Lecture 05.1 Pivot

September 7, 2023

1 Pivot

Duncan Callaway

This notebook gives an introduction to using the Pandas' `pivot` method. It can be accompanied by a simple set of power point slides on how pivoting works.

Pivot is used to examine aggregates with respect to two characteristics. You might construct a pivot of sales data if you wanted to look at average sales broken down by year and market.

The pivot operation is essentially a `groupby` operation that transforms the rows *and the columns.*

[1]:
```
import pandas as pd
import numpy as np
```

1.0.1 Warm up example

[2]:
```
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)})
df
```

[2]:

	key1	key2	data1	data2
0	a	one	1.949277	-1.300230
1	a	two	-0.395316	-1.592518
2	b	one	-0.093050	0.303410
3	b	two	0.509737	1.018390
4	a	one	-1.538981	0.232306

We can do a groupby operation on the data1 column by key1:

[3]:
```
grouped = df['data1'].groupby(df['key1'])
```

And then aggregate. For example averaging all data1 values with the same key1 would give:

[4]:
```
grouped.mean()
```

[4]:
```
key1
a    0.004994
b    0.208344
```

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

Now let's talk about pivot. It is a generalization of groupby:

[5]:
```
df.pivot_table(
    values  = 'data1', # the entry to aggregate over
    index   = 'key1',  # the row grouping attributes
    columns = 'key2',    # the column grouping attributes
    aggfunc = 'mean'   # the aggregation function
)
```

[5]:
```
key2       one       two
key1
a     0.205148 -0.395316
b    -0.093050  0.509737
```

1.0.2 Q: How do you think you might return the same info as in our groupby above, but using pivot?

[6]:
```
df.pivot_table(
    values  = 'data1', # the entry to aggregate over
    index   = 'key1',  # the row grouping attributes
    #columns = 'key2',    # the column grouping attributes
    aggfunc = 'mean'   # the aggregation function
)
```

[6]:
```
          data1
key1
a     0.004994
b     0.208344
```

1.0.3 A: drop the column key, as above.

1.0.4 Using pivot on a more detailed data set

First let's get our data in order.

[7]:
```
cds = pd.read_csv('CAISO_2017to2018_stack.csv', index_col=0)
cds
```

[7]:
```
                            Source   MWh
2017-08-29 00:00:00     GEOTHERMAL  1181
2017-08-29 00:00:00        BIOMASS   340
2017-08-29 00:00:00          BIOGAS   156
2017-08-29 00:00:00    SMALL HYDRO   324
2017-08-29 00:00:00     WIND TOTAL  1551
…                                  …   …
2018-08-28 23:00:00          BIOGAS   235
2018-08-28 23:00:00    SMALL HYDRO   262
```

```
2018-08-28 23:00:00     WIND TOTAL  2921
2018-08-28 23:00:00       SOLAR PV     0
2018-08-28 23:00:00  SOLAR THERMAL     0

[61320 rows x 2 columns]
```

1.0.5 Q: I'd like to organize the data by source and hour of day. How should I do that?

We'll start by extracting the hour of day from the index.

An amazing trick to take text that represents date/time info and turn it into more meaningful data is the `pd.to_datetime` method:

[8]:
```
cds_time = pd.to_datetime(cds.index)
type(cds_time)
```

[8]:
```
pandas.core.indexes.datetimes.DatetimeIndex
```

Now we can extract year, month, day… information from the datetimeindex:

[9]:
```
cds_time[0].month
```

[9]:
```
8
```

You can even do this for the entire object in one fell swoop:

[10]:
```
cds_time.hour
```

[10]:
```
Index([ 0,  0,  0,  0,  0,  0,  0,  1,  1,  1,
       …
       22, 22, 22, 23, 23, 23, 23, 23, 23, 23],
      dtype='int32', length=61320)
```

1.0.6 Q: Now that we have hours, what next?

1.0.7 A: We can add hours from the `cds_time` object into the dataframe as follows:

[11]:
```
cds['hour'] = cds_time.hour
cds.head()
```

[11]:
```
                        Source   MWh  hour
2017-08-29 00:00:00   GEOTHERMAL  1181     0
2017-08-29 00:00:00      BIOMASS   340     0
2017-08-29 00:00:00       BIOGAS   156     0
2017-08-29 00:00:00  SMALL HYDRO   324     0
2017-08-29 00:00:00   WIND TOTAL  1551     0
```

1.0.8 Q: Try it for yourself: Create a pivot table with average hourly generation

[12]:
```
cds.pivot_table(
    values  = 'MWh',    # the entry to aggregate over
    index   = 'hour',   # the row grouping attributes
    columns = 'Source', # the column grouping attributes
    aggfunc = 'mean'    # the aggregation function
)
```

[12]:
Source	BIOGAS	BIOMASS	GEOTHERMAL	SMALL HYDRO	SOLAR PV	SOLAR THERMAL	WIND TOTAL
hour							
0	225.591781	318.301370	958.720548	330.824658	0.679452	0.000000	2173.268493
1	225.964384	318.369863	959.235616	322.421918	0.643836	0.000000	2120.778082
2	225.953425	319.846575	959.367123	318.249315	0.635616	0.000000	2051.832877
3	225.887671	320.567123	958.367123	316.909589	0.419178	0.000000	1973.969863
4	225.753425	321.742466	956.347945	322.254795	0.413699	0.000000	1881.463014
5	225.243836	323.863014	956.230137	375.180822	0.482192	0.021918	1772.484932
6	224.479452	330.808219	955.682192	426.931507	352.956164	4.372603	1646.630137
7	222.454795	333.178082	953.263014	422.564384	2489.268493	58.317808	1490.194521
8	221.536986	333.936986	949.024658	376.813699	5552.531507	208.106849	1363.402740
9	221.539726	332.273973	946.210959	343.756164	7174.468493		
10	221.408219	330.134247	943.405479	336.980822	7829.561644		
11	221.802740	329.369863	938.967123	338.473973	8047.476712		
12	222.731507	330.736986	936.627397	346.378082	8080.778082		
13	223.290411	332.043836	936.600000	353.904110	7985.413699		
14	223.465753	333.649315	935.613699	362.726027	7619.213699		
15	223.717808	335.821918	935.564384	397.095890	6639.180822		
16	224.342466	338.265753	937.736986	448.476712	5025.484932		
17	224.767123	337.016438	940.526027	497.013699	3299.101370		
18	225.473973	337.273973	946.139726	524.641096	1412.446575		
19	226.136986	337.273973	950.232877	531.112329	210.783562		
20	225.958904	334.961644	953.882192	515.457534	4.750685		
21	225.797260	331.964384	956.684932	474.386301	1.542466		
22	225.430137	321.104110	958.164384	423.049315	1.312329		
23	225.498630	318.375342	958.898630	372.060274	1.046575		

```
9          316.841096  1290.512329
10         355.441096  1250.408219
11         368.904110  1247.643836
12         369.643836  1308.287671
13         356.778082  1412.410959
14         337.479452  1561.602740
15         304.389041  1726.652055
16         247.583562  1878.441096
17         185.915068  2005.934247
18          85.958904  2109.093151
19          10.364384  2181.361644
20           1.279452  2229.408219
21           0.539726  2231.687671
22           0.136986  2220.109589
23           0.104110  2216.526027
```

1.0.9 Q: In class challenge:

create a pivot table where source is the columns, the *month* is the row, and you aggregate into maximum values.

Hint: write `max` to represent standard deviation.

[13]:
```
cds['month'] = cds_time.month
```

[14]:
```
cds_piv = cds.pivot_table(
    values  = 'MWh',
    index   = 'month',
    columns = 'Source',
    aggfunc = 'max'
)
cds_piv
```

[14]:
```
Source  BIOGAS  BIOMASS  GEOTHERMAL  SMALL HYDRO  SOLAR PV  SOLAR THERMAL  \
month
1          249      376         999          585      8024            397
2          248      374        1012          572      9369            441
3          248      344        1012          564      9795            583
4          248      336         967          681     10027            589
5          253      359        1005          663     10050            604
6          247      436        1009          659     10102            652
7          240      482        1009          662      9997            636
8          243      399        1212          652      9930            679
9          178      421        1230          577      9044            667
10         238      423        1009          583      8909            541
11         242      380        1003          636      7550            443
12         248      390         983          654      7178            431
```

Source	WIND TOTAL
month	
1	4015
2	4420
3	4108
4	4531
5	4925
6	5006
7	4466
8	4675
9	3943
10	4426
11	3567
12	3978