

Module 5

Use Case 5

LESSON 5

Data Preparation: Cleaning and Wrangling II

José Manuel García Nieto – University of Málaga





- Data Wrangling II: Advanced GroupBy and Chaining
 - Advanced GroupBy Use
 - Group Transforms and "Unwrapped" GroupBys
 - Grouped Time Resampling
 - Techniques for Method Chaining
 - The pipe Method





- Advanced GroupBy Use: Group Transforms and "Unwrapped" GroupBys
 - Another built-in method is transform
 - It is similar to apply but imposes more constraints on the kind of function
 - It can produce a scalar value to be broadcast to the shape of the group
 - It can produce an object of the same shape as the input group

```
It must not mutate its input
                                                                                     In [78]: q.mean()
                                                                                     Out[78]:
In [75]: df = pd.DataFrame({'key': ['a', 'b', 'c'] * 4,
                                                                                     key
                                                                                          4.5
                          'value': np.arange(12.)})
                                                                                                                     In [79]: g.transform(lambda x: x.mean())
                                                                                         5.5
                                                                                                                     Out[79]:
                                                                                         6.5
In [76]: df
                                                                                     Name: value, dtype: float64
Out[76]:
                                                                                                                           5.5
                                                                                                                           6.5
                                 In [77]: g = df.groupby('key').value
                                                                                                                           4.5
                                                                                                                           5.5
                                                                                                                           6.5
                                                                                                                           4.5
                                                                                                                           5.5
                                                                                                                           6.5
                                                                                                                           4.5
                                                              g Object compiling the groupby
                                                                                                                           5.5
        10.0
                                                                                                                     Name: value, dtype: float64
       11.0
```





- Advanced GroupBy Use: <u>Group Transforms and</u> <u>"Unwrapped" GroupBys</u>
 - Another uses of transform
 - Computing the ranks in descending order for each group
 - Built-in aggregate functions like 'mean', 'sum', or 'std' are often much faster than a general apply function

```
In [86]: g.transform('mean')
Out[86]:
0     4.5
1     5.5
2     6.5
3     4.5
4     5.5
5     6.5
6     4.5
7     5.5
8     6.5
9     4.5
10     5.5
11     6.5
Name: value, dtype: float64
```

This allows to perform a so-called *unwrapped* group operation



```
In [82]: g.transform(lambda x: x.rank(ascending=False))
Out[82]:
0     4.0
1     4.0
2     4.0
3     3.0
4     3.0
5     3.0
6     2.0
7     2.0
8     2.0
9     1.0
10     1.0
Name: value, dtype: float64
```

```
In [87]: normalized = (df['value'] - g.transform('mean')) / g.transform('std')
In [88]: normalized
Out[88]:
     -1.161895
     -1.161895
     -1.161895
     -0.387298
     -0.387298
     -0.387298
     0.387298
      0.387298
      0.387298
      1.161895
      1.161895
      1,161895
Name: value, dtype: float64
```

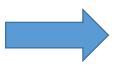




- Advanced GroupBy Use: Grouped Time Resampling
 - For time series data, the resample method is semantically a group operation based on a time intervalization

```
In [89]: N = 15
In [90]: times = pd.date_range('2017-05-20 00:00', freq='1min', periods=N)
In [91]: df = pd.DataFrame({'time': times,
                            'value': np.arange(N)})
In [92]: df
Out[92]:
0 2017-05-20 00:00:00
1 2017-05-20 00:01:00
2 2017-05-20 00:02:00
3 2017-05-20 00:03:00
4 2017-05-20 00:04:00
5 2017-05-20 00:05:00
6 2017-05-20 00:06:00
7 2017-05-20 00:07:00
8 2017-05-20 00:08:00
9 2017-05-20 00:09:00
10 2017-05-20 00:10:00
11 2017-05-20 00:11:00
                           11
                           12
12 2017-05-20 00:12:00
13 2017-05-20 00:13:00
                           13
14 2017-05-20 00:14:00
                           14
```

Index by 'time' and then resample

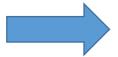






- Advanced GroupBy Use: <u>Grouped Time</u> <u>Resampling</u>
 - Suppose that a DataFrame contains multiple time series, marked by an additional group key column
 - To do the same resampling for each value of 'key', we introduce the pandas.Time Grouper object

```
In [96]: time_key = pd.TimeGrouper('5min')
```



 Then it is possible to set the time index, group by 'key' and time_key, and aggregate

```
In [94]: df2 = pd.DataFrame({'time': times.repeat(3),
                              'key': np.tile(['a', 'b', 'c'], N),
                              'value': np.arange(N * 3.)})
   . . . . :
In [95]: df2[:7]
Out[95]:
  key
                     time
                           value
    a 2017-05-20 00:00:00
                              0.0
                              1.0
    b 2017-05-20 00:00:00
    c 2017-05-20 00:00:00
                              2.0
    a 2017-05-20 00:01:00
                              3.0
    b 2017-05-20 00:01:00
                              4.0
    c 2017-05-20 00:01:00
                              5.0
    a 2017-05-20 00:02:00
                              6.0
```

```
In [97]: resampled = (df2.set_index('time')
                      .groupby(['key', time_key])
                      .sum())
In [98]: resampled
Out[98]:
                         value
kev time
    2017-05-20 00:00:00
                          30.0
    2017-05-20 00:05:00
                         105.0
    2017-05-20 00:10:00 180.0
    2017-05-20 00:00:00
                          35.0
    2017-05-20 00:05:00
                         110.0
    2017-05-20 00:00:00
                          40.0
    2017-05-20 00:05:00
                         115.0
    2017-05-20 00:10:00 190.0
```





- Techniques for Method Chaining
 - When applying a sequence of transformations to a dataset, it is usual to create numerous temporary variables that are never used in your analysis
 - When using functions that accept and return Series or DataFrame objects, you can rewrite this using calls to pipe

The statement f(df) and df.pipe(f) are equivalent, but pipe makes chained invocation easier





References

• Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Ipython, by Wes McKinney (O'Reilly)

Introduction to Machine Learning with Python by Andreas Mueller and Sara

Guido (O'Reilly)

https://pandas.pydata.org/

