Grouping data in pandas

You can group and aggregate data in pandas in ways that will be familiar if you've ever done a pivot table in Excel or a GROUP BY statement in SQL.

In this notebook, we'll use the U.S. eel import data (source) that lives at `../data/eels.csv`.

- value_counts()
- groupby()
- Grouping by multiple columns
- pivot_table()

In [19]:
```
# import pandas
import pandas as pd
```

In [20]:
```
# read the CSV into a data frame
df = pd.read_csv('../data/eels.csv')
```

In [21]:
```
# check the output with `head()`
df.head()
```

Out[21]:

	year	month	country	product	kilos	dollars
0	2010	1	CHINA	EELS FROZEN	49087	393583
1	2010	1	JAPAN	EELS FRESH	263	7651
2	2010	1	TAIWAN	EELS FROZEN	9979	116359
3	2010	1	VIETNAM	EELS FRESH	1938	10851
4	2010	1	VIETNAM	EELS FROZEN	21851	69955

value_counts

If all you need to do is count the occurrences of a value in a column, you can use the `value_counts()` method.

In our eel data, every row is one month's of shipments of a particular eel product from one country. Let's count up how many months each country is represented in the data.

In [ ]:
```
# get value counts of country column
df.country.value_counts()
```

groupby

Let's group the data by country and sum the kilos for each country.

If this were a pivot table in Excel, we'd drag the `country` column into Rows and the `kilos` column into Values, then summarize by Sum.

If this were SQL, we might write something like:

```
SELECT country, SUM(kilos)
FROM eels
GROUP BY country
ORDER BY 2 desc
```

Let's do the same thing in pandas using `groupby`:

- Select a list with our two columns of interest (`country` and `kilos`)
- Call the `groupby()` method on the grouping column (`country`)
- Call the `sum()` method
- Sort by kilos descending

In [23]:
```
df[['country', 'kilos']].groupby('country').sum().sort_values('kilos', ascending=False)
```

Out[23]:

country	kilos
CHINA	15965996
VIETNAM	637737
TAIWAN	442740
JAPAN	361364
CANADA	346075
SOUTH KOREA	243540
THAILAND	137556
PORTUGAL	41453
PAKISTAN	22453
MEXICO	20860
NORWAY	17391
PANAMA	12823
UKRAINE	11414
CHILE	6185
SPAIN	3998
NEW ZEALAND	3822
INDIA	2200
POLAND	2160
SENEGAL	1350
CHINA - HONG KONG	735
BURMA	699
BANGLADESH	613
PHILIPPINES	610
COSTA RICA	563

You can use other aggregations, too -- let's do the `median()`.

In [24]:
```
df[['country', 'kilos']].groupby('country').median().sort_values('kilos', ascending=False)
```

Out[24]:

country	kilos
CHINA	40000.0
PAKISTAN	22453.0
UKRAINE	5707.0
VIETNAM	5326.5
MEXICO	5307.0
TAIWAN	4195.0
CHILE	3092.5
NORWAY	3062.0
INDIA	2200.0
CANADA	2063.0
THAILAND	1598.0
SENEGAL	1350.0
POLAND	1080.0
SOUTH KOREA	998.0
CHINA - HONG KONG	735.0
BURMA	699.0
PHILIPPINES	610.0
PANAMA	602.0
NEW ZEALAND	600.0
COSTA RICA	563.0
PORTUGAL	387.5
BANGLADESH	300.0
SPAIN	243.5
JAPAN	223.0

... and you can do *multiple* aggregations, too, if that's useful. Just use the `agg()` function and pass it a list of functions that you'd like to compute on numeric columns:

In [25]:
```
df[['country', 'kilos']].groupby('country').agg(['sum', 'median', 'mean'])
```

Out[25]:

	kilos		
	sum	median	mean
country			
BANGLADESH	613	300.0	204.333333
BURMA	699	699.0	699.000000
CANADA	346075	2063.0	3426.485149
CHILE	6185	3092.5	3092.500000

	kilos		
	sum	median	mean
country			
CHINA	15965996	40000.0	85379.657754
CHINA - HONG KONG	735	735.0	735.000000
COSTA RICA	563	563.0	563.000000
INDIA	2200	2200.0	2200.000000
JAPAN	361364	223.0	2492.165517
MEXICO	20860	5307.0	6953.333333
NEW ZEALAND	3822	600.0	764.400000
NORWAY	17391	3062.0	4347.750000
PAKISTAN	22453	22453.0	22453.000000
PANAMA	12823	602.0	4274.333333
PHILIPPINES	610	610.0	610.000000
POLAND	2160	1080.0	1080.000000
PORTUGAL	41453	387.5	545.434211
SENEGAL	1350	1350.0	1350.000000
SOUTH KOREA	243540	998.0	3746.769231
SPAIN	3998	243.5	285.571429
TAIWAN	442740	4195.0	5749.870130
THAILAND	137556	1598.0	5502.240000
UKRAINE	11414	5707.0	5707.000000
VIETNAM	637737	5326.5	7592.107143

Grouping by multiple columns

You can group by multiple columns! Just pass a *list* of columns to the `groupby()` method instead of a column name. If, for example, we want to get the total kilos by country by year, we could select our three columns of data to pass to `groupby()` and call the `sum()` function. Like this:

In [26]:
```
df[['country', 'year', 'kilos']].groupby(['country', 'year']).sum()
```

Out[26]:

		kilos
country	year	
BANGLADESH	**2012**	13
	2015	600
BURMA	**2016**	699
CANADA	**2010**	13552
	2011	24968
...	...	...

		kilos
country	**year**	
VIETNAM	**2013**	100828
	2014	38112
	2015	36859
	2016	96179
	2017	28490

89 rows × 1 columns

`pivot_table`

... which is fine and all, but there's a more intuitive way to look at this data, I think: using the `pivot_table()` method.

If we were making this pivot table in Excel, we would drag `country` to Rows, `kilos` to Values and `year` to Columns. But we're gonna do it in pandas. We're gonna hand the `pivot_table()` method four things:

- The data frame you're pivoting (`df`)
- The `index` column -- what to group your data by (`index='country'`)
- The `columns` column -- the second grouping factor (`columns='year'`)
- The `values` column -- what column are we doing math on? (`values='kilos'`)
- The `aggfunc` -- what function to use to aggregate the data; the default is to use an average, but we'll use Python's built-in `sum` function

Then we'll sort the results by the latest year of data -- 2017 -- and fill null values with zeroes.

In [27]:
```
pd.pivot_table(df,
               index='country',
               columns='year',
               values='kilos',
               aggfunc=sum).sort_values(2017, ascending=False) \
                           .fillna(0)
```

Out[27]:

year	2010	2011	2012	2013	2014	2015	2016	2017
country								
CHINA	372397.0	249232.0	1437392.0	1090135.0	1753140.0	4713882.0	4578546.0	1771272.0
TAIWAN	73842.0	0.0	53774.0	39752.0	83478.0	48272.0	99535.0	44087.0
SOUTH KOREA	42929.0	41385.0	28146.0	27353.0	37708.0	8386.0	14729.0	42904.0
JAPAN	1326.0	2509.0	32255.0	105758.0	40177.0	69699.0	71748.0	37892.0
THAILAND	2866.0	5018.0	9488.0	4488.0	15110.0	41771.0	26931.0	31884.0
VIETNAM	63718.0	155488.0	118063.0	100828.0	38112.0	36859.0	96179.0	28490.0
CANADA	13552.0	24968.0	110796.0	44455.0	31546.0	28619.0	68568.0	23571.0
PORTUGAL	2081.0	3672.0	2579.0	2041.0	7215.0	8013.0	9105.0	6747.0
PANAMA	0.0	0.0	0.0	11849.0	0.0	0.0	0.0	974.0

year country	2010	2011	2012	2013	2014	2015	2016	2017
BANGLADESH	0.0	0.0	13.0	0.0	0.0	600.0	0.0	0.0
BURMA	0.0	0.0	0.0	0.0	0.0	0.0	699.0	0.0
CHILE	0.0	0.0	0.0	0.0	6185.0	0.0	0.0	0.0
CHINA - HONG KONG	0.0	0.0	0.0	0.0	0.0	0.0	735.0	0.0
COSTA RICA	0.0	0.0	0.0	0.0	0.0	0.0	563.0	0.0
INDIA	0.0	0.0	0.0	0.0	0.0	0.0	2200.0	0.0
MEXICO	0.0	0.0	0.0	4000.0	0.0	16860.0	0.0	0.0
NEW ZEALAND	0.0	2652.0	900.0	270.0	0.0	0.0	0.0	0.0
NORWAY	0.0	0.0	0.0	17391.0	0.0	0.0	0.0	0.0
PAKISTAN	0.0	0.0	0.0	22453.0	0.0	0.0	0.0	0.0
PHILIPPINES	0.0	0.0	0.0	610.0	0.0	0.0	0.0	0.0
POLAND	0.0	0.0	1296.0	0.0	864.0	0.0	0.0	0.0
SENEGAL	0.0	1350.0	0.0	0.0	0.0	0.0	0.0	0.0
SPAIN	0.0	0.0	977.0	275.0	1019.0	719.0	1008.0	0.0
UKRAINE	0.0	0.0	0.0	0.0	0.0	0.0	11414.0	0.0