1. Библиотеки / данные

Разбиение набора данных на группы и применение некоторой функции к каждой группе, будь то в целях агрегирования или преобразования, зачастую является одной из важнейших операций анализа данных. После загрузки, слияния и подготовки набора данных обычно вычисляется статистика по группам. В библиотеке pandas имеется гибкий и быстрый механизм GroupBy, который позволяет формировать продольные и поперечные срезы, а также агрегировать наборы данных естественным образом.

Одна из причин популярности реляционных баз данных и языка SQL – простота соединения, фильтрации, преобразования и агрегирования данных. Однако в том, что касается групповых операций, языки запросов типа SQL несколько ограничены. Как мы увидим, выразительность и мощь языка Python и библиотеки pandas позволяют выполнять гораздо более сложные групповые операции с помощью функций, принимающих произвольный объект pandas или массив NumPy.

2. Механизм GroupBy

Хедли Уикхем, автор многих популярных пакетов на языке программирования R, предложил для групповых операций термин «разделение-применение-объединение» («Split-Apply-Combine»), который, как мне кажется, удачно описывает процесс.

На первом этапе данные, хранящиеся в объекте pandas, будь то Series или DataFrame, разделяются на группы по одному или нескольким указанным вами ключам. Разделение производится вдоль одной оси объекта, например, объект DataFrame можно группировать по строкам или по столбцам. Затем к каждой группе применяется некоторая функция, которая порождает новое значение. Наконец, результаты применения всех функций объединяются в результирующий объект. Форма результирующего объекта обычно зависит от того, что именно проделывается с данными.

Чтобы вычислить среднее по группе, есть несколько способов. Первый из них - это вызвать метод groupby(). Переменная groups - это объект типа DataFrameGroupBy. Пока что он не вычислил ничего, кроме промежуточных данных о групповом ключе. Идея в том, что этот объект хранит всю информацию, необходимую для последующего применения некоторой операции к каждой группе.

Также groups имеет много полезных свойств. Свойство ngroups извлекает количество групп, которые будут сформированы в итоге.

Метод size() возвращает объект Series, содержащий размеры групп. Каждая группа состоит из двух строк.

Свойство groups возвращает «питоновский» словарь, ключи которого будут представлять имена каждой группы, а значение в словаре - это массив индексных меток, принадлежащих каждой соответствующей группе.

С помощью метода get\_group() можно извлечь любую конкретную группу.

Метод nth() возвращает строку в каждой группе.

Объект DataFrameGroupBy поддерживает итерирование, в результате которого генерируется последовательность кортежей, содержащих имя группы и блок данных.

Мы познакомились с этапом «разделение», создали промежуточный объект groups. Как все-таки вычислить среднее по группам? Очень просто – вызвать метод mean(). В результате создан новый объект Series, индексированный уникальными значениями в столбце x со значениями, равными среднему по каждой группе.

3. Типы группировок

Информацию о группировке можно передать различными способами. Рассмотрим их.

**3.1 по столбцам**

Самый простой способ, с которым мы уже встречались, - это передать методу groupby() список меток столбцов.

Для анализа полученных значений средних очень полезным будет дополнительно использовать метод unstack(). Такое представление информация удобнее для анализа.

**3.2 по уровням индекса**

Иерархически индексированные наборы данных можно агрегировать по одному из уровней индекса оси. Для этого нужно передать номер или имя уровня в параметр level. Также можно передать названия нескольких уровней. Важно отметить, что мы не указывали столбец, по которому хотим посчитать среднее. По умолчанию выводится среднее по всем числовым столбцам в разрезе групп.

**3.3 с использованием функции**

Использование функций Python - более абстрактный способ определения соответствия групп по сравнению с группировкой по значениям в столбцах или в индексе. Функция, переданная в качестве группового ключа, будет вызвана по одному разу для каждого значения в индексе, а возвращённые ею значения будут именами групп.

Пусть требуется сгруппировать по длине значений меток индекса уровня day. Для этого просто передаём функцию len() в метод groupby().

**3.4 с использованием массива**

Функция choice() из модуля random библиотеки NumPy позволяет нам случайным образом выбрать значение из заданного списка.

**3.5 бонус**

До сих пор мы рассматривали примеры группировки значений построчно, однако метод groupby() также позволяет группировать столбцы.

Напомню, что свойство dtypes возвращает объект Series с информацией о типе данных для каждого столбца. Эту информацию можно использовать в качестве ключей группировки, при этом, так как группировка производится по оси столбцов, в методе groupby() указываем параметр axis=1. Метод size() покажет количество элементов в каждой группе.

**4. Агрегирование групп**

Под агрегированием обычно понимают любое преобразование данных, которое порождает скалярное значение из массива. В примерах выше мы встречали несколько типов таких преобразований: вычисление среднего, количества значений.

Для использования собственных функций агрегирования можно передать функцию в метод aggregate() или кратко - agg().

Агрегирование - лишь одна из групповых операций. Это частный случай более общего класса преобразований, в котором применяемая функция преобразует одномерный массив в скалярное значение.

**5. Преобразование групп**

В этом разделе мы познакомимся с методами transform() и apply(), которые позволяют выполнять групповые операции других видов.

**5.1 метод transform()**

Основным отличием метода transform() от метода aggregate() является этап COMBINED - объединение результатов применения функций к группам.

Как и aggregate(), метод transform() - более специализированная функция, предъявляющая жесткие требования. Передаваемая ему функция должна возвращать либо скалярное значение. либо преобразованный массив такого же размера, что и исходный.

**5.2 метод apply**

Самым общим из методов класса GroupBy является метод apply().

Функция top() вызывается для каждой части объекта DataFrame, после чего результаты склеиваются методом pd.concat(), а частям сопоставляются метки, совпадающие с именами групп, поэтому результат имеет иерархический индекс. Внутренний уровень содержит индексные значения из исходного объекта DataFrame, а внешний - название групп.

**6. Исключение групп**

С помощью метода filter() возможно выборочно удалить группы данных из обработки. В этот метод передают функцию, с помощью которой по каждой группе можно принять решение о включении данной группы в итоговый результат. Если функция возвращает значение True, то группа включается в результат. Если функция возвращает значение False, то группа исключается.

**7. Сводная таблица**

Сводная таблица - это средство обобщения данных, применяемое в электронных таблицах и других аналитических программах. Она агрегирует таблицы по одному или нескольким ключам и строит другую таблицу, в которой одни групповые ключи расположены в строках, а другие - в столбцах.

Библиотека pandas позволяет строить сводные таблицы с помощью описанного выше механизма GroupBy в сочетании с операциями stack() и unstack(). В классе DataFrame также имеется метод pivot\_table().

Эту информацию можно было бы дополнить, включив в частичные итоги, для чего следует задать параметр margins = True. Для применения другой функции агрегирования её нужно передать в параметр aggfunc.