Трансформация данных с dplyr : : ШПАРГАЛКА

Функции **dplyr** работают с конвейерами (pipes) и предполагают **опрятные** (tidy) **данные**. В опрятных данных:

Каждая **переменная** - **столбец**

&

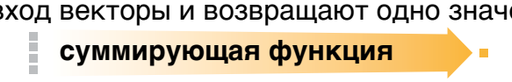
Каждое **наблюдение** - **строка**

pipes

x %>% f(y) становится **f(x, y)**

Суммирование наблюдений

Способы применения **суммирующих функций** к столбцам для создания новой таблицы. Суммирующие функции принимают на вход векторы и возвращают одно значение (см. оборот).



summarise(.data, …)
Вычисляет таблицу сводных значений. Также **summarise_**().
summarise(mtcars, avg = mean(mpg))

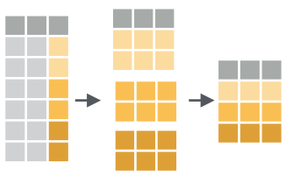
count(x, ..., wt = NULL, sort = FALSE)
Подсчитывает количество строк в группах, заданных переменными в ... Также **tally**().
count(iris, Species)

ВАРИАНТЫ

summarise_all() - Применяет функции ко всем столбцам.
summarise_at() - Применяет функции к некоторым столбцам.
summarise_if() - Применяет функции к стб. одного типа.

Группировка наблюдений

Используйте **group_by()** для создания "сгруппированной" копии таблицы. Функции dplyr оперируют отдельно каждой "группой" и потом совмещают результаты.



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

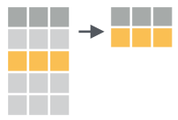
group_by(.data, ..., add = FALSE**)**
Возвращает копию таблицы, сгрупп. по ...
g_iris <- group_by(iris, Species)

ungroup(x, …**)**
Возвращает разгруппированную копию таблцы
ungroup(g_iris)

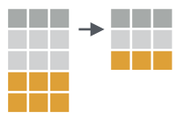
Обработка наблюдений

ИЗВЛЕЧЕНИЕ НАБЛЮДЕНИЙ

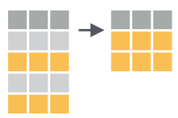
Строковые функции возвращают подмножество строк как новую таблицу. Используйте вариант с _ на конце для согласования с нестандартным вычислением.



filter(.data, …**)** Извлекает строки на основании логических критериев. Также **filter_()**.
filter(iris, Sepal.Length > 7)

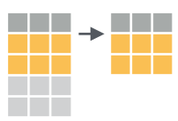


distinct(.data, ..., .keep_all = FALSE**)**
Удаляет строки с дублирующимися значениями. Также **distinct_()**.
distinct(iris, Species)



sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = parent.frame()**)**
Случайно выбирает опр. долю строк.
sample_frac(iris, 0.5, replace = TRUE)

sample_n(tbl, size, replace = FALSE, weight = NULL, .env = parent.frame()**)** Случайно выбирает опр. кол-во строк.
sample_n(iris, 10, replace = TRUE)



slice(.data, …**)** Выбирает строки по позиции. Также **slice_()**.
slice(iris, 10:15)

top_n(x, n, wt**)** Выбирает и сортирует топ n строк (по группам для сгрупп. данных).
top_n(iris, 5, Sepal.Width)

Логические и булевые операторы для filter()

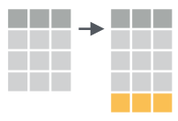
<	<=	is.na()	%in%	|	xor()
>	>=	!is.na()	!	&	

См. **?base::logic** и **?Comparison** для помощи.

УПОРЯДОЧИВАНИЕ НАБЛЮДЕНИЙ

arrange(.data, …**)**
Сортирует строки по значениям столбцов (от меньшего к большему), с **desc()** - от большего к меньшему.
arrange(mtcars, mpg)
arrange(mtcars, desc(mpg))

ДОБАВЛЕНИЕ НАБЛЮДЕНИЙ



add_row(.data, ..., .before = NULL, .after = NULL**)**
Добавляет одну или несколько строк к таблице.
add_row(faithful, eruptions = 1, waiting = 1)

Обработка переменных

ИЗВЛЕЧЕНИЕ ПЕРЕМЕННЫХ

Столбцовые функции возвращают множество столбцов как новую таблицу. Используйте вариант с _ на конце для согласования с нестандартным вычислением.

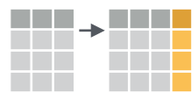
select(.data, …**)** Извлекает столбцы по имени. Также **select_if()**
select(iris, Sepal.Length, Species)

Используйте эти вспом. функции с select (),
например 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

СОЗДАНИЕ НОВЫХ ПЕРЕМЕННЫХ

Способы применения векторизованных функций к столбцам. Векторизованные функции принимают на вход векторы и возвращают векторы такой же длины (см. оборот).



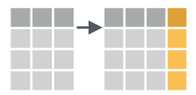
mutate(.data, …**)**
Вычисляет новый(е) столбец(цы).
mutate(mtcars, gpm = 1/mpg)

transmute(.data, …**)**
Вычисляет новый(е) столбец(цы), убирает остальные.
transmute(mtcars, gpm = 1/mpg)

mutate_all(.tbl, .funs, …**)** Применяет функции ко всем столбцам. Используйте с **funs()**.
mutate_all(faithful, funs(log(.), log2(.)))

mutate_at(.tbl, .cols, .funs, …**)** Применяет функции к некоторым столбцам. Используйте с **funs()**, **vars()** и другими вспомогательными функциями для select().
mutate_at(iris, vars(-Species), funs(log(.)))

mutate_if(.tbl, .predicate, .funs, …**)**
Применяет функции ко всем столбцам одного типа. Используйте с **funs()**.
mutate_if(iris, is.numeric, funs(log(.)))



add_column(.data, ..., .before = NULL, .after = NULL**)** Добавляет новый(е) столбец(цы).
add_column(mtcars, new = 1:32)

rename(.data, …**)** Переименовывает столбцы.
rename(iris, Length = Sepal.Length)

Векторизованные функции

ДЛЯ ИСПОЛЬЗОВАНИЯ С MUTATE ()

mutate() и **transmute()** применяют векторизованные функции к столбцам для создания новых столбцов.
Векторизованные функции принимают на вход векторы и возвращают векторы такой же длины.

векторизованная функция

СМЕЩЕНИЯ

dplyr::**lag()** - Смещает элементы на 1
dplyr::**lead()** - Смещает элементы на -1

КУМУЛЯТИВНОЕ АГРЕГИРОВАНИЕ

dplyr::**cumall()** - Кумулятивное all()
dplyr::**cumany()** - Кумулятивное any()
cummax() - Кумулятивный max()
dplyr::**cummean()** - Кумулятивное mean()
cummin() - Кумулятивный min()
cumprod() - Кумулятивное prod()
cumsum() - Кумулятивная sum()

РАНЖИРОВАНИЕ

dplyr::**cume_dist()** - Доля элементов <=
dplyr::**dense_rank()** - Ранг с ничьими = min, без пробелов
dplyr::**min_rank()** - Ранг с ничьими = min
dplyr::**ntile()** - Распределяет по n ячейкам
dplyr::**percent_rank()** - min_rank, нормированный к [0,1]
dplyr::**row_number()** - Ранг с ничьими = "первый элемент"

МАТЕМАТИЧЕСКИЕ ФУНКЦИИ

+, - , *, /, ^, %/%, %% - Арифметические операции
log(), log2(), log10() - Логарифмы
<, <=, >, >=, !=, == - Логические сравнения

РАЗНОЕ

dplyr::**between()** - x >= left & x <= right
dplyr::**case_when()** - Множественное if_else()
dplyr::**coalesce()** - Выбирает поэлементно первое не-NA значение среди набора векторов
dplyr::**if_else()** - Поэлементное if() + else()
dplyr::**na_if()** - Заменяет некоторые значения на NA
pmax() - Поэлементный max()
pmin() - Поэлементный min()
dplyr::**recode()** - Векторизованный switch()
dplyr::**recode_factor()** - Векторизованный switch() для факторов

Суммирующие функции

ДЛЯ ИСПОЛЬЗОВАНИЯ С SUMMARISE ()

summarise() применяет суммирующие функции к столбцам для создания новой таблицы. Суммирующие функции принимают на вход векторы и возвращают одно значение.

суммирующая функция

ПОДСЧЕТ

dplyr::**n()** - Количество значений/строк
dplyr::**n_distinct()** - Кол-во уникальных
sum(!is.na()) - Кол-во не-NA

ЦЕНТР

mean() - Среднее, также **mean(!is.na())**
median() - Медиана

ЛОГИЧЕСКОЕ

mean() - Доля значений TRUE
sum() - Кол-во TRUE

ПОЛОЖЕНИЕ/ПОРЯДОК

dplyr::**first()** - Первое значение
dplyr::last() - Последнее значение
dplyr::nth() - Значение на n-м месте в векторе

РАНГ

quantile() - n-й квантиль
min() - Минимальное значение
max() - Максимальное значение

РАЗБРОС

IQR() - Межквартильный размах
mad() - Медианное абс. отклонение
sd() - Стандартное отклонение
var() - Дисперсия

Имена строк

Опрятные данные не используют имена строк вне столбцов. Для работы с именами строк переместите их в столбец.

rownames_to_column()
Имена строк => столбец.
a <- rownames_to_column(iris, var = "C")

column_to_rownames()
Столбец => имена строк.
column_to_rownames(a, var = "C")

Также **has_rownames()**, **remove_rownames()**

Комбинирование таблиц

КОМБИНИРОВАНИЕ ПЕРЕМЕННЫХ

x + y =

Используйте **bind_cols()** для соединения столбцов таблиц без изменений.

bind_cols(…) Возвращает таблицы, помещенные друг рядом с другом, как одну таблицу.
УБЕДИТЕСЬ В СООТВЕТСТВИИ СТРОК.

Используйте "**Изменяющий JOIN**" для соединения таблицы со столбцами из другой таблицы, сочетая значения из их строк. Каждый JOIN сохраняет разные комбинации значений.

left_join(x, y, by = NULL, copy=FALSE, suffix=c(".x",".y"), …**)**
Соединяет соотв. зн-ия из y в x.

right_join(x, y, by = NULL, copy = FALSE, suffix=c(".x",".y"),…**)**
Соединяет соотв. зн-ия из x в y.

inner_join(x, y, by = NULL, copy = FALSE, suffix=c(".x",".y"),…**)**
Соединяет данные. Сохраняет только соотв. строки.

full_join(x, y, by = NULL, copy=FALSE, suffix=c(".x",".y"), …**)**
Соединяет данные. Сохраняет все значения, все строки.

by = c("col1", "col2") задает столбец(цы) для соединения.
left_join(x, y, by = "A")

Именованный вектор в **by = c("col1" = "col2")** задает столбцы для соединения с разными именами.
left_join(x, y, by = c("C" = "D"))

suffix задает суффиксы для дублирующихся имен столбцов.
left_join(x, y, by = c("C" = "D"), suffix = c("1", "2"))

КОМБИНИРОВАНИЕ НАБЛЮДЕНИЙ

x + y =

Используйте **bind_rows()** для соединения таблиц одна под другой без изменений.

bind_rows(…, .id = NULL**)**
Возвращает таблицы одна над другой как одну таблицу. Задавайте в .id имя столбца с исходными именами таблиц для его добавления (как на рисунке).

intersect(x, y, …)
Строки из x и из y.

setdiff(x, y, …)
Строки из x, но не из y.

union(x, y, …)
Строки из x или y. (Дублирующиеся удалены). union_all() их сохраняет.

Используйте **setequal()** для проверки, содержат ли две таблицы одинаковый набор строк (в любом порядке).

ИЗВЛЕЧЕНИЕ СТРОК

x + y =

Используйте "**Фильтрующий JOIN**" для фильтрации одной таблицы с использованием строк другой.

semi_join(x, y, by = NULL, …**)**
Возвращает строки из x, у которых есть соответствие в y. ПОЛЕЗНО ВИДЕТЬ, ЧТО ОСТАНЕТСЯ.

anti_join(x, y, by = NULL, …**)**
Возвращает строки из x, у которых нет соответствия в y. ПОЛЕЗНО ВИДЕТЬ, ЧТО НЕ ОСТАНЕТСЯ.