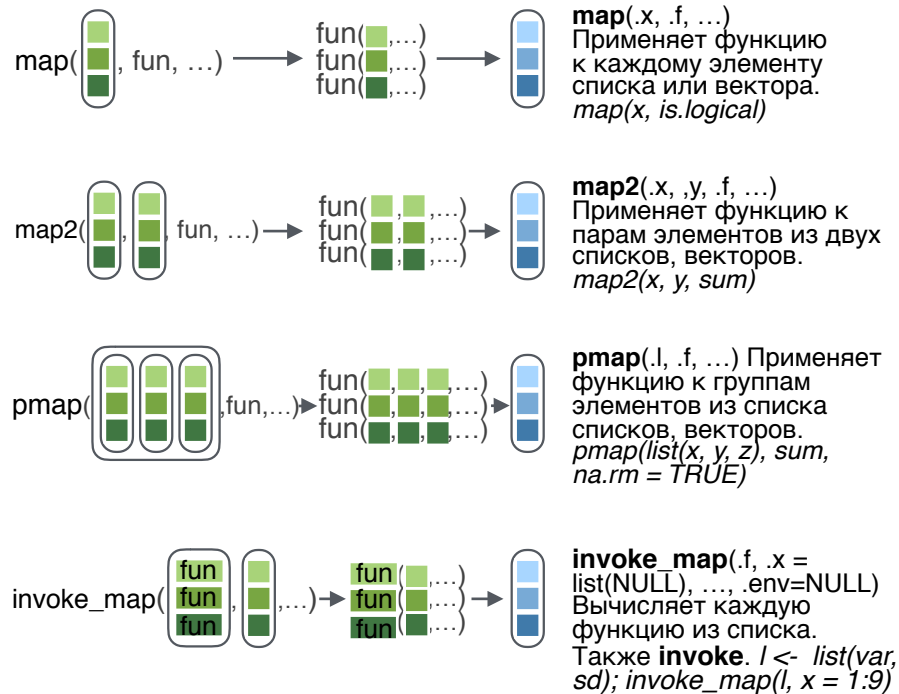


Применение функций с purrr : : ШПАРГАЛКА



Применение функций

Map-функции итеративно применяют функции к каждому элементу списка или вектора.



lmap(x, .f, ...) Применяет .f к списковым элементам списка, вектора.
imap(x, .f, ...) Применяет .f к элементам списка, вектора и его индексу.

РЕЗУЛЬТАТ

map(), **map2()**, **pmap()**, **lmap** и **invoke_map** возвращают список. Используйте версии с суффиксом для получения результата определенного типа, например **map2_chr**, **pmap_lgl**, и т.д.

Используйте **walk**, **walk2** и **pwalk** для совершения побочных действий. Все невидимо возвращают входные данные.

функция	результат
map	список
map_chr	символьный вектор
map_dbl	числовой вектор
map_dfc	data frame (соед. по столбцам)
map_dfr	data frame (соед. по строкам)
map_int	целочисленный вектор
map_lgl	логический вектор
walk	совершает побоч. действия, невидимо возвращ. входные данные

СОКРАЩЕНИЯ - внутри функций purrr:

"name" становится **function(x) x[["name"]]**, напр. `map(l, "a")` извлекает `a` из каждого элемента `l`

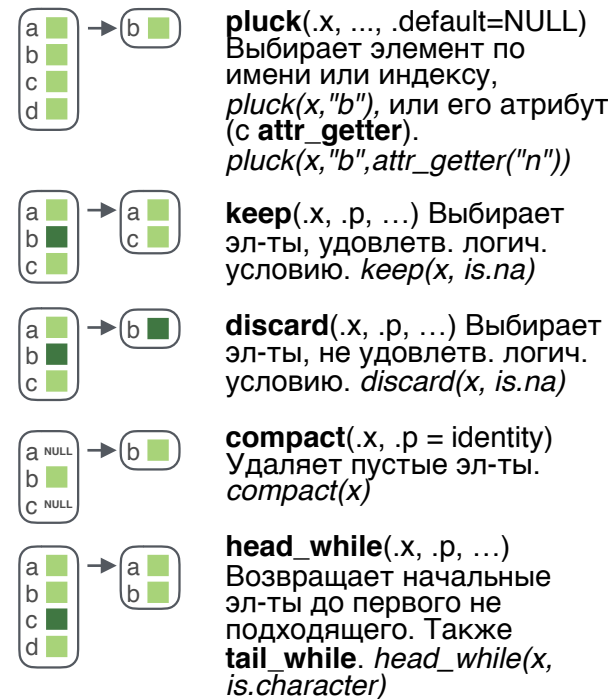
~ .x .y становится **function(x, y) .x .y**, напр. `map2(l, p, ~ .x + .y)` становится `map2(l, p, function(l, p) l + p)`

~ . становится **function(x) x**, напр. `map(l, ~ 2 + .)` становится `map(l, function(x) 2 + x)`

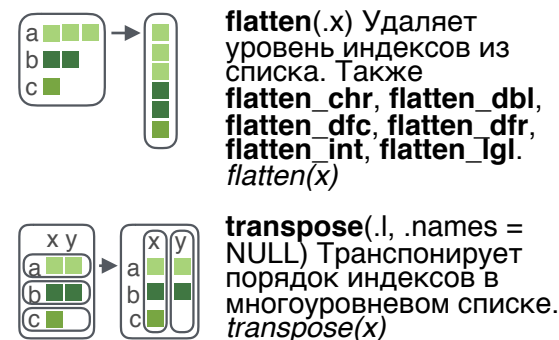
~ ..1 ..2 etc становится **function(..1, ..2, etc) ..1 ..2 etc** напр. `pmap(list(a, b, c), ~ ..3 + ..1 - ..2)` становится `pmap(list(a, b, c), function(a, b, c) c + a - b)`

Работа со списками

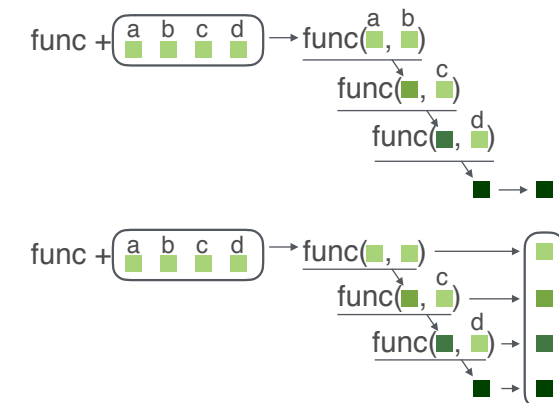
ФИЛЬТРОВАНИЕ СПИСКОВ



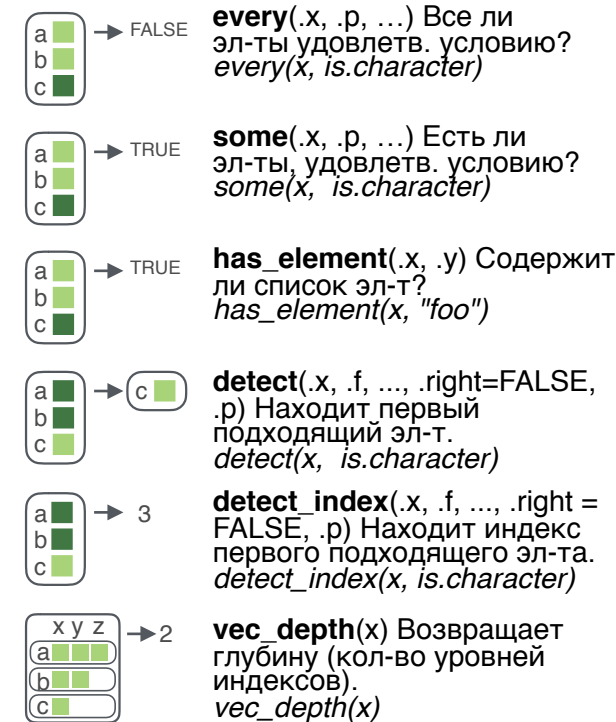
ФОРМАТИРОВАНИЕ СПИСКОВ



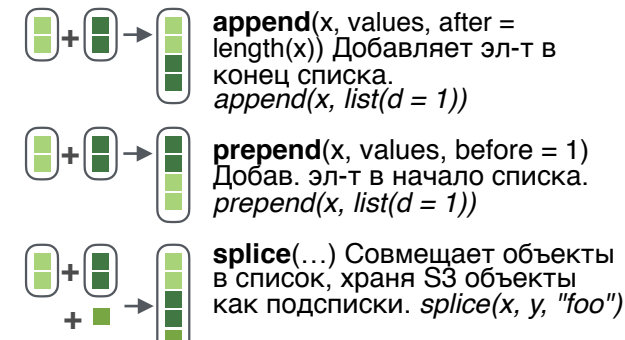
Свертка списков



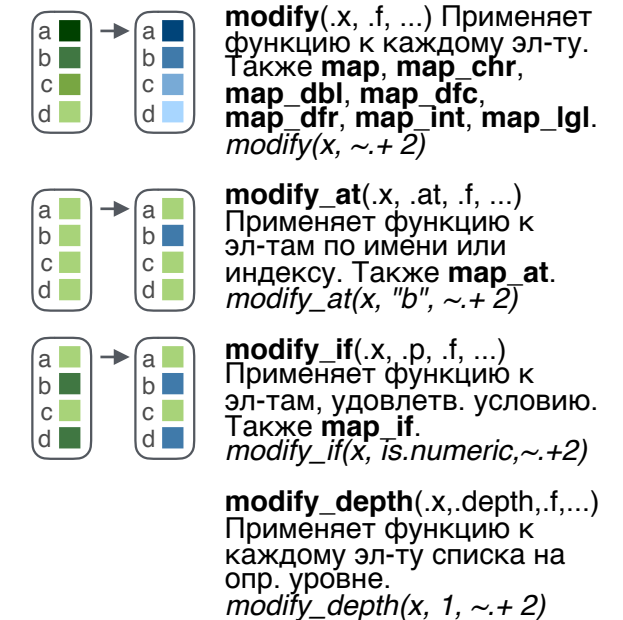
СУММИРОВАНИЕ СПИСКОВ



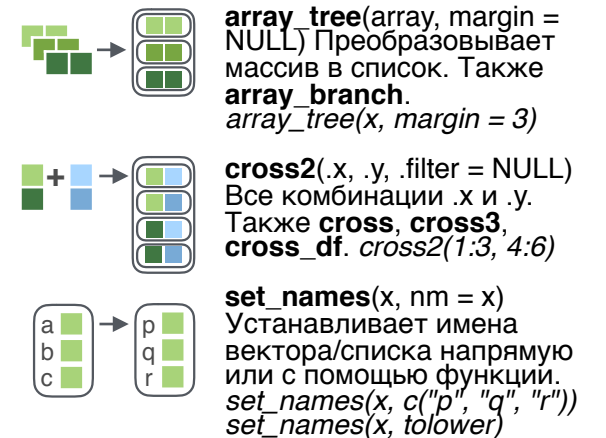
СОЕДИНЕНИЕ СПИСКОВ



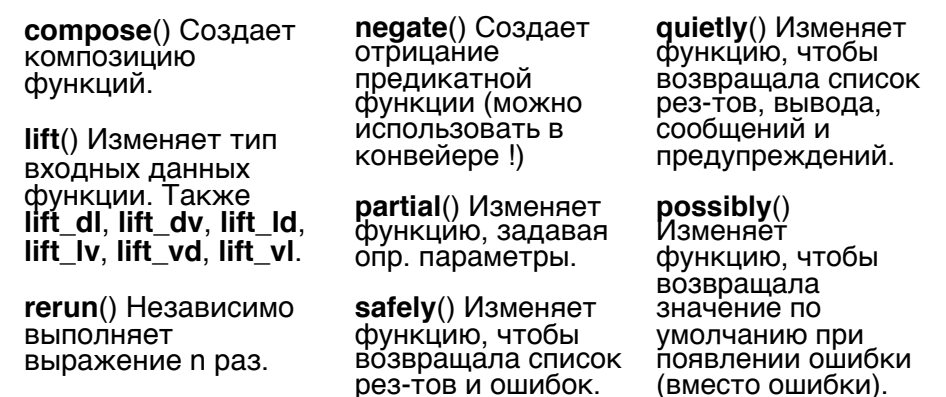
ПРЕОБРАЗОВАНИЕ СПИСКОВ



РАБОТА СО СПИСКАМИ



Изменение функций



Вложенные данные

Вложенный data frame хранит отдельные таблицы внутри ячеек большей упорядочивающей таблицы

вложенный data frame

Species	data
setosa	<tibble [50 x 4]>
versicolor	<tibble [50 x 4]>
virginica	<tibble [50 x 4]>

n_iris

Используйте вложенный data frame для:

- сохранения отношений между наблюдениями и подмножествами данных

- манипулирования сразу несколькими подтаблицами с **purrr** функциями **map()**, **map2()** или **pmap()**.

Используйте двухэтапный процесс для создания вложенного data frame:

1. Сгруппируйте data frame в группы с помощью **dplyr::group_by()**
2. Используйте **nest()** для создания вложенного data frame с одной строкой на группу.

Species	S.L	S.W	P.L	P.W
setosa	5.1	3.5	1.4	0.2
setosa	4.9	3.0	1.4	0.2
setosa	4.7	3.2	1.3	0.2
setosa	4.6	3.1	1.5	0.2
setosa	5.0	3.6	1.4	0.2
versi	7.0	3.2	4.7	1.4
versi	6.4	3.2	4.5	1.5
versi	6.9	3.1	4.9	1.5
versi	5.5	2.3	4.0	1.3
versi	6.5	2.8	4.6	1.5
virgini	6.3	3.3	6.0	2.5
virgini	5.8	2.7	5.1	1.9
virgini	7.1	3.0	5.9	2.1
virgini	6.3	2.9	5.6	1.8
virgini	6.5	3.0	5.8	2.2

Species	S.L	S.W	P.L	P.W
setosa	5.1	3.5	1.4	0.2
setosa	4.9	3.0	1.4	0.2
setosa	4.7	3.2	1.3	0.2
setosa	4.6	3.1	1.5	0.2
setosa	5.0	3.6	1.4	0.2
versi	7.0	3.2	4.7	1.4
versi	6.4	3.2	4.5	1.5
versi	6.9	3.1	4.9	1.5
versi	5.5	2.3	4.0	1.3
versi	6.5	2.8	4.6	1.5
virgini	6.3	3.3	6.0	2.5
virgini	5.8	2.7	5.1	1.9
virgini	7.1	3.0	5.9	2.1
virgini	6.3	2.9	5.6	1.8
virgini	6.5	3.0	5.8	2.2

содержимое "ячеек"

Sepal.L	Sepal.W	Petal.L	Petal.W
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2

n_iris\$data[[1]]

Sepal.L	Sepal.W	Petal.L	Petal.W
7.0	3.2	4.7	1.4
6.4	3.2	4.5	1.5
6.9	3.1	4.9	1.5
5.5	2.3	4.0	1.3
6.5	2.8	4.6	1.5

n_iris\$data[[2]]

Sepal.L	Sepal.W	Petal.L	Petal.W
6.3	3.3	6.0	2.5
5.8	2.7	5.1	1.9
7.1	3.0	5.9	2.1
6.3	2.9	5.6	1.8
6.5	3.0	5.8	2.2

n_iris\$data[[3]]

n_iris <- iris %>% group_by(Species) %>% nest()

tidyr::nest(data, ..., key = data)

Сворачивает группы в ячейки в виде data frame (для сгрупп. данных).

Разворачивайте вложенный data frame при помощи **unnest()**:

n_iris %>% unnest()

tidyr::unnest(data, ..., .drop = NA, .id=NULL, .sep=NULL)

Разворачивает вложенный data frame.

Species	data
setos	<tibble [50x4]>
versi	<tibble [50x4]>
virgini	<tibble [50x4]>

Species	S.L	S.W	P.L	P.W
setosa	5.1	3.5	1.4	0.2
setosa	4.9	3.0	1.4	0.2
setosa	4.7	3.2	1.3	0.2
setosa	4.6	3.1	1.5	0.2
setosa	5.0	3.6	1.4	0.2
versi	7.0	3.2	4.7	1.4
versi	6.4	3.2	4.5	1.5
versi	6.9	3.1	4.9	1.5
versi	5.5	2.3	4.0	1.3
versi	6.5	2.8	4.6	1.5
virgini	6.3	3.3	6.0	2.5
virgini	5.8	2.7	5.1	1.9
virgini	7.1	3.0	5.9	2.1
virgini	6.3	2.9	5.6	1.8
virgini	6.5	3.0	5.8	2.2

Процесс работы со СПИСКОВЫМИ СТОЛБЦАМИ

1 Создание спискового столбца

Species	S.L	S.W	P.L	P.W
setosa	5.1	3.5	1.4	0.2
setosa	4.9	3.0	1.4	0.2
setosa	4.7	3.2	1.3	0.2
setosa	4.6	3.1	1.5	0.2
setosa	5.0	3.6	1.4	0.2
versi	7.0	3.2	4.7	1.4
versi	6.4	3.2	4.5	1.5
versi	6.9	3.1	4.9	1.5
versi	5.5	2.3	4.0	1.3
versi	6.3	3.3	6.0	2.5
virgini	5.8	2.7	5.1	1.9
virgini	7.1	3.0	5.9	2.1
virgini	6.3	2.9	5.6	1.8

Species	data
setos	<tibble [50x4]>
versi	<tibble [50x4]>
virgini	<tibble [50x4]>

**n_iris <- iris %>%
group_by(Species) %>%
nest()**

S.L	S.W	P.L	P.W
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
7.0	3.2	4.7	1.4
6.4	3.2	4.5	1.5
6.9	3.1	4.9	1.5
5.5	2.3	4.0	1.3
6.3	3.3	6.0	2.5
5.8	2.7	5.1	1.9
7.1	3.0	5.9	2.1
6.3	2.9	5.6	1.8

2 Работа со СПИСКОВЫМИ столбцами

Species	data	model
setosa	<tibble [50x4]>	<S3: lm>
versi	<tibble [50x4]>	<S3: lm>
virgini	<tibble [50x4]>	<S3: lm>

**mod_fun <- function(df)
lm(Sepal.Length ~ ., data = df)**

**m_iris <- n_iris %>%
mutate(model = map(data, mod_fun))**

3 Упрощение СПИСКОВОГО столбца

Species	beta
setos	2.35
versi	1.89
virgini	0.69

**b_fun <- function(mod)
coefficients(mod)[1]]**

**m_iris %>% transmute(Species,
beta = map_dbl(model,
b_fun))**



1. СОЗДАНИЕ СПИСКОВОГО СТОЛБЦА - создавайте с помощью функций из пакетов **tibble** и **dplyr**, а также **tidyr::nest()**

tibble::tribble(...)

Создает списковый столбец при необходимости

**tribble(~max, ~seq,
3, 1:3,
4, 1:4,
5, 1:5)**

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

tibble::tibble(...)

Сохраняет список как списковый столбец.
tibble(max = c(3, 4, 5), seq = list(1:3, 1:4, 1:5))

tibble::enframe(x, name="name", value="value")
Конвертирует многоуровневый список в tibble со списковыми столбцами.
enframe(list('3'=1:3, '4'=1:4, '5'=1:5), 'max', 'seq')

dplyr::mutate(.data, ...) Также **transmute()**

Возвращает списковый столбец, когда результат - список.
mtcars %>% mutate(seq = map(cyl, seq))

dplyr::summarise(.data, ...)
Возвращает списковый столбец, когда результат создан с **list()**.
**mtcars %>% group_by(cyl) %>%
summarise(q = list(quantile(mpg)))**

2. РАБОТА СО СПИСКОВЫМИ СТОЛБЦАМИ - Используйте функции **purrr map()**, **map2()** и **pmap()** для поэлементного применения функции с возвращением результата в списковый столбец. **walk()**, **walk2()** и **pwalk()** работают аналогично, но совершают побочные действия.

purrr::map(.x, .f, ...)

Применяет .f поэлементно к .x как .f(.x)
n_iris %>% mutate(n = map(data, dim))

purrr::map2(.x, .y, .f, ...)

Применяет .f поэлементно к .x и .y как .f(.x, .y)
m_iris %>% mutate(n = map2(data, model, list))

purrr::pmap(.l, .f, ...)

Применяет .f поэлементно к векторам из .l
**m_iris %>%
mutate(n = pmap(list(data, model, data), list))**

data	fun	result
<tibble [50x4]>	fun	result 1
<tibble [50x4]>	fun	result 2
<tibble [50x4]>	fun	result 3

data	model	fun	result
<tibble [50x4]>	<S3: lm>	fun	result 1
<tibble [50x4]>	<S3: lm>	fun	result 2
<tibble [50x4]>	<S3: lm>	fun	result 3

data	model	funcs	result
<tibble [50x4]>	<S3: lm>	coef	result 1
<tibble [50x4]>	<S3: lm>	AIC	result 2
<tibble [50x4]>	<S3: lm>	BIC	result 3

3. УПРОЩЕНИЕ СПИСКОВОГО СТОЛБЦА (в обычный столбец)

Используйте функции **purrr map_lgl()**, **map_int()**, **map_dbl()**, **map_chr()**, а также **tidyr::unnest()** для преобразования спискового столбца в обычный.

purrr::map_lgl(.x, .f, ...) Применяет .f поэл-но к .x, возвращает логич. вектор
n_iris %>% transmute(n = map_lgl(data, is.matrix))
purrr::map_int(.x, .f, ...) Применяет .f поэл-но к .x, возвращает целочисл. вектор
n_iris %>% transmute(n = map_int(data, nrow))

purrr::map_dbl(.x, .f, ...) Применяет .f поэл-но к .x, возвращает числ. вектор
n_iris %>% transmute(n = map_dbl(data, nrow))
purrr::map_chr(.x, .f, ...) Применяет .f поэл-но к .x, возвращает симв. вектор
n_iris %>% transmute(n = map_chr(data, nrow))