunas de listas também podem conter outros data frames. Ver tidyr para mais info sobre dados aninhados e colunas de listas.

TRABALHE COM COLUNAS DE LISTAS

Manipular colunas de listas é como qualquer outro tipo de coluna, usando funções dplyr como mutate() ou transmute(). Como cada elemento é uma lista, use funçãp map dentro da coluna para manipular cada elemento.

map(), map2() ou pmap() retorna uma lista e irá criar uma coluna de lista.

```
starwars %>%
  transmute(ships = map2(vehicles,
                         starships,
                         append)
```

Função de lista retorna lista
Coluna de lista
Função de coluna

Função derivada de map como map_int() retorna um tipo de dado atômico, portanto simplifica a coluna de lista como uma coluna normal.

```
starwars %>%
  mutate(n_films = map_int(films, length))
```

Função de lista retorna int
Função de coluna
Coluna de lista

Traduzido por: Eric Scopinho • linkedin.com/in/scopinho