Lecture 11: Merging and reshaping

Because data isn’t always in the right packaging to start!

Combine data with merge

```
In [2]: df1
Out[2]:
  name  val1
0    a     1
1    b     2
2    c     3
```

```
In [3]: df2
Out[3]:
  name  val2
0    b     4
1    c     5
2    d     6
```

Combine data with merge

```
In [2]: df1
Out[2]:
  name  val1
0    a     1
1    b     2
2    c     3
```

```
In [3]: df2
Out[3]:
  name  val2
0    b     4
1    c     5
2    d     6
```

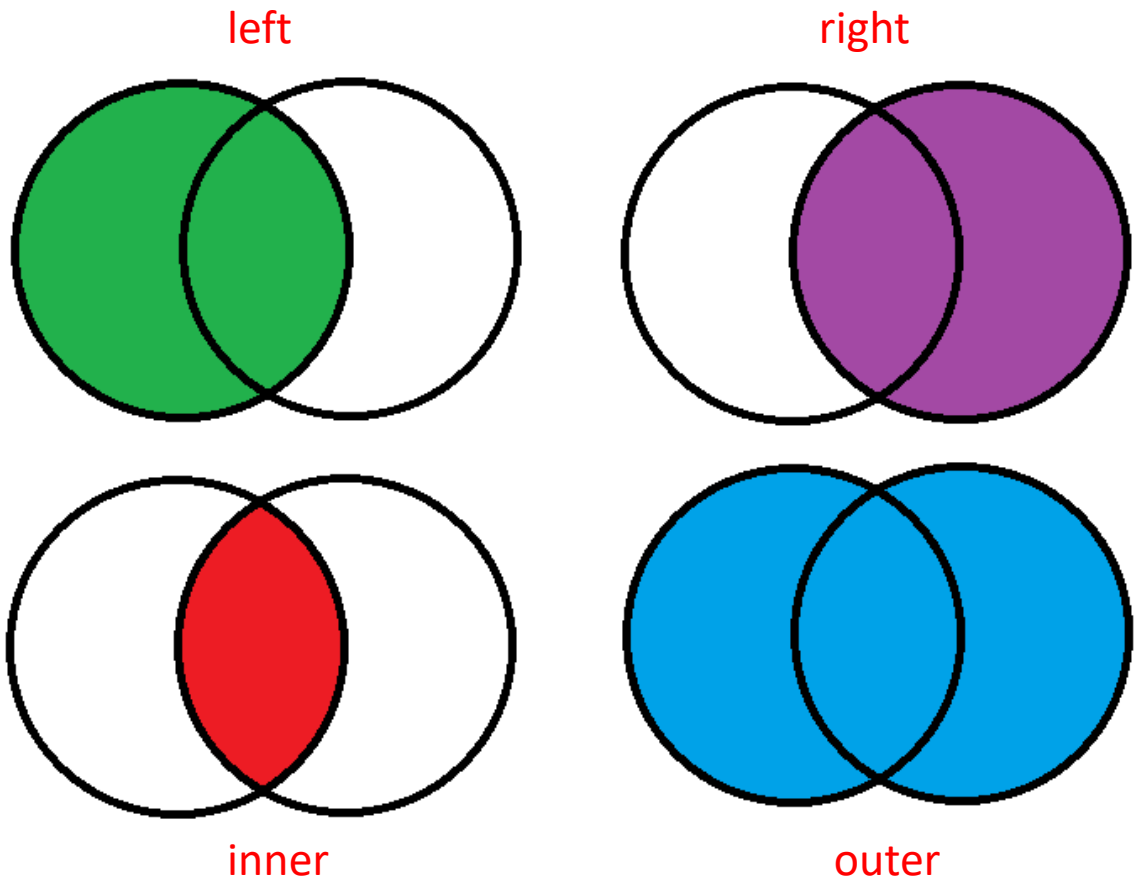
Arguments for matching data:

- on = column name
- left_on, right_on = column name, column name
- left_index, right_index = boolean, boolean

Arguments for match logic:

- how = “left”
- how = “right”
- how = “inner”
- how = “outer”

Combine data with merge



Combine data with merge: outer

```
In [2]: df1
Out[2]:
  name  val1
0    a     1
1    b     2
2    c     3
```

```
In [3]: df2
Out[3]:
  name  val2
0    b     4
1    c     5
2    d     6
```

```
In [6]: df1.merge(df2, on='name', how='outer')
Out[6]:
  name  val1  val2
0    a   1.0   NaN
1    b   2.0   4.0
2    c   3.0   5.0
3    d   NaN   6.0
```

Combine data with merge: outer

```
In [2]: df1
Out[2]:
  name  val1
0    a     1
1    b     2
2    c     3
```

```
In [3]: df2
Out[3]:
  name  val2
0    b     4
1    c     5
2    d     6
```

```
In [6]: df1.merge(df2, on='name', how='outer')
Out[6]:
  name  val1  val2
0    a   1.0   NaN
1    b   2.0   4.0
2    c   3.0   5.0
3    d   NaN   6.0
```

Why did the integers turn to floats?

Combine data with merge: outer

```
In [2]: df1
Out[2]:
  name  val1
0    a     1
1    b     2
2    c     3
```

```
In [3]: df2
Out[3]:
  name  val2
0    b     4
1    c     5
2    d     6
```

```
In [6]: df1.merge(df2, on='name', how='outer')
Out[6]:
  name  val1  val2
0    a   1.0   NaN
1    b   2.0   4.0
2    c   3.0   5.0
3    d   NaN   6.0
```

Combine data with merge: inner

```
In [2]: df1
Out[2]:
  name  val1
0    a     1
1    b     2
2    c     3
```

```
In [3]: df2
Out[3]:
  name  val2
0    b     4
1    c     5
2    d     6
```

```
In [9]: df1.merge(df2, on='name', how='inner')
Out[9]:
  name  val1  val2
0    b     2     4
1    c     3     5
```

Combine data with merge: left

```
In [2]: df1
Out[2]:
  name  val1
0    a     1
1    b     2
2    c     3
```

```
In [3]: df2
Out[3]:
  name  val2
0    b     4
1    c     5
2    d     6
```

```
In [8]: df1.merge(df2, on='name', how='left')
Out[8]:
  name  val1  val2
0    a     1   NaN
1    b     2   4.0
2    c     3   5.0
```

Combine data with merge: right

```
In [2]: df1
Out[2]:
  name  val1
0    a     1
1    b     2
2    c     3
```

```
In [3]: df2
Out[3]:
  name  val2
0    b     4
1    c     5
2    d     6
```

```
In [10]: df1.merge(df2, on='name', how='right')
Out[10]:
  name  val1  val2
0    b   2.0     4
1    c   3.0     5
2    d   NaN     6
```

Combine data with merge

```
In [11]: df1.merge(df2, on='name', how='left')
Out[11]:
  name  val1  val2
0    a     1   NaN
1    b     2   4.0
2    c     3   5.0
```

```
In [12]: df2.merge(df1, on='name', how='right')
Out[12]:
  name  val2  val1
0    a   NaN     1
1    b   4.0     2
2    c   5.0     3
```

Auditing merges

```
In [2]: df1
Out[2]:
  name  val1
0    a     1
1    b     2
2    c     3
```

```
In [19]: df3
Out[19]:
  name  val3
0    a     7
1    a     8
2    b     9
```

Auditing merges

```
In [2]: df1
Out[2]:
  name  val1
0    a     1
1    b     2
2    c     3
```

```
In [19]: df3
Out[19]:
  name  val3
0    a     7
1    a     8
2    b     9
```

```
In [20]: df1.merge(df3, on='name', how='left')
Out[20]:
  name  val1  val3
0    a     1   7.0
1    a     1   8.0
2    b     2   9.0
3    c     3   NaN
```

?

Auditing merges

```
In [2]: df1
Out[2]:
  name  val1
0    a     1
1    b     2
2    c     3
```

```
In [19]: df3
Out[19]:
  name  val3
0    a     7
1    a     8
2    b     9
```

```
25  df1.merge(df3, on='name', how='left', validate='one_to_one')
```

```
MergeError: Merge keys are not unique in right dataset; not a one-to-
one merge
```

Auditing merges

```
start_len = len(df1)
df_merged = df1.merge(df2, on='name', how='outer')
end_len = len(df_merged)

assert(start_len == end_len), 'Unexpected dataframe expansion after merge'
```

Auditing merges

```
start_len = len(df1)
df_merged = df1.merge(df2, on='name', how='outer')
end_len = len(df_merged)

assert(start_len == end_len), 'Unexpected dataframe expansion after merge'
```

```
Traceback (most recent call last):

  File "<ipython-input-24-2c04b756cabe>", line 1, in <module>
    assert(start_len == end_len), 'Unexpected dataframe expansion
after merge'

AssertionError: Unexpected dataframe expansion after merge
```

Auditing merges

```
36    df_merged = df1.merge(df2, on='name', how='outer', indicator=True)
```

```
In [29]: df_merged
Out[29]:
  name  val1  val2     _merge
0    a   1.0   NaN  left_only
1    b   2.0   4.0       both
2    c   3.0   5.0       both
3    d   NaN   6.0 right_only
```

Auditing merges

```
In [29]: df_merged
Out[29]:
  name  val1  val2     _merge
0    a   1.0   NaN  left_only
1    b   2.0   4.0       both
2    c   3.0   5.0       both
3    d   NaN   6.0 right_only
```

```
In [31]: df_merged.dtypes
Out[31]:
name        object
val1       float64
val2       float64
_merge    category
dtype: object
```

Auditing merges

```
In [32]: df_merged['_merge']
Out[32]:
0      left_only
1           both
2           both
3     right_only
Name: _merge, dtype: category
Categories (3, object): ['left_only', 'right_only', 'both']
```

```
In [29]: df_merged
Out[29]:
  name  val1  val2      _merge
0    a   1.0   NaN   left_only
1    b   2.0   4.0        both
2    c   3.0   5.0        both
3    d   NaN   6.0  right_only
```

```
In [31]: df_merged.dtypes
Out[31]:
name         object
val1        float64
val2        float64
_merge     category
dtype: object
```

Auditing merges

```
In [32]: df_merged['_merge']
Out[32]:
0      left_only
1           both
2           both
3     right_only
Name: _merge, dtype: category
Categories (3, object): ['left_only', 'right_only', 'both']
```

```
In [29]: df_merged
Out[29]:
  name  val1  val2      _merge
0    a   1.0   NaN   left_only
1    b   2.0   4.0        both
2    c   3.0   5.0        both
3    d   NaN   6.0  right_only
```

```
In [31]: df_merged.dtypes
Out[31]:
name         object
val1        float64
val2        float64
_merge     category
dtype: object
```

```
In [34]: df_merged[df_merged['_merge'] != 'both']
Out[34]:
  name  val1  val2      _merge
0    a   1.0   NaN   left_only
3    d   NaN   6.0  right_only
```

Combine data with merge: different merge keys

```
In [2]: df1
Out[2]:
  name  val1
0    a     1
1    b     2
2    c     3
```

```
In [36]: df4
Out[36]:
  NAMES  val4
0     a    10
1     b    11
2     c    12
```

```
In [37]: df1.merge(df4, left_on='name', right_on='NAMES', how='inner')
Out[37]:
  name  val1 NAMES  val4
0    a     1     a    10
1    b     2     b    11
2    c     3     c    12
```

Combine data with merge: multiple keys

```
In [43]: df5
Out[43]:
  name  month  val5
0    a      1    13
1    a      6    14
2    b      1    15
3    b      6    16
```

```
In [44]: df6
Out[44]:
  name  month  val6
0    a      1    17
1    a      6    18
2    b      1    19
3    b      6    20
```

```
In [45]: df5.merge(df6, on=['name', 'month'], how='inner')
Out[45]:
  name  month  val5  val6
0    a      1    13    17
1    a      6    14    18
2    b      1    15    19
3    b      6    16    20
```

Combine data with concat

```
In [2]: df1
Out[2]:
  name  val1
0    a     1
1    b     2
2    c     3
```

```
In [48]: df7
Out[48]:
  name  val1
0    d    21
1    e    22
2    f    23
```

```
In [49]: pd.concat([df1, df7])
Out[49]:
  name  val1
0    a     1
1    b     2
2    c     3
0    d    21
1    e    22
2    f    23
```

Reshaping: wide to long (with stubs)

```
In [51]: df
Out[51]:
  student  grade2019  grade2018  grade2017
0       A       4.00        4.0        3.0
1       B       3.50        4.0        3.0
2       C       3.75        3.5        3.5
```

“wide” data →

Reshaping: wide to long (with stubs)

```
In [51]: df
Out[51]:
  student  grade2019  grade2018  grade2017
0       A       4.00        4.0        3.0
1       B       3.50        4.0        3.0
2       C       3.75        3.5        3.5
```

"wide" data →

```
In [53]: pd.wide_to_long(df, stubnames='grade', i='student', j='year')
Out[53]:
               grade
student year
A       2019    4.00
B       2019    3.50
C       2019    3.75
A       2018    4.00
B       2018    4.00
C       2018    3.50
A       2017    3.00
B       2017    3.00
C       2017    3.50
```

"long" data

Reshaping: wide to long (no stubs)

```
In [56]: df
Out[56]:
  student  2019  2018  2017
0       A  4.00   4.0   3.0
1       B  3.50   4.0   3.0
2       C  3.75   3.5   3.5
```

Reshaping: wide to long (no stubs)

```
In [56]: df
Out[56]:
  student  2019  2018  2017
0       A  4.00   4.0   3.0
1       B  3.50   4.0   3.0
2       C  3.75   3.5   3.5
```

```
In [57]: df.melt(id_vars='student', value_vars=None, var_name='year',
value_name='grade')
Out[57]:
  student  year  grade
0       A  2019   4.00
1       B  2019   3.50
2       C  2019   3.75
3       A  2018   4.00
4       B  2018   4.00
5       C  2018   3.50
6       A  2017   3.00
7       B  2017   3.00
8       C  2017   3.50
```

Reshaping: long to wide

```
In [59]: df
Out[59]:
  student  year  grade
0       A  2019   4.00
1       B  2019   3.50
2       C  2019   3.75
3       A  2018   4.00
4       B  2018   4.00
5       C  2018   3.50
6       A  2017   3.00
7       B  2017   3.00
8       C  2017   3.50
```

```
In [60]: df.pivot(index='student', columns='year', values='grade')
Out[60]:
year     2017  2018  2019
student
A         3.0   4.0  4.00
B         3.0   4.0  3.50
C         3.5   3.5  3.75
```

Reshaping guide

https://pandas.pydata.org/docs/user_guide/reshaping.html