PivotTable_ed

May 25, 2022

A pivot table is a way of summarizing data in a DataFrame for a particular purpose. It makes heavy use of the aggregation function. A pivot table is itself a DataFrame, where the rows represent one variable that you're interested in, the columns another, and the cell's some aggregate value. A pivot table also tends to includes marginal values as well, which are the sums for each column and row. This allows you to be able to see the relationship between two variables at just a glance.

```python
[1]: # Lets take a look at pivot tables in pandas
     import pandas as pd
     import numpy as np
```

```python
[3]: # Here we have the Times Higher Education World University Ranking dataset,
     # ↪which is one of the most
     # influential university measures. Let's import the dataset and see what it
     # ↪looks like
     df = pd.read_csv('datasets/cwurData.csv')
     df.head()
```

```
[3]: 2200
```

```python
[5]: # Here we can see each institution's rank, country, quality of education, other
     # ↪metrics, and overall score.
     # Let's say we want to create a new column called Rank_Level, where
     # ↪institutions with world ranking 1-100 are
     # categorized as first tier and those with world ranking 101 - 200 are second
     # ↪tier, ranking 201 - 300 are
     # third tier, after 301 is other top universities.

     # Now, you actually already have enough knowledge to do this, so why don't you
     # ↪pause the video and give it a
     # try?

     # Here's my solution, I'm going to create a function called create_category
     # ↪which will operate on the first
     # column in the dataframe, world_rank
     def create_category(ranking):
         # Since the rank is just an integer, I'll just do a bunch of if/elif
         # ↪statements
         if (ranking >= 1) & (ranking <= 100):
```

```python
        return "First Tier Top Unversity"
    elif (ranking >= 101) & (ranking <= 200):
        return "Second Tier Top Unversity"
    elif (ranking >= 201) & (ranking <= 300):
        return "Third Tier Top Unversity"
    return "Other Top Unversity"

# Now we can apply this to a single column of data to create a new series
# poner la función lambda acá es redundande pero viene bien entenderla
df['Rank_Level'] = df['world_rank'].apply(lambda x: create_category(x))

# otra forma de hacer lo mismo
df['Rank_Level'] = df['world_rank'].apply(create_category)
# And lets look at the result
df.head()
```

[5]:

	world_rank	institution	country \
0	1	Harvard University	USA
1	2	Massachusetts Institute of Technology	USA
2	3	Stanford University	USA
3	4	University of Cambridge	United Kingdom
4	5	California Institute of Technology	USA

	national_rank	quality_of_education	alumni_employment	quality_of_faculty \
0	1	7	9	1
1	2	9	17	3
2	3	17	11	5
3	1	10	24	4
4	4	2	29	7

	publications	influence	citations	broad_impact	patents	score	year \
0	1	1	1	NaN	5	100.00	2012
1	12	4	4	NaN	1	91.67	2012
2	4	2	2	NaN	15	89.50	2012
3	16	16	11	NaN	50	86.17	2012
4	37	22	22	NaN	18	85.21	2012

	Rank_Level
0	First Tier Top Unversity
1	First Tier Top Unversity
2	First Tier Top Unversity
3	First Tier Top Unversity
4	First Tier Top Unversity

[20]:
```python
# Vemos que score es una columna

df.columns
```

```
[20]: Index(['world_rank', 'institution', 'country', 'national_rank',
             'quality_of_education', 'alumni_employment', 'quality_of_faculty',
             'publications', 'influence', 'citations', 'broad_impact', 'patents',
             'score', 'year', 'Rank_Level', 'Rank_Level_1'],
            dtype='object')
```

```
[6]: # A pivot table allows us to pivot out one of these columns a new column␣
     →headers and compare it against
     # another column as row indices. Let's say we want to compare rank level versus␣
     →country of the universities
     # and we want to compare in terms of overall score

     # To do this, we tell Pandas we want the values to be Score, and index to be␣
     →the country and the columns to be
     # the rank levels. Then we specify that the aggregation function, and here␣
     →we'll use the NumPy mean to get the
     # average rating for universities in that country

     df.pivot_table(values='score', index='country', columns='Rank_Level',␣
     →aggfunc=[np.mean]).head()
```

```
[6]:                              mean                          \
     Rank_Level First Tier Top Unversity Other Top Unversity
     country
     Argentina                     NaN            44.672857
     Australia                 47.9425            44.645750
     Austria                       NaN            44.864286
     Belgium                   51.8750            45.081000
     Brazil                        NaN            44.499706


     Rank_Level Second Tier Top Unversity Third Tier Top Unversity
     country
     Argentina                        NaN                      NaN
     Australia                    49.2425                47.285000
     Austria                          NaN                47.066667
     Belgium                      49.0840                46.746667
     Brazil                       49.5650                      NaN
```

```
[12]: df[df['country']=='Argentina']
```

```
[12]:       world_rank                    institution     country  national_rank  \
      577          378        University of Buenos Aires   Argentina              1
      940          741   National University of La Plata   Argentina              2
      1096         897    National University of Córdoba  Argentina              3
      1177         978    National University of Rosario  Argentina              4
      1566         367        University of Buenos Aires   Argentina              1
      1943         744   National University of La Plata   Argentina              2
      2132         933    National University of Córdoba  Argentina              3
```

	quality_of_education	alumni_employment	quality_of_faculty
577	192	477	176
940	355	476	210
1096	355	478	210
1177	355	478	210
1566	126	500	180
1943	367	513	218
2132	367	567	218

	publications	influence	citations	broad_impact	patents	score	year
577	268	348	363	349.0	737	45.66	2014
940	546	523	406	703.0	737	44.59	2014
1096	713	818	609	849.0	737	44.39	2014
1177	976	811	800	956.0	737	44.29	2014
1566	276	350	321	344.0	871	45.37	2015
1943	546	559	511	686.0	871	44.31	2015
2132	717	796	645	896.0	871	44.10	2015

	Rank_Level
577	Other Top Unversity
940	Other Top Unversity
1096	Other Top Unversity
1177	Other Top Unversity
1566	Other Top Unversity
1943	Other Top Unversity
2132	Other Top Unversity

[5]:
```python
# We can see a  hierarchical dataframe where the index, or rows, are by country
 ↪and the columns have two
# levels, the top level indicating that the mean value is being used and the
 ↪second level being our ranks. In
# this example we only have one variable, the mean, that we are looking at, so
 ↪we don't really need a
# heirarchical index.

# We notice that there are some NaN values, for example, the first row,
 ↪Argentia. The NaN values indicate that
# Argentia has only observations in the "Other Top Unversities" category
```

[13]:
```python
# Now, pivot tables aren't limited to one function that you might want to apply.
 ↪ You can pass a named
# parameter, aggfunc, which is a list of the different functions to apply, and
 ↪pandas will provide you with
# the result using hierarchical column names.  Let's try that same query, but
 ↪pass in the max() function too
```

```python
df.pivot_table(values='score', index='country', columns='Rank_Level',
               aggfunc=[np.mean, np.max]).head()
```

[13]:

	mean	\
Rank_Level	First Tier Top Unversity	Other Top Unversity
country		
Argentina	NaN	44.672857
Australia	47.9425	44.645750
Austria	NaN	44.864286
Belgium	51.8750	45.081000
Brazil	NaN	44.499706

		\
Rank_Level	Second Tier Top Unversity	Third Tier Top Unversity
country		
Argentina	NaN	NaN
Australia	49.2425	47.285000
Austria	NaN	47.066667
Belgium	49.0840	46.746667
Brazil	49.5650	NaN

	amax	\
Rank_Level	First Tier Top Unversity	Other Top Unversity
country		
Argentina	NaN	45.66
Australia	51.61	45.97
Austria	NaN	46.29
Belgium	52.03	46.21
Brazil	NaN	46.08

Rank_Level	Second Tier Top Unversity	Third Tier Top Unversity
country		
Argentina	NaN	NaN
Australia	50.40	47.47
Austria	NaN	47.78
Belgium	49.73	47.14
Brazil	49.82	NaN

[7]:
```python
# So now we see we have both the mean and the max. As mentioned earlier, we can
 also summarize the values
# within a given top level column. For instance, if we want to see an overall
 average for the country for the
# mean and we want to see the max of the max, we can indicate that we want
 pandas to provide marginal values
df.pivot_table(values='score', index='country', columns='Rank_Level',
 aggfunc=[np.mean, np.max],
               margins=True).head()
```

	mean	\
Rank_Level	First Tier Top Unversity	Other Top Unversity
country		
Argentina	NaN	44.672857
Australia	47.9425	44.645750
Austria	NaN	44.864286
Belgium	51.8750	45.081000
Brazil	NaN	44.499706

		\
Rank_Level	Second Tier Top Unversity	Third Tier Top Unversity	All
country			
Argentina	NaN	NaN	44.672857
Australia	49.2425	47.285000	45.825517
Austria	NaN	47.066667	45.139583
Belgium	49.0840	46.746667	47.011000
Brazil	49.5650	NaN	44.781111

	amax	\
Rank_Level	First Tier Top Unversity	Other Top Unversity
country		
Argentina	NaN	45.66
Australia	51.61	45.97
Austria	NaN	46.29
Belgium	52.03	46.21
Brazil	NaN	46.08

Rank_Level	Second Tier Top Unversity	Third Tier Top Unversity	All
country			
Argentina	NaN	NaN	45.66
Australia	50.40	47.47	51.61
Austria	NaN	47.78	47.78
Belgium	49.73	47.14	52.03
Brazil	49.82	NaN	49.82

```python
# A pivot table is just a multi-level dataframe, and we can access series or
 ↪cells in the dataframe in a similar way
# as we do so for a regular dataframe.

# Let's create a new dataframe from our previous example
new_df=df.pivot_table(values='score', index='country', columns='Rank_Level',
 ↪aggfunc=[np.mean, np.max],
             margins=True)
# Now let's look at the index
print(new_df.index)
# And let's look at the columns
```

```
print(new_df.columns)
```

```
Index(['Argentina', 'Australia', 'Austria', 'Belgium', 'Brazil', 'Bulgaria',
       'Canada', 'Chile', 'China', 'Colombia', 'Croatia', 'Cyprus',
       'Czech Republic', 'Denmark', 'Egypt', 'Estonia', 'Finland', 'France',
       'Germany', 'Greece', 'Hong Kong', 'Hungary', 'Iceland', 'India', 'Iran',
       'Ireland', 'Israel', 'Italy', 'Japan', 'Lebanon', 'Lithuania',
       'Malaysia', 'Mexico', 'Netherlands', 'New Zealand', 'Norway', 'Poland',
       'Portugal', 'Puerto Rico', 'Romania', 'Russia', 'Saudi Arabia',
       'Serbia', 'Singapore', 'Slovak Republic', 'Slovenia', 'South Africa',
       'South Korea', 'Spain', 'Sweden', 'Switzerland', 'Taiwan', 'Thailand',
       'Turkey', 'USA', 'Uganda', 'United Arab Emirates', 'United Kingdom',
       'Uruguay', 'All'],
      dtype='object', name='country')
MultiIndex([('mean',  'First Tier Top Unversity'),
            ('mean',        'Other Top Unversity'),
            ('mean', 'Second Tier Top Unversity'),
            ('mean',  'Third Tier Top Unversity'),
            ('mean',                       'All'),
            ('amax',  'First Tier Top Unversity'),
            ('amax',        'Other Top Unversity'),
            ('amax', 'Second Tier Top Unversity'),
            ('amax',  'Third Tier Top Unversity'),
            ('amax',                       'All')],
           names=[None, 'Rank_Level'])
```

```
[16]: # We can see the columns are hierarchical. The top level column indices have
      ↪two categories: mean and max, and
      # the lower level column indices have four categories, which are the four rank
      ↪levels. How would we query this
      # if we want to get the average scores of First Tier Top Unversity levels in
      ↪each country? We would just need
      # to make two dataframe projections, the first for the mean, then the second
      ↪for the top tier
      new_df['mean']['First Tier Top Unversity'].head()
```

```
[16]: country
      Argentina         NaN
      Australia     47.9425
      Austria           NaN
      Belgium       51.8750
      Brazil            NaN
      Name: First Tier Top Unversity, dtype: float64
```

```
[10]: # We can see that the output is a series object which we can confirm by
      ↪printing the type. Remember that when
      # you project a single column of values out of a DataFrame you get a series.
      type(new_df['mean']['First Tier Top Unversity'])
```

```python
[10]:  pandas.core.series.Series

[28]:  # What if we want to find the country that has the maximum average score on
       →First Tier Top University level?
       # We can use the idxmax() function.
       new_df['mean']['First Tier Top Unversity'].idxmax()

[28]:  'United Kingdom'

[12]:  # Now, the idxmax() function isn't special for pivot tables, it's a built in
       →function to the Series object.
       # We don't have time to go over all pandas functions and attributes, and I want
       →to encourage you to explore
       # the API to learn more deeply what is available to you.

[37]:  # If you want to achieve a different shape of your pivot table, you can do so
       →with the stack and unstack
       # functions. Stacking is pivoting the lowermost column index to become the
       →innermost row index. Unstacking is
       # the inverse of stacking, pivoting the innermost row index to become the
       →lowermost column index. An example
       # will help make this clear

       # Let's look at our pivot table first to refresh what it looks like
       new_df.head()
```

[37]:
```
                               mean                        \
Rank_Level First Tier Top Unversity Other Top Unversity
country
Argentina                       NaN            44.672857
Australia                    47.9425            44.645750
Austria                         NaN            44.864286
Belgium                      51.8750            45.081000
Brazil                          NaN            44.499706


                                                                     \
Rank_Level Second Tier Top Unversity Third Tier Top Unversity        All
country
Argentina                        NaN                     NaN   44.672857
Australia                     49.2425               47.285000   45.825517
Austria                          NaN               47.066667   45.139583
Belgium                       49.0840               46.746667   47.011000
Brazil                        49.5650                     NaN   44.781111


                               amax                        \
Rank_Level First Tier Top Unversity Other Top Unversity
country
Argentina                       NaN                45.66
Australia                     51.61                45.97
```

```
Austria                          NaN                   46.29
Belgium                        52.03                   46.21
Brazil                           NaN                   46.08


Rank_Level Second Tier Top Unversity Third Tier Top Unversity     All
country
Argentina                        NaN                          NaN  45.66
Australia                      50.40                        47.47  51.61
Austria                          NaN                        47.78  47.78
Belgium                        49.73                        47.14  52.03
Brazil                         49.82                          NaN  49.82
```

[38]:
```python
# Con .stack() las columnas de nivel más bajas, o sea, aquellas incluidas en
 ↪Rank_Level pasar a ser filas.
# Entonces quedan como columnas las columnas de primer nivel jerárquico: mean y
 ↪amax



# Now let's try stacking, this should move the lowermost column, so the tiers
 ↪of the university rankings, to
# the inner most row
new_df=new_df.stack()
new_df.head()
```

[38]:
```
                                         mean    amax
country   Rank_Level
Argentina Other Top Unversity         44.672857  45.66
          All                         44.672857  45.66
Australia First Tier Top Unversity    47.942500  51.61
          Other Top Unversity         44.645750  45.97
          Second Tier Top Unversity   49.242500  50.40
```

[ ]:

[41]:
```python
# In the original pivot table, rank levels are the lowermost column, after
 ↪stacking, rank levels become the
# innermost index, appearing to the right after country


# Si hacemos unstack() vuelve a su forma original
# Now let's try unstacking
new_df.unstack().head()
```

[41]:
```
                              mean                                \
Rank_Level First Tier Top Unversity Other Top Unversity
country
Argentina                        NaN              44.672857
Australia                     47.9425              44.645750
```

```
Austria                          NaN         44.864286
Belgium                       51.8750        45.081000
Brazil                           NaN         44.499706


                                                                      \
Rank_Level Second Tier Top Unversity Third Tier Top Unversity       All
country
Argentina                         NaN                       NaN  44.672857
Australia                      49.2425                 47.285000  45.825517
Austria                           NaN                 47.066667  45.139583
Belgium                        49.0840                 46.746667  47.011000
Brazil                         49.5650                       NaN  44.781111


                                      amax                      \
Rank_Level First Tier Top Unversity Other Top Unversity
country
Argentina                         NaN               45.66
Australia                       51.61               45.97
Austria                           NaN               46.29
Belgium                         52.03               46.21
Brazil                            NaN               46.08


Rank_Level Second Tier Top Unversity Third Tier Top Unversity      All
country
Argentina                         NaN                       NaN  45.66
Australia                       50.40                     47.47  51.61
Austria                           NaN                     47.78  47.78
Belgium                        49.73                     47.14  52.03
Brazil                         49.82                       NaN  49.82
```

[44]: `# That seems to restore our dataframe to its original shape. What do you think`
`→would happen if we unstacked twice in a row?`
`new_df.unstack().unstack().head()`

[44]:
```
       Rank_Level                country
mean   First Tier Top Unversity  Argentina          NaN
                                 Australia      47.9425
                                 Austria            NaN
                                 Belgium        51.8750
                                 Brazil             NaN
dtype: float64
```

[17]: `# We actually end up unstacking all the way to just a single column, so a`
`→series object is returned. This`
`# column is just a "value", the meaning of which is denoted by the`
`→heirarachical index of operation, rank, and`
`# country.`

So that's pivot tables. This has been a pretty short description, but they're incredibly useful when dealing with numeric data, especially if you're trying to summarize the data in some form. You'll regularly be creating new pivot tables on slices of data, whether you're exploring the data yourself or preparing data for others to report on. And of course, you can pass any function you want to the aggregate function, including those that you define yourself.