



РАНХиГС

РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА
И ГОСУДАРСТВЕННОЙ СЛУЖБЫ
ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ



РАНХиГС
экономический
факультет



Агрегирование и слияние данных

проф. кафедры Эконометрики и математической экономики ЭФ
д.т.н. Шилин Кирилл Юрьевич

РАНХиГС каб. 419/3

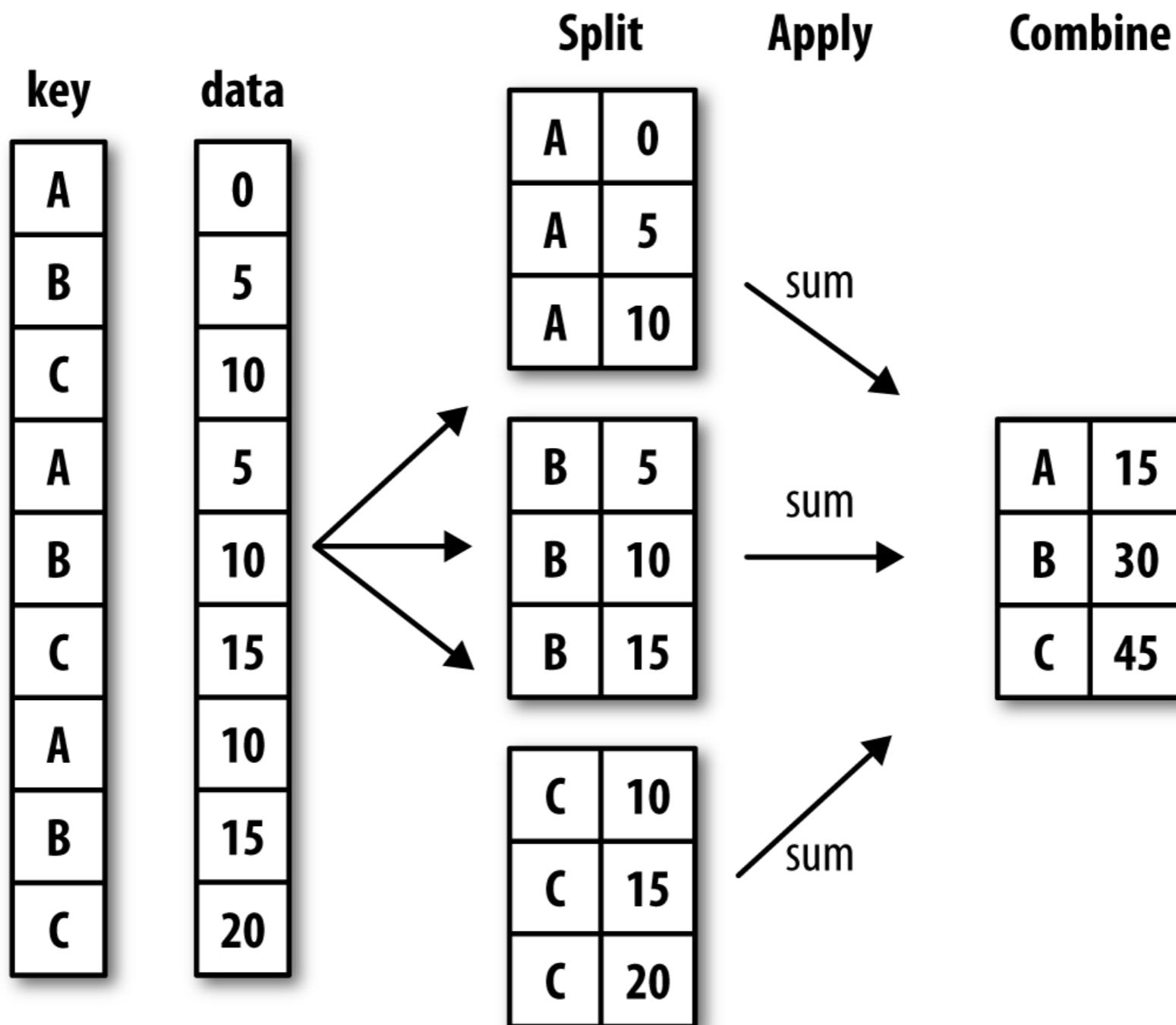
email: kshilin@ranepa.ru



Группировка и агрегирование

1. Данные, хранящиеся в объекте pandas разделяются на группы по одному или нескольким ключам. Разделение производится вдоль одной оси объекта.
2. К каждой группе применяется некоторая функция, которая порождает новое значение.
3. Результаты применения всех функций объединяются в результирующий объект. Форма результирующего объекта обычно зависит от того, что именно проделывается с данными.

Задача разделить и подсчитать



Исходные данные

```
In [10]: df = pd.DataFrame({'key1' : ['a', 'a', 'b', 'b', 'a'],
....: 'key2' : ['one', 'two', 'one', 'two', 'one'],
....: 'data1' : np.random.randn(5),
....: 'data2' : np.random.randn(5)})
```

```
In [11]: df
```

```
Out[11]:
```

	data1	data2	key1	key2
0	-0.204708	1.393406	a	one
1	0.478943	0.092908	a	two
2	-0.519439	0.281746	b	one
3	-0.555730	0.769023	b	two
4	1.965781	1.246435	a	one

Решение по одному ключу

```
In [12]: grouped = df['data1'].groupby(df['key1'])
```

```
In [13]: grouped
```

```
Out[13]: <pandas.core.groupby.SeriesGroupBy object at 0x7faa31537390>
```

```
In [14]: grouped.mean()
```

```
Out[14]:
```

```
key1
```

```
 a    0.746672
```

```
 b   -0.537585
```

```
Name: data1, dtype: float64
```

Сократим код

```
In [21]: df.groupby('key1').mean()
```

```
Out[21]:
```

	data1	data2
key1		
a	0.746672	0.910916
b	-0.537585	0.525384

Решение по двум ключам

```
In [15]: means = df['data1'].groupby([df['key1'], df['key2']]).mean()
```

```
In [16]: means
```

```
Out[16]:
```

```
key1  key2
a    one      0.880536
      two      0.478943
b    one     -0.519439
      two     -0.555730
```

```
Name: data1, dtype: float64
```

```
In [17]: means.unstack()
```

```
Out[17]:
```

```
key2      one      two
key1
a    0.880536  0.478943
b   -0.519439 -0.555730
```

Сократим код

```
In [22]: df.groupby(['key1', 'key2']).mean()
```

```
Out[22]:
```

		data1	data2
key1	key2		
a	one	0.880536	1.319920
	two	0.478943	0.092908
b	one	-0.519439	0.281746
	two	-0.555730	0.769023

Продолжим с заменой ключей



```
In [18]: states = np.array(['Ohio', 'California', 'California', 'Ohio', 'Ohio'])
```

['a', 'a', 'b', 'b', 'a']



```
In [19]: years = np.array([2005, 2005, 2006, 2005, 2006])
```

['one', 'two', 'one', 'two', 'one']



```
In [20]: df['data1'].groupby([states, years]).mean()
```

Out[20]:

```
California 2005    0.478943
           2006   -0.519439
Ohio       2005   -0.380219
           2006    1.965781
```

```
Name: data1, dtype: float64
```

Заглянем внутрь

```
In [24]: for name, group in df.groupby('key1'):
....:     print(name)
....:     print(group)
```

```
a
  data1      data2 key1 key2
0 -0.204708  1.393406    a  one
1  0.478943  0.092908    a  two
4  1.965781  1.246435    a  one
b
  data1      data2 key1 key2
2 -0.519439  0.281746    b  one
3 -0.555730  0.769023    b  two
```

```
In [25]: for (k1, k2), group in df.groupby(['key1', 'key2']):
....:     print((k1, k2))
....:     print(group)
```

```
('a', 'one')
  data1      data2 key1 key2
0 -0.204708  1.393406    a  one
4  1.965781  1.246435    a  one
('a', 'two')
  data1      data2 key1 key2
1  0.478943  0.092908    a  two
('b', 'one')
  data1      data2 key1 key2
2 -0.519439  0.281746    b  one
('b', 'two')
  data1      data2 key1 key2
3 -0.555730  0.769023    b  two
```

```
In [26]: pieces = dict(list(df.groupby('key1')))
```

```
In [27]: pieces['b']
Out[27]:
  data1      data2 key1 key2
2 -0.519439  0.281746    b  one
3 -0.555730  0.769023    b  two
```

Повеселимся с индексами

```
In [35]: people = pd.DataFrame(np.random.randn(5, 5),
.....:                               columns=['a', 'b', 'c', 'd', 'e'],
.....:                               index=['Joe', 'Steve', 'Wes', 'Jim', 'Travis'])
```

```
In [36]: people.iloc[2:3, [1, 2]] = np.nan # Add a few NA values
```

```
In [37]: people
```

```
Out[37]:
```

	a	b	c	d	e
Joe	1.007189	-1.296221	0.274992	0.228913	1.352917
Steve	0.886429	-2.001637	-0.371843	1.669025	-0.438570
Wes	-0.539741	NaN	NaN	-1.021228	-0.577087
Jim	0.124121	0.302614	0.523772	0.000940	1.343810
Travis	-0.713544	-0.831154	-2.370232	-1.860761	-0.860757

```
In [38]: mapping = {'a': 'red', 'b': 'red', 'c': 'blue',
.....:                 'd': 'blue', 'e': 'red', 'f': 'orange'}
```

```
In [39]: by_column = people.groupby(mapping, axis=1)
```

```
In [40]: by_column.sum()
```

```
Out[40]:
```

	blue	red
Joe	0.503905	1.063885
Steve	1.297183	-1.553778
Wes	-1.021228	-1.116829
Jim	0.524712	1.770545
Travis	-4.230992	-2.405455

```
In [41]: map_series = pd.Series(mapping)
```

```
In [42]: map_series
```

```
Out[42]:
```

	color
a	red
b	red
c	blue
d	blue
e	red
f	orange

```
dtype: object
```

```
In [43]: people.groupby(map_series, axis=1).count()
```

```
Out[43]:
```

	blue	red
Joe	2	3
Steve	2	3
Wes	1	2
Jim	2	3
Travis	2	3

Сломаем мозг окончательно

In [37]: people

Out[37]:

	a	b	c	d	e
Joe	1.007189	-1.296221	0.274992	0.228913	1.352917
Steve	0.886429	-2.001637	-0.371843	1.669025	-0.438570
Wes	-0.539741	NaN	NaN	-1.021228	-0.577087
Jim	0.124121	0.302614	0.523772	0.000940	1.343810
Travis	-0.713544	-0.831154	-2.370232	-1.860761	-0.860757

по длине Индекса
(количество букв в имени)



In [44]: people.groupby(len).sum()

Out[44]:

	a	b	c	d	e
3	0.591569	-0.993608	0.798764	-0.791374	2.119639
5	0.886429	-2.001637	-0.371843	1.669025	-0.438570
6	-0.713544	-0.831154	-2.370232	-1.860761	-0.860757

In [45]: key_list = ['one', 'one', 'one', 'two', 'two']

In [46]: people.groupby([len, key_list]).min()

Out[46]:

	a	b	c	d	e	
3	one	-0.539741	-1.296221	0.274992	-1.021228	-0.577087
	two	0.124121	0.302614	0.523772	0.000940	1.343810
5	one	0.886429	-2.001637	-0.371843	1.669025	-0.438570
6	two	-0.713544	-0.831154	-2.370232	-1.860761	-0.860757

по длине Индекса
и добавочному ключу



Агрегирование при иерархическом индексировании

```
In [47]: columns = pd.MultiIndex.from_arrays([['US', 'US', 'US', 'JP', 'JP'],
                                             [1, 3, 5, 1, 3]],
                                             names=['cty', 'tenor'])
```

```
In [48]: hier_df = pd.DataFrame(np.random.randn(4, 5), columns=columns)
```

```
In [49]: hier_df
```

```
Out[49]:
```

	cty	US		JP	
tenor	1	3	5	1	3
0	0.560145	-1.265934	0.119827	-1.063512	0.332883
1	-2.359419	-0.199543	-1.541996	-0.970736	-1.307030
2	0.286350	0.377984	-0.753887	0.331286	1.349742
3	0.069877	0.246674	-0.011862	1.004812	1.327195

```
In [51]: df
```

```
Out[51]:
```

	data1	data2	key1	key2
0	-0.204708	1.393406	a	one
1	0.478943	0.092908	a	two
2	-0.519439	0.281746	b	one
3	-0.555730	0.769023	b	two
4	1.965781	1.246435	a	one

```
In [50]: hier_df.groupby(level='cty', axis=1).count()
```

```
Out[50]:
```

	cty	JP	US
0	2	3	
1	2	3	
2	2	3	
3	2	3	

```
In [52]: grouped = df.groupby('key1')
```

```
In [53]: grouped['data1'].quantile(0.9)
```

```
Out[53]:
```

key1	
a	1.668413
b	-0.523068

```
Name: data1, dtype: float64
```



Команды агрегирования

Имя функции	Описание
count	Количество отличных от NA значений в группе
sum	Сумма отличных от NA значений
mean	Среднее отличных от NA значений
median	Медиана отличных от NA значений
std, vag	Несмещенное (со знаменателем $n - 1$) стандартное отклонение и дисперсия
min, max	Минимальное и максимальное отличные от NA значения
prod	Произведение отличных от NA значений
first, last	Первое и последнее отличные от NA значения

Также можно использовать любую функцию определенную на сгруппированном объекте или создать собственную

Существующий агрегатор

```
In [51]: df
Out[51]:
      data1      data2 key1 key2
0 -0.204708  1.393406    a  one
1  0.478943  0.092908    a  two
2 -0.519439  0.281746    b  one
3 -0.555730  0.769023    b  two
4  1.965781  1.246435    a  one
```

```
In [52]: grouped = df.groupby('key1')
```

```
In [53]: grouped['data1'].quantile(0.9)
Out[53]:
key1
a    1.668413
b   -0.523068
Name: data1, dtype: float64
```

Свой агрегатор

```
In [54]: def peak_to_peak(agg):
....:     return agg.max() - agg.min()

In [55]: grouped.agg(peak_to_peak)
Out[55]:
      data1    data2
key1
a    2.170488  1.300498
b    0.036292  0.487276
```

```
In [51]: df
Out[51]:
      data1    data2 key1 key2
0   -0.204708  1.393406  a  one
1    0.478943  0.092908  a  two
2   -0.519439  0.281746  b  one
3   -0.555730  0.769023  b  two
4    1.965781  1.246435  a  one
```

Недокументированные агрегаторы (работают вопреки)

```
In [56]: grouped.describe()
Out[56]:
      data1
      count      mean       std      min      25%      50%      75%
key1
a    3.0  0.746672  1.109736 -0.204708  0.137118  0.478943  1.222362
b    2.0 -0.537585  0.025662 -0.555730 -0.546657 -0.537585 -0.528512
      data2
      max  count      mean       std      min      25%      50%
key1
a    1.965781  3.0  0.910916  0.712217  0.092908  0.669671  1.246435
b   -0.519439  2.0  0.525384  0.344556  0.281746  0.403565  0.525384
      75%      max
key1
a    1.319920  1.393406
b    0.647203  0.769023
```