



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

Merge



Соединение данных из двух разных таблиц

Соединение выполняется по столбцам или индексам.

При объединении столбцов в столбцы индексы DataFrame игнорируются.

При объединении по индексам или индексам и столбцам индекс будет сохранен.

Merge - 1

```
In [35]: df1 = pd.DataFrame({'key': ['b', 'b', 'a', 'c', 'a', 'a', 'b'],  
....:  
.....: 'data1': range(7)})
```

```
In [36]: df2 = pd.DataFrame({'key': ['a', 'b', 'd'],  
....:  
.....: 'data2': range(3)})
```

```
In [37]: df1
```

```
Out[37]:
```

```
data1 key  
0    0   b  
1    1   b  
2    2   a  
3    3   c  
4    4   a  
5    5   a  
6    6   b
```

```
In [38]: df2
```

```
Out[38]:
```

```
data2 key  
0    0   a  
1    1   b  
2    2   d
```

```
In [39]: pd.merge(df1, df2)
```

```
Out[39]:
```

	data1	key	data2
0	0	b	1
1	1	b	1
2	6	b	1
3	2	a	0
4	4	a	0
5	5	a	0

```
In [40]: pd.merge(df1, df2, on='key')
```

```
Out[40]:
```

	data1	key	data2
0	0	b	1
1	1	b	1
2	6	b	1
3	2	a	0
4	4	a	0
5	5	a	0

Merge - 2

```
In [41]: df3 = pd.DataFrame({'lkey': ['b', 'b', 'a', 'c', 'a', 'a', 'b'],
....:                      'data1': range(7)})
```

```
In [42]: df4 = pd.DataFrame({'rkey': ['a', 'b', 'd'],
....:                      'data2': range(3)})
```

```
In [43]: pd.merge(df3, df4, left_on='lkey', right_on='rkey')
```

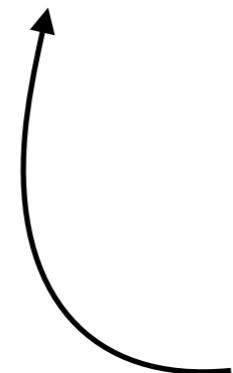
Out[43]:

	data1	lkey	data2	rkey
0	0	b	1	b
1	1	b	1	b
2	6	b	1	b
3	2	a	0	a
4	4	a	0	a
5	5	a	0	a

```
In [44]: pd.merge(df1, df2, how='outer')
```

Out[44]:

	data1	key	data2
0	0.0	b	1.0
1	1.0	b	1.0
2	6.0	b	1.0
3	2.0	a	0.0
4	4.0	a	0.0
5	5.0	a	0.0
6	3.0	c	NaN
7	NaN	d	2.0



Значение	Поведение
'inner'	Брать только комбинации ключей, встречающиеся в обеих таблицах
'left'	Брать все ключи, встречающиеся в левой таблице
'right'	Брать все ключи, встречающиеся в правой таблице
'outer'	Брать все комбинации ключей

Merge - 3

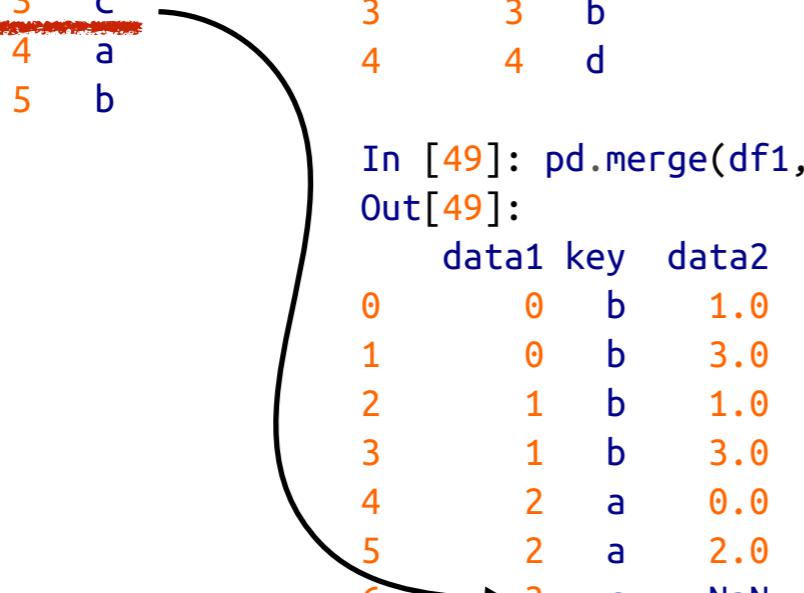
```
In [45]: df1 = pd.DataFrame({'key': ['b', 'b', 'a', 'c', 'a', 'b'],
....:                      'data1': range(6)})
```

```
In [46]: df2 = pd.DataFrame({'key': ['a', 'b', 'a', 'b', 'd'],
....:                      'data2': range(5)})
```

```
In [47]: df1
Out[47]:
   data1  key
0      0    b
1      1    b
2      2    a
3      3    c
4      4    a
5      5    b
```

```
In [48]: df2
Out[48]:
   data2  key
0      0    a
1      1    b
2      2    a
3      3    b
4      4    d
```

Diagram illustrating the merge operation:



```
In [49]: pd.merge(df1, df2, on='key', how='left')
Out[49]:
   data1  key  data2
0      0    b    1.0
1      0    b    3.0
2      1    b    1.0
3      1    b    3.0
4      2    a    0.0
5      2    a    2.0
6      3    c    NaN
7      4    a    0.0
8      4    a    2.0
9      5    b    1.0
10     5    b    3.0
```

```
In [50]: pd.merge(df1, df2, how='inner')
Out[50]:
   data1  key  data2
0      0    b    1
1      0    b    3
2      1    b    1
3      1    b    3
4      5    b    1
5      5    b    3
6      2    a    0
7      2    a    2
8      4    a    0
9      4    a    2
```



Merge - 4

```
In [51]: left = pd.DataFrame({'key1': ['foo', 'foo', 'bar'],
.....:                      'key2': ['one', 'two', 'one'],
.....:                      'lval': [1, 2, 3]})
```

```
In [52]: right = pd.DataFrame({'key1': ['foo', 'foo', 'bar', 'bar'],
.....:                      'key2': ['one', 'one', 'one', 'two'],
.....:                      'rval': [4, 5, 6, 7]})
```

```
In [53]: pd.merge(left, right, on=['key1', 'key2'], how='outer')
```

Out[53]:

	key1	key2	lval	rval
0	foo	one	1.0	4.0
1	foo	one	1.0	5.0
2	foo	two	2.0	NaN
3	bar	one	3.0	6.0
4	bar	two	NaN	7.0

Merge - 5

```
In [54]: pd.merge(left, right, on='key1')
```

```
Out[54]:
```

	key1	key2_x	lval	key2_y	rval
0	foo	one	1	one	4
1	foo	one	1	one	5
2	foo	two	2	one	4
3	foo	two	2	one	5
4	bar	one	3	one	6
5	bar	one	3	two	7

```
In [55]: pd.merge(left, right, on='key1', suffixes=('_left', '_right'))
```

```
Out[55]:
```

	key1	key2_left	lval	key2_right	rval
0	foo	one	1	one	4
1	foo	one	1	one	5
2	foo	two	2	one	4
3	foo	two	2	one	5
4	bar	one	3	one	6
5	bar	one	3	two	7

Merge опции

Аргумент	Описание
left	Объект DataFrame в левой части операции слияния
right	Объект DataFrame в правой части операции слияния
how	Допустимые значения: 'inner', 'outer', 'left', 'right'
on	Имена столбцов, по которым производится соединение. Должны присутствовать в обоих объектах DataFrame. Если не заданы и не указаны никакие другие ключи соединения, то используются имена столбцов, общих для обоих объектов
left_on	Столбцы левого DataFrame, используемые как ключи соединения
right_on	Столбцы правого DataFrame, используемые как ключи соединения
left_index	Использовать индекс строк левого DataFrame в качестве его ключа соединения (или нескольких ключей в случае мультииндекса)
right_index	То же, что left_index, но для правого DataFrame
sort	Сортировать слитые данные лексикографически по ключам соединения; по умолчанию True. Иногда при работе с большими наборами данных лучше отключить
suffixes	Кортеж строк, которые дописываются в конец совпадающих имен столбцов; по умолчанию ('_x', '_y'). Например, если в обоих объектах DataFrame встречается столбец 'data', то в результирующем объекте появятся столбцы 'data_x' и 'data_y'
copy	Если равен False, то в некоторых особых случаях разрешается не копировать данные в результирующую структуру. По умолчанию данные копируются всегда
indicator	Добавляет специальный столбец _merge, который сообщает об источнике каждой строки; он может принимать значения 'left_only', 'right_only' или 'both' в зависимости от того, как строка попала в результат соединения

Merge индексов

```
In [56]: left1 = pd.DataFrame({'key': ['a', 'b', 'a', 'a', 'b', 'c'],
....: 'value': range(6)})
```

```
In [57]: right1 = pd.DataFrame({'group_val': [3.5, 7]}, index=['a', 'b'])
```

```
In [58]: left1
```

```
Out[58]:
```

```
   key  value
0   a      0
1   b      1
2   a      2
3   a      3
4   b      4
5   c      5
```

```
In [59]: right1
```

```
Out[59]:
```

```
   group_val
a      3.5
b      7.0
```

```
In [60]: pd.merge(left1, right1, left_on='key', right_index=True)
Out[60]:
```

```
   key  value  group_val
0   a      0      3.5
2   a      2      3.5
3   a      3      3.5
1   b      1      7.0
4   b      4      7.0
```

```
In [61]: pd.merge(left1, right1, left_on='key', right_index=True, how='outer')
Out[61]:
```

```
   key  value  group_val
0   a      0      3.5
2   a      2      3.5
3   a      3      3.5
1   b      1      7.0
4   b      4      7.0
5   c      5      NaN
```

Соединение нескольких таблиц по осям

Соединение выполняется вдоль определенной оси

Можно добавить условия для соединения по дополнительным осям.

Можно добавить еще один уровень иерархической индексации, что может быть полезно при условии совпадения меток..

Concatenate (NumPy)

```
In [79]: arr = np.arange(12).reshape((3, 4))
```

```
In [80]: arr
```

```
Out[80]:
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

```
In [81]: np.concatenate([arr, arr], axis=1)
```

```
Out[81]:
```

```
array([[ 0,  1,  2,  3,  0,  1,  2,  3],
       [ 4,  5,  6,  7,  4,  5,  6,  7],
       [ 8,  9, 10, 11,  8,  9, 10, 11]])
```

Concat (Pandas)

```
In [82]: s1 = pd.Series([0, 1], index=['a', 'b'])
```

```
In [83]: s2 = pd.Series([2, 3, 4], index=['c', 'd', 'e'])
```

```
In [84]: s3 = pd.Series([5, 6], index=['f', 'g'])
```

```
In [85]: pd.concat([s1, s2, s3])
```

```
Out[85]:
```

```
a    0  
b    1  
c    2  
d    3  
e    4  
f    5  
g    6  
dtype: int64
```

```
In [86]: pd.concat([s1, s2, s3], axis=1)
```

```
Out[86]:
```

```
      0    1    2  
a  0.0  NaN  NaN  
b  1.0  NaN  NaN  
c  NaN  2.0  NaN  
d  NaN  3.0  NaN  
e  NaN  4.0  NaN  
f  NaN  NaN  5.0  
g  NaN  NaN  6.0
```

```
In [87]: s4 = pd.concat([s1, s3])
```

```
In [88]: s4
```

```
Out[88]:
```

```
a    0  
b    1  
f    5  
g    6  
dtype: int64
```

```
In [89]: pd.concat([s1, s4], axis=1)
```

```
Out[89]:
```

```
      0    1  
a  0.0  0  
b  1.0  1  
f  NaN  5  
g  NaN  6
```

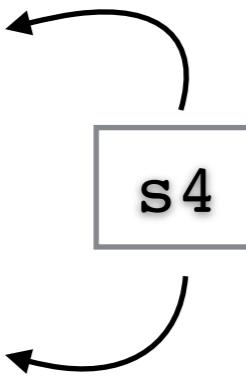
```
In [90]: pd.concat([s1, s4], axis=1, join='inner')
```

```
Out[90]:
```

```
      0    1  
a  0.0  0  
b  1.0  1
```

Concat -2

s1	a	0
	b	1
	c	2
s2	d	3
	e	4
s3	f	5
	g	6



A diagram illustrating the concatenation process. Three separate tables labeled s1, s2, and s3 are shown on the left. An arrow points from each of these three tables to a single large square box labeled 's4' on the right, representing the result of concatenating them.

```
In [91]: pd.concat([s1, s4], axis=1, join_axes=[['a', 'c', 'b', 'e']])
```

```
Out[91]:
```

	0	1
a	0.0	0.0
c	NaN	NaN
b	1.0	1.0
e	NaN	NaN

```
In [92]: result = pd.concat([s1, s1, s3], keys=['one', 'two', 'three'])
```

```
In [93]: result
```

```
Out[93]:
```

one	a	0
	b	1
two	a	0
	b	1
three	f	5
	g	6

dtype: int64

```
In [94]: result.unstack()
```

```
Out[94]:
```

	a	b	f	g
one	0.0	1.0	NaN	NaN
two	0.0	1.0	NaN	NaN
three	NaN	NaN	5.0	6.0



Concat - 3

s1	a	0
	b	1
	c	2
s2	d	3
	e	4
s3	f	5
	g	6

The diagram illustrates the concatenation process. Three separate DataFrames, labeled s1, s2, and s3, are shown on the left. An arrow points from each of these three DataFrames to a larger, central box labeled s4, representing the resulting concatenated DataFrame.

In [95]: `pd.concat([s1, s2, s3], axis=1, keys=['one', 'two', 'three'])`
Out[95]:

	one	two	three
a	0.0	NaN	NaN
b	1.0	NaN	NaN
c	NaN	2.0	NaN
d	NaN	3.0	NaN
e	NaN	4.0	NaN
f	NaN	NaN	5.0
g	NaN	NaN	6.0

Concat - 4

```
In [96]: df1 = pd.DataFrame(np.arange(6).reshape(3, 2), index=['a', 'b', 'c'],
....:                      columns=['one', 'two'])
```

```
In [97]: df2 = pd.DataFrame(5 + np.arange(4).reshape(2, 2), index=['a', 'c'],
....:                      columns=['three', 'four'])
```

```
In [98]: df1
```

```
Out[98]:
```

	one	two
a	0	1
b	2	3
c	4	5

```
In [99]: df2
```

```
Out[99]:
```

	three	four
a	5	6
c	7	8

```
In [100]: pd.concat([df1, df2], axis=1, keys=['level1', 'level2'])
```

```
Out[100]:
```

	level1	level2		
	one	two	three	four
a	0	1	5.0	6.0
b	2	3	NaN	NaN
c	4	5	7.0	8.0

```
In [101]: pd.concat({'level1': df1, 'level2': df2}, axis=1)
```

```
Out[101]:
```

	level1	level2		
	one	two	three	four
a	0	1	5.0	6.0
b	2	3	NaN	NaN
c	4	5	7.0	8.0

```
In [102]: pd.concat([df1, df2], axis=1, keys=['level1', 'level2'],
....:                  names=['upper', 'lower'])
```

```
Out[102]:
```

	upper	level1	level2		
	lower	one	two	three	four
a		0	1	5.0	6.0
b		2	3	NaN	NaN
c		4	5	7.0	8.0



Concat - 5

```
In [103]: df1 = pd.DataFrame(np.random.randn(3, 4), columns=['a', 'b', 'c', 'd'])
```

```
In [104]: df2 = pd.DataFrame(np.random.randn(2, 3), columns=['b', 'd', 'a'])
```

```
In [105]: df1
```

```
Out[105]:
```

	a	b	c	d
0	1.246435	1.007189	-1.296221	0.274992
1	0.228913	1.352917	0.886429	-2.001637
2	-0.371843	1.669025	-0.438570	-0.539741

```
In [106]: df2
```

```
Out[106]:
```

	b	d	a
0	0.476985	3.248944	-1.021228
1	-0.577087	0.124121	0.302614

```
In [107]: pd.concat([df1, df2], ignore_index=True)
```

```
Out[107]:
```

	a	b	c	d
0	1.246435	1.007189	-1.296221	0.274992
1	0.228913	1.352917	0.886429	-2.001637
2	-0.371843	1.669025	-0.438570	-0.539741
3	-1.021228	0.476985	NaN	3.248944
4	0.302614	-0.577087	NaN	0.124121

Concat опции

Аргумент	Описание
<code>objs</code>	Список или словарь конкатенируемых объектов pandas. Единственный обязательный аргумент
<code>axis</code>	Ось, вдоль которой производится конкатенация, по умолчанию 0
<code>join</code>	Допустимые значения: 'inner', 'outer', по умолчанию 'outer'; следует ли пересекать (inner) или объединять (outer) индексы вдоль других осей
<code>join_axes</code>	Какие конкретно индексы использовать для других $n - 1$ осей вместо выполнения пересечения или объединения
<code>keys</code>	Значения, которые ассоциируются с конкатенируемыми объектами и образуют иерархический индекс вдоль оси конкатенации. Может быть список или массив произвольных значений, а также массив кортежей или список массивов (если в параметре <code>levels</code> передаются массивы для нескольких уровней)
<code>levels</code>	Конкретные индексы, которые используются на одном или нескольких уровнях иерархического индекса, если задан параметр <code>keys</code>
<code>names</code>	Имена создаваемых уровней иерархического индекса, если заданы параметры <code>keys</code> и (или) <code>levels</code>
<code>verify_integrity</code>	Проверить новую ось в конкатенированном объекте на наличие дубликатов и, если они имеются, возбудить исключение. По умолчанию <code>False</code> – дубликаты разрешены
<code>ignore_index</code>	Не сохранять индексы вдоль оси конкатенации, а вместо этого создать новый индекс <code>range(total_length)</code>



Присоединение столбцов друг к другу

Соединение по индексу или ключевому столбцу



```
In [68]: left2 = pd.DataFrame([[1., 2.], [3., 4.], [5., 6.]],
....:                           index=['a', 'c', 'e'],
....:                           columns=['Ohio', 'Nevada'])

In [69]: right2 = pd.DataFrame([[7., 8.], [9., 10.], [11., 12.], [13., 14.]],
....:                            index=['b', 'c', 'd', 'e'],
....:                            columns=['Missouri', 'Alabama'])

In [70]: left2
Out[70]:
      Ohio  Nevada
a    1.0    2.0
c    3.0    4.0
e    5.0    6.0

In [71]: right2
Out[71]:
      Missouri  Alabama
b        7.0     8.0
c        9.0    10.0
d       11.0    12.0
e       13.0    14.0
```

```
In [72]: pd.merge(left2, right2, how='outer', left_index=True, right_index=True)
Out[72]:
      Ohio  Nevada  Missouri  Alabama
a    1.0    2.0      NaN      NaN
b    NaN    NaN      7.0     8.0
c    3.0    4.0      9.0    10.0
d    NaN    NaN     11.0    12.0
e    5.0    6.0     13.0    14.0
```

```
In [73]: left2.join(right2, how='outer')
Out[73]:
      Ohio  Nevada  Missouri  Alabama
a    1.0    2.0      NaN      NaN
b    NaN    NaN      NaN      7.0
c    3.0    4.0      9.0     10.0
d    NaN    NaN      NaN    11.0
e    5.0    6.0     13.0    14.0
```



Combine_first



Сращивание перекрывающихся данных

Заполняет отсутствующие в одном объекте данные значениями из другого объекта

Combine_first

```
In [115]: df1 = pd.DataFrame({'a': [1., np.nan, 5., np.nan],  
.....:  
.....:  
.....: 'b': [np.nan, 2., np.nan, 6.],  
.....:  
.....: 'c': range(2, 18, 4)})
```

```
In [116]: df2 = pd.DataFrame({'a': [5., 4., np.nan, 3., 7.],  
.....:  
.....: 'b': [np.nan, 3., 4., 6., 8.]})
```

```
In [117]: df1  
Out[117]:
```

	a	b	c
0	1.0	NaN	2
1	NaN	2.0	6
2	5.0	NaN	10
3	NaN	6.0	14

```
In [118]: df2  
Out[118]:
```

	a	b
0	5.0	NaN
1	4.0	3.0
2	NaN	4.0
3	3.0	6.0
4	7.0	8.0

```
In [119]: df1.combine_first(df2)  
Out[119]:
```

	a	b	c
0	1.0	NaN	2.0
1	4.0	2.0	6.0
2	5.0	4.0	10.0
3	3.0	6.0	14.0
4	7.0	8.0	NaN

«Расшить» и «сшить»

```
In [120]: data = pd.DataFrame(np.arange(6).reshape((2, 3)),  
.....: index=pd.Index(['Ohio', 'Colorado'], name='state'),  
.....: columns=pd.Index(['one', 'two', 'three'],  
.....: name='number'))
```

```
In [121]: data  
Out[121]:  
number      one    two    three  
state  
Ohio          0      1      2  
Colorado     3      4      5
```

```
In [122]: result = data.stack()  
  
In [123]: result  
Out[123]:  
state      number  
Ohio        one        0  
             two        1  
                 three     2  
Colorado    one        3  
             two        4  
                 three     5  
dtype: int64
```

```
In [124]: result.unstack()  
Out[124]:  
number      one    two    three  
state  
Ohio          0      1      2  
Colorado     3      4      5
```

```
In [125]: result.unstack(0)
Out[125]:
state    Ohio    Colorado
number
one        0        3
two        1        4
three      2        5
```

```
In [126]: result.unstack('state')
Out[126]:
state    Ohio    Colorado
number
one        0        3
two        1        4
three      2        5
```

«Расширить» и «сшить»

```
In [127]: s1 = pd.Series([0, 1, 2, 3], index=['a', 'b', 'c', 'd'])
```

```
In [128]: s2 = pd.Series([4, 5, 6], index=['c', 'd', 'e'])
```

```
In [129]: data2 = pd.concat([s1, s2], keys=['one', 'two'])
```

```
In [130]: data2
```

```
Out[130]:
```

```
one   a    0  
      b    1  
      c    2  
      d    3  
two   c    4  
      d    5  
      e    6
```

```
dtype: int64
```

```
In [131]: data2.unstack()
```

```
Out[131]:
```

```
      a    b    c    d    e  
one  0.0  1.0  2.0  3.0  NaN  
two  NaN  NaN  4.0  5.0  6.0
```

```
In [133]: data2.unstack().stack()
```

```
Out[133]:
```

```
one   a    0.0  
      b    1.0  
      c    2.0  
      d    3.0  
two   c    4.0  
      d    5.0  
      e    6.0
```

```
dtype: float64
```

```
In [134]: data2.unstack().stack(dropna=False)
```

```
Out[134]:
```

```
one   a    0.0  
      b    1.0  
      c    2.0  
      d    3.0  
      e    NaN  
two   a    NaN  
      b    NaN  
      c    4.0  
      d    5.0  
      e    6.0
```

```
dtype: float64
```

Pivot-melt (свести-разобрать?)

```
In [157]: df = pd.DataFrame({'key': ['foo', 'bar', 'baz'],
.....:                 'A': [1, 2, 3],
.....:                 'B': [4, 5, 6],
.....:                 'C': [7, 8, 9]})
```

```
In [158]: df
Out[158]:
   A  B  C  key
0  1  4  7  foo
1  2  5  8  bar
2  3  6  9  baz
```

```
In [159]: melted = pd.melt(df, ['key'])
```

```
In [160]: melted
Out[160]:
    key variable  value
0  foo        A      1
1  bar        A      2
2  baz        A      3
3  foo        B      4
4  bar        B      5
5  baz        B      6
6  foo        C      7
7  bar        C      8
8  baz        C      9
```

```
In [161]: reshaped = melted.pivot('key', 'variable', 'value')
```

```
In [162]: reshaped
Out[162]:
variable  A  B  C
key
bar      2  5  8
baz      3  6  9
foo      1  4  7
```

Как это сделать?

```
key variable value
0   foo       A    1
1   bar       A    2
2   baz       A    3
3   foo       B    4
4   bar       B    5
5   baz       B    6
```

```
variable value
0       A    1
1       A    2
2       A    3
3       B    4
4       B    5
5       B    6
6       C    7
7       C    8
8       C    9
```

```
variable value
0       key  foo
1       key  bar
2       key  baz
3           A   1
4           A   2
5           A   3
6           B   4
7           B   5
8           B   6
```

```
In [157]: df = pd.DataFrame({'key': ['foo', 'bar', 'baz'],
.....:                   'A': [1, 2, 3],
.....:                   'B': [4, 5, 6],
.....:                   'C': [7, 8, 9]})
```

```
In [158]: df
```

```
Out[158]:
```

	A	B	C	key
0	1	4	7	foo
1	2	5	8	bar
2	3	6	9	baz