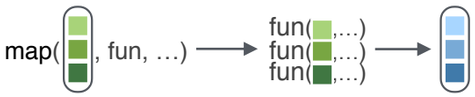
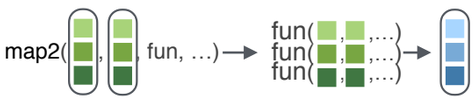
Застосування функцій з purrr : : ШПАРГАЛКА

Застосування функцій

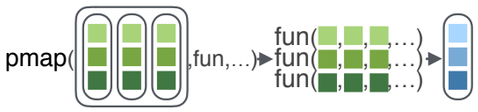
Map-функції ітеративно застосовують функції до кожного елементу списку або вектора.



map(.x, .f, …)
Застосовує функцію до кожного елемента списку або вектора.
map(x, is.logical)



map2(.x, ,y, .f, …)
Застосовує функцію до пар елементів з двох списків, векторів.
map2(x, y, sum)



pmap(.l, .f, …) Застосовує функцію до груп елементів зі списку списків, векторів.
pmap(list(x, y, z), sum, na.rm = TRUE)

invoke_map(.f, .x = list(NULL), …, .env=NULL)
Обчислює кожну функцію зі списку.
Також **invoke**. *l <- list(var, sd); invoke_map(l, x = 1:9)*

lmap(.x, .f, ...) Застосовує .f до спискових елементів списку, вектора.
imap(.x, .f, ...) Застосовує .f до елементів списку, вектора та його індексу.

РЕЗУЛЬТАТ

map(), map2(), pmap(), imap та **invoke_map** повертають список. Використовуйте версії з суфіксом для отримання результату певного типу, наприклад **map2_chr**, **pmap_lgl**, і т.д.

Використовуйте **walk**, **walk2** та **pwalk** для здійснення побічних дій. Усі невидимо повертають вхідні дані.

функция	результат
map	список
map_chr	символьный вектор
map_dbl	числовий вектор
map_dfc	data frame (з'єд. по стовпч.)
map_dfr	data frame (з'єд. по рядкам)
map_int	цілочисельний вектор
map_lgl	логічний вектор
walk	здійснює побічні дії, невидимо повертає вхідні дані

СКОРОЧЕННЯ - всередині функцій purrr:

"name" стає **function(x) x[["name"]]**, напр. *map(l, "a")* витягує *a* з кожного елемента *l*

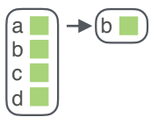
~ . стає **function(x) x**, напр. *map(l, ~ 2 +.)* стає *map(l, function(x) 2 + x)*

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

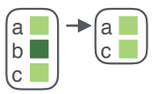
~ ..1 ..2 etc стає **function(..1, ..2,** etc**) ..1 ..2** etc напр. *pmap(list(a, b, c), ~ ..3 + ..1 - ..2)* стає *pmap(list(a, b, c), function(a, b, c) c + a - b)*

Робота зі списками

ФІЛЬТРУВАННЯ СПИСКІВ

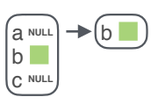


pluck(.x, ..., .default=NULL) Вибирає елемент за ім'ям або індексом, *pluck(x,"b")*, або його атрибут (з **attr_getter**).
pluck(x,"b",attr_getter("n"))

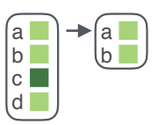


keep(.x, .p, …) Вибирає ел-ти, задовол. логіч. умові.
keep(x, is.na)

discard(.x, .p, …) Вибирає ел-ти, не задовол. логіч. умови. *discard(x, is.na)*

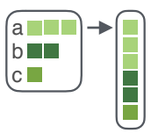


compact(.x, .p = identity) Видаляє пусті ел-ти.
compact(x)

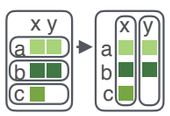


head_while(.x, .p, …) Повертає початкові ел-ти до першого, що не задовол. умові. Також **tail_while**.
head_while(x, is.character)

ФОРМАТУВАННЯ СПИСКІВ



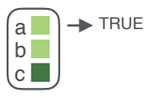
flatten(.x) Видаляє рівень індексів зі списку. Також **flatten_chr**, **flatten_dbl**, **flatten_dfc**, **flatten_dfr**, **flatten_int**, **flatten_lgl**.
flatten(x)



transpose(.l, .names = NULL) Транспонує порядок індексів у багаторівневому списку.
transpose(x)

ПІДСУМОВУВАННЯ СПИСКІВ

every(.x, .p, …) Чи усі ел-ти задовол. умові?
every(x, is.character)

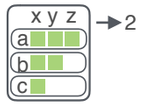


some(.x, .p, …) Чи є ел-ти, що задовол. умові?
some(x, is.character)

has_element(.x, .y) Чи міститься ел-т у списку?
has_element(x, "foo")

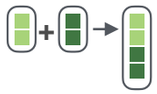
detect(.x, .f, ..., .right=FALSE, .p) Знаходить перший відповідний ел-т.
detect(x, is.character)

detect_index(.x, .f, ..., .right = FALSE, .p) Знаходить індекс першого відповідного ел-та.
detect_index(x, is.character)

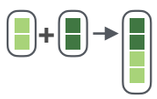


vec_depth(x) Повертає глибину (кількість рівнів індексів).
vec_depth(x)

З'ЄДНАННЯ СПИСКІВ



append(x, values, after = length(x)) Додає ел-т у кінець списку.
append(x, list(d = 1))

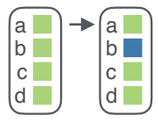


prepend(x, values, before = 1) Додає ел-т у початок списку.
prepend(x, list(d = 1))

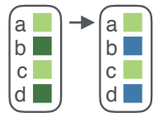
splice(…) Поєднує об'єкти в список, зберігаючи S3 об'єкти як підсписки. *splice(x, y, "foo")*

ПЕРЕТВОРЕННЯ СПИСКІВ

modify(.x, .f, ...) Застосовує функцію до кожного ел-та. Також **map**, **map_chr**, **map_dbl**, **map_dfc**, **map_dfr**, **map_int**, **map_lgl**.
modify(x, ~.+ 2)



modify_at(.x, .at, .f, ...) Застосовує функцію до ел-тів за ім'ям або індексом. Також **map_at**.
modify_at(x, "b", ~.+ 2)



modify_if(.x, .p, .f, ...) Застосовує функцію до ел-тів, що задовол. певній умові. Також **map_if**.
modify_if(x, is.numeric,~.+2)

modify_depth(.x,.depth,.f,...) Застосовує функцію до кожного ел-та списку на певному рівні.
modify_depth(x, 1, ~.+ 2)

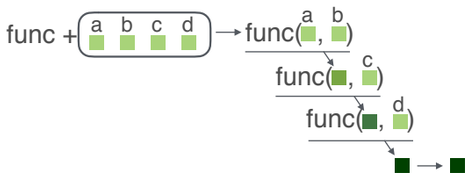
РОБОТА ЗІ СПИСКАМИ

array_tree(array, margin = NULL) Перетворює масив у список. Також **array_branch**.
array_tree(x, margin = 3)

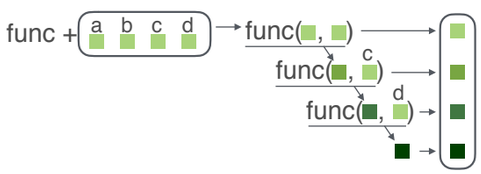
cross2(.x, .y, .filter = NULL) Усі комбінації .x та .y. Також **cross**, **cross3**, **cross_df**. *cross2(1:3, 4:6)*

set_names(x, nm = x) Встановлює імена вектора / списку безпосередньо або за допомогою функції.
set_names(x, c("p", "q", "r"))
set_names(x, tolower)

Згортка списків



reduce(.x, .f, ..., .init) Застосовує функцію рекурсивно до кожного ел-та списку, вектора. Також **reduce_right**, **reduce2**, **reduce2_right**.
reduce(x, sum)



accumulate(.x, .f, ..., .init) Як reduce(), але повертає проміжні результати. Також **accumulate_right**.
accumulate(x, sum)

Модифікація функцій

compose() Створює композицію функцій.

lift() Змінює тип вхідних даних функції. Також **lift_dl**, **lift_dv**, **lift_ld**, **lift_lv**, **lift_vd**, **lift_vl**.

rerun() Незалежно виконує вираз n разів.

negate() Створює заперечення предикатної функції (можна використовувати в конвеєрі!)

partial() Змінює функцію, задаючи певні параметри.

safely() Змінює функцію, щоб повертала список рез-тів та помилок.

quietly() Змінює функцію, щоб повертала список рез-тів, виведення, повідомлень і попереджень.

possibly() Змінює функцію, щоб повертала значення за замовчуванням при появі помилки (замість помилки).

Вкладені дані

Вкладений data frame зберігає окремі таблиці всередині клітинок більшої упорядковуючої таблиці

вкладений data frame

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

n_iris

вміст "клітинок"

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]]

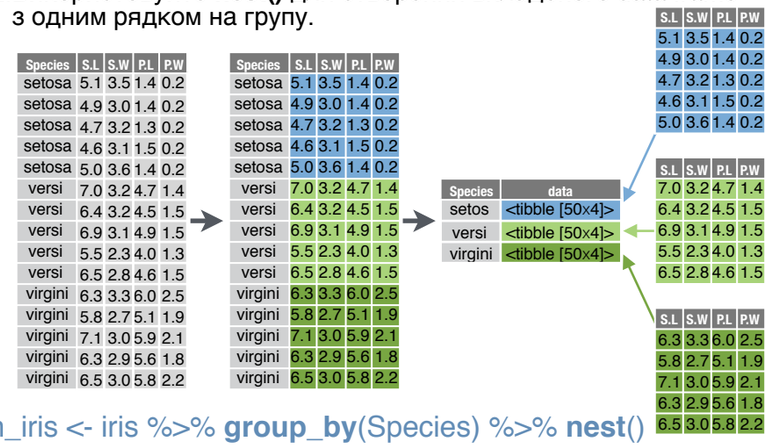
Використовуйте вкладений data frame для:

- збереження відносин між спостереженнями та підмножинами даних
- маніпулювання відразу декількома підтаблицями з **purrr** функциями **map()**, **map2()** або **pmap()**.

Використовуйте двоетапний процес для створення вкладеного data frame:

1.Згрупуйте data frame у групи за допомогою **dplyr::group_by()**

2.Використовуйте **nest()** для створення вкладеного data frame з одним рядком на групу.



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

tidyr::**nest(**data, ..., .key = data**)**
Згротає групи в клітинки у вигляді data frame (для згруп. даних).

Розгортайте вкладений data frame за допомогою **unnest()**:

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

n_iris %>% **unnest**()

tidyr::**unnest(**data, ..., .drop = NA, .id=NULL, .sep=NULL**)**
Розгортає вкладений data frame.

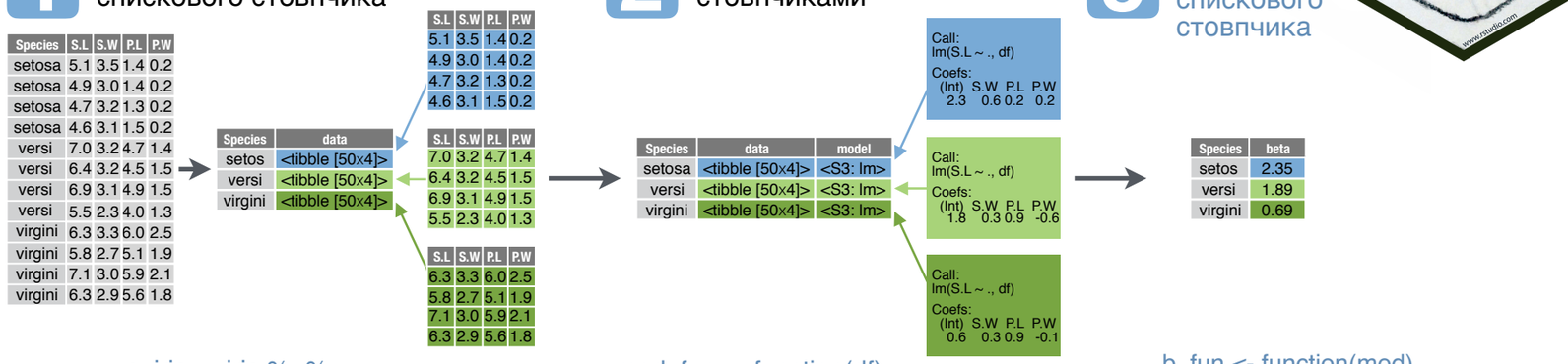
Процес роботи зі списковими стовпчиками

Вкладений data frame використовує **списковий стовпчик**, тобто список, який зберігається як стовпчик в data frame. Типовий **робочий процес** зі списковими стовпчиками:

1 **Створення** спискового стовпчика

2 **Робота** зі списковими стовпчиками

3 **Спрощення** спискового стовпчика




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

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

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

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

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

1. СТВОРЕННЯ СПИСКОВОГО СТОВПЧИКА - створюйте за допом. функцій з пакетів **tibble** та **dplyr**, а також **tidyr**::nest()

tibble::**tribble(**…**)**
Створює списковий стовпчик за необхідністю

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(**…**)**
Зберігає список як списковий стовпчик.
tibble(max = c(3, 4, 5), seq = list(1:3, 1:4, 1:5))

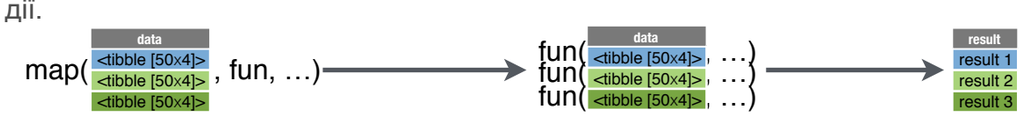
tibble::**enframe(**x, name="name", value="value"**)**
Конвертує багаторівневий список у tibble зі списковими стовпчиками.
enframe(list('3'=1:3, '4'=1:4, '5'=1:5), 'max', 'seq')

dplyr::**mutate(**.data, …**)** Також **transmute()**
Повертає списковий стовпчик, коли результат - список.
mtcars %>% **mutate**(seq = **map**(cyl, seq))

dplyr::**summarise(**.data, …**)**
Повертає списковий стовпчик, коли результат створений з **list()**.
mtcars %>% group_by(cyl) %>% **summarise**(q = **list**(quantile(mpg)))

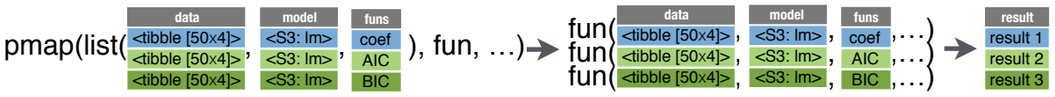
2. РОБОТА ЗІ СПИСКОВИМИ СТОВПЧИКАМИ - Використовуйте функції purrr **map()**, **map2()** та **pmap()** для поелементного застосування функції з поверненням результату в списковий стовпчик. **walk()**, **walk2()** та **pwalk()** працюють аналогічно, але здійснюють побічні дії.

purrr::**map(**.x, .f, ...**)**
Застосовує .f поелементно до .x як .f(.x)
n_iris %>% **mutate**(n = **map**(data, dim))



purrr::**map2(**.x, .y, .f, ...**)**
Застосовує .f поелементно до .x та .y як .f(.x, .y)
m_iris %>% **mutate**(n = **map2**(data, model, list))

purrr::**pmap(**.l, .f, ...**)**
Застосовує .f поелементно до векторів з .l
m_iris %>%
mutate(n = **pmap**(list(data, model, data), list))



3. СПРОЩЕННЯ СПИСКОВОГО СТОВПЧИКА (до звичайного стовпчика)

Використовуйте функції purrr **map_lgl()**, **map_int()**, **map_dbl()**, **map_chr()**, а також tidyr::**unnest()** для перетворення спискового стовпчика на звичайний.

purrr::**map_lgl(**.x, .f, ...**)** Застосовує .f поелементно до .x, повертає логічний вектор
n_iris %>% **transmute**(n = **map_lgl**(data, is.matrix))

purrr::**map_int(**.x, .f, ...**)** Застосовує .f поелементно до .x, повертає цілочис. вектор
n_iris %>% **transmute**(n = **map_int**(data, nrow))

purrr::**map_dbl(**.x, .f, ...**)** Застосовує .f поелементно до .x, повертає числовий вектор
n_iris %>% **transmute**(n = **map_dbl**(data, nrow))

purrr::**map_chr(**.x, .f, ...**)** Застосовує .f поелементно до .x, повертає симв. вектор
n_iris %>% **transmute**(n = **map_chr**(data, nrow))