Трансформація даних з dplyr : : ШПАРГАЛКА

Функції **dplyr** працюють з конвеєрами (pipes) і припускають **охайні** (tidy) **дані**. У охайних даних:

Кожна **змінна - стовпчик**

&

Кожне **спостереження - рядок**

pipes

x %>% f(y)
стає **f(x, y)**

Підсумовування спостережень

Способи застосування підсумовуючих функцій до стовпчиків для створення нової таблиці. Підсум. функції приймають на вхід вектори і повертають одне значення (див. зворот).

підсумовуюча функція

summarise(.data, …)
Обчислює таблицю зведених значень. Також **summarise_**().
summarise(mtcars, avg = mean(mpg))

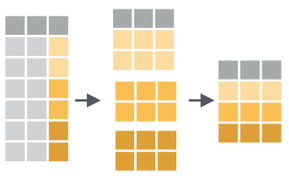
count(x, ..., wt = NULL, sort = FALSE)
Підраховує кількість рядків у групах, заданих змінними у ...
Також **tally**().
count(iris, Species)

ВАРІАНТИ

summarise_all() - Застосовує функції до всіх стовпчиків.
summarise_at() - Застосовує функції до деяких стовпчиків.
summarise_if() - Застосовує функції до стовп. одного типу.

Групування спостережень

Використовуйте **group_by()** для створення "згрупованої" копії таблиці. Функції dplyr оперують окремо кожної "групою" і потім поєднують результати.



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

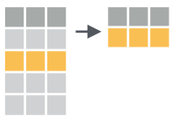
group_by(.data, ..., add = FALSE**)**
Повертає копію таблиці, згруповану по ...
g_iris <- group_by(iris, Species)

ungroup(x, …**)**
Повертає разгруповану копію таблиці
ungroup(g_iris)

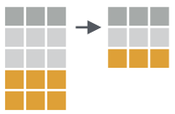
Обробка спостережень

ВИТЯГ СПОСТЕРЕЖЕНЬ

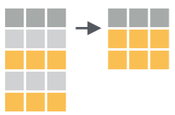
Рядкові функції повертають підмножину рядків як нову таблицю. Використовуйте варіант з _ на кінці для узгодження з нестандартним обчисленням.



filter(.data, …**)** Витягує рядки, які відповідають логічним критеріям. Також **filter_()**.
filter(iris, Sepal.Length > 7)

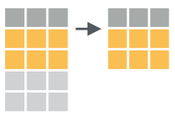


distinct(.data, ..., .keep_all = FALSE**)**
Видаляє рядки зі значеннями, що дублюються. Також **distinct_()**.
distinct(iris, Species)



sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = parent.frame()**)**
Випадково вибирає певну частку рядків.
sample_frac(iris, 0.5, replace = TRUE)

sample_n(tbl, size, replace = FALSE, weight = NULL, .env = parent.frame()**)** Випадково вибирає певну кількість рядків.
sample_n(iris, 10, replace = TRUE)



slice(.data, …**)** Вибирає рядки за позицією. Також **slice_()**.
slice(iris, 10:15)

top_n(x, n, wt**)** Вибирає та сортує топ n рядків (за групами для згрупованих даних).
top_n(iris, 5, Sepal.Width)

Логічні та булеві оператори для filter()

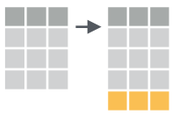
<	<=	is.na()	%in%	|	xor()
>	>=	!is.na()	!	&	

Див. **?base::logic** та **?Comparison** для допомоги.

УПОРЯДКУВАННЯ СПОСТЕРЕЖЕНЬ

arrange(.data, …**)**
Сортує рядки за значеннями стовпців (від меншого до більшого), с **desc ()** - від більшого до меншого.
arrange(mtcars, mpg)
arrange(mtcars, desc(mpg))

ДОДАВАННЯ СПОСТЕРЕЖЕНЬ



add_row(.data, ..., .before = NULL, .after = NULL**)**
Додає один або кілька рядків до таблиці.
add_row(faithful, eruptions = 1, waiting = 1)

Обробка змінних

ВИТЯГ ЗМІННИХ

Стовпчикові функції повертають набір стовпчиків як нову таблицю. Використовуйте варіант з _ на кінці для узгодження з нестандартним обчисленням.

select(.data, …**)** Витягує стовпчики за назвою. Також **select_if()**
select(iris, Sepal.Length, Species)

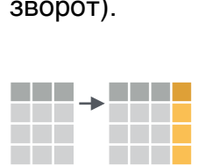
Використовуйте ці допом. функції з select (),
наприклад select(iris, starts_with("Sepal"))

contains(match**)**
ends_with(match**)**
matches(match**)**
num_range(prefix, range**)**
one_of(…**)**
starts_with(match**)**
:, e.g. mpg:cyl
-, e.g, -Species

СТВОРЕННЯ НОВИХ ЗМІННИХ

Способи застосування векторизованих функцій до стовпчиків. Векторизовані функції приймають на вхід вектори і повертають вектори такої ж довжини (див. зворот).

векторизована функція



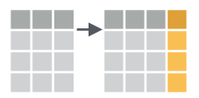
mutate(.data, …**)**
Обчислює новий (нові) стовпчик(и).
mutate(mtcars, gpm = 1/mpg)

transmute(.data, …**)**
Обчислює новий (нові) стовпчик(и), усуває інші.
transmute(mtcars, gpm = 1/mpg)

mutate_all(.tbl, .funs, …**)** Застосовує функції до всіх стовпчиків. Використовуйте з **funs()**.
mutate_all(faithful, funs(log(.), log2(.)))

mutate_at(.tbl, .cols, .funs, …**)** Застосовує функції до деяких стовпчиків. Використовуйте з **funs()**, **vars()** та іншими допоміжними функціями для select().
mutate_at(iris, vars(-Species), funs(log(.)))

mutate_if(.tbl, .predicate, .funs, …**)**
Застосовує функції до стовпчиків одного типу. Використовуйте з **funs()**.
mutate_if(iris, is.numeric, funs(log(.)))



add_column(.data, ..., .before = NULL, .after = NULL**)** Додає новий (нові) стовпчик(и).
add_column(mtcars, new = 1:32)

rename(.data, …**)** Перейменовує стовпчики.
rename(iris, Length = Sepal.Length)

Векторизовані функції

ДЛЯ ВИКОРИСТАННЯ З MUTATE ()

mutate() та **transmute()** застосовують векторизовані функції до стовпчиків для створення нових стовпчиків. Векторизовані функції приймають на вхід вектори і повертають вектори такої ж довжини.

векторизована функція

ЗМІЩЕННЯ

dplyr::**lag()** - Зміщує елементи на 1
dplyr::**lead()** - Зміщує елементи на -1

КУМУЛЯТИВНЕ АГРЕГУВАННЯ

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

РАНЖУВАННЯ

dplyr::**cume_dist()** - Частка елементів <=
dplyr::**dense_rank()** - Ранг з нічиїми = min, без пропусків
dplyr::**min_rank()** - Ранг з нічиїми = min
dplyr::**ntile()** - Розподіляє по n коміркам
dplyr::**percent_rank()** - min_rank, нормований до [0,1]
dplyr::**row_number()** - Ранг з нічиїми = "перший елемент"

МАТЕМАТИЧНІ ФУНКЦІЇ

+, - , *, /, ^, %/%, %% - Арифметичні операції
log(), log2(), log10() - Логарифми
<, <=, >, >=, !=, == - Логічні порівняння

РІЗНЕ

dplyr::**between()** - x >= left & x <= right
dplyr::**case_when()** - Множинне if_else()
dplyr::**coalesce()** - Вибирає поелементно перше не-NA значення серед набору векторів
dplyr::**if_else()** - Поелементне if() + else()
dplyr::**na_if()** - Замінює деякі значення на NA
pmax() - Поелементний max()
pmin() - Поелементний min()
dplyr::**recode()** - Векторизований switch()
dplyr::**recode_factor()** - Векторизований switch() для факторів

Підсумовуючі функції

ДЛЯ ВИКОРИСТАННЯ З SUMMARISE ()

summarise() застосовує підсумовуючі функції до стовпців для створення нової таблиці. Підсумовуючі функції приймають на вхід вектори і повертають одне значення.

підсумовуюча функція

ПІДРАХУНОК

dplyr::**n()** - Кількість значень/рядків
dplyr::**n_distinct()** - Кількість унікальних
sum(!is.na()) - Кількість не-NA

ЦЕНТР

mean() - Середнє, також **mean(!is.na())**
median() - Медіана

ЛОГІЧНЕ

mean() - Частка значень TRUE
sum() - Кількість TRUE

ПОЛОЖЕННЯ/ПОРЯДОК

dplyr::**first()** - Перше значення
dplyr::last() - Останнє значення
dplyr::nth() - Значення на n-му місці у векторі

РАНГ

quantile() - n-й квантиль
min() - Мінімальне значення
max() - Максимальне значення

РОЗКИД

IQR() - Міжквартільний розмах
mad() - Медіанне абс. відхилення
sd() - Стандартне відхилення
var() - Дисперсія

Імена рядків

Охайні дані не використовують імена рядків поза стовпців. Для роботи з іменами рядків перемістіть їх в стовпець.

rownames_to_column()
Імена рядків => стовпчик.
a <- rownames_to_column(iris, var = "C")

column_to_rownames()
Стовпчик => імена рядків.
column_to_rownames(a, var = "C")

Також **has_rownames()**, **remove_rownames()**

Комбінування таблиць

КОМБІНУВАННЯ ЗМІННИХ

Використовуйте **bind_cols()** для з'єднання стовпців таблиць без змін.

bind_cols(…) Повертає таблиці, одна поруч з іншою, як одну таблицю. ПЕРЕКОНАЙТЕСЯ У ВІДПОВІДНОСТІ СТРОК.

Використовуйте **"Змінюючий JOIN"** для з'єднання таблиці зі стовпцями з іншої таблиці, поєднуючи значення з їх рядків. Кожен JOIN зберігає різні комбінації значень.

left_join(x, y, by = NULL, copy=FALSE, suffix=c(".x",".y"), …**)**
З'єднує відпов. зн-ня із y в x.

right_join(x, y, by = NULL, copy = FALSE, suffix=c(".x",".y")…**)**
З'єднує відпов. зн-ня із x в y.

inner_join(x, y, by = NULL, copy = FALSE, suffix=c(".x",".y"),…**)**
З'єднує дані. Зберігає тільки відпов. рядки.

full_join(x, y, by = NULL, copy=FALSE, suffix=c(".x",".y"), …**)**
З'єднує дані. Зберігає усі значення, усі рядки.

by = c("col1", "col2") задає стовпчик(и) для з'єднання.
left_join(x, y, by = "A")

Іменований вектор у **by = c("col1" = "col2")** задає стовпчики для з'єднання з різними іменами.
left_join(x, y, by = c("C" = "D"))

suffix задає суфікси для імен стовпчиків, що дублюються.
left_join(x, y, by = c("C" = "D"), suffix = c("1", "2"))

КОМБІНУВАННЯ СПОСТЕРЕЖЕНЬ

Використовуйте **bind_rows()** для з'єднання таблиць одна під іншою без змін.

bind_rows(…, .id = NULL**)**
Повертає таблиці одна над іншою як одну таблицю. Задавайте в .id ім'я стовпчика з початковими іменами таблиць, щоб його додати (як на малюнку).

intersect(x, y, …)
Рядки із x та із y.

setdiff(x, y, …)
Рядки із x, але не із y.

union(x, y, …)
Рядки із x або y.
(Повторювані видалені).
union_all() їх зберігає.

Використовуйте **setequal()** для перевірки, чи містять дві таблиці однаковий набір рядків (у будь-якому порядку).

ВИТЯГ РЯДКІВ

Використовуйте **"Фільтруючий JOIN"** для фільтрації однієї таблиці з використанням рядків іншої.

semi_join(x, y, by = NULL, …**)**
Повертає рядки з x, у яких є відповідність в y. КОРИСНО БАЧИТИ, ЩО ЗАЛИШИТЬСЯ.

anti_join(x, y, by = NULL, …**)**
Повертає рядки з x, у яких немає відповідності в y. КОРИСНО БАЧИТИ, ЩО НЕ ЗАЛИШИТЬСЯ.