*You are currently looking at **version 1.5** of this notebook. To download notebooks and datafiles, as well as get help on Jupyter notebooks in the Coursera platform, visit the Jupyter Notebook FAQ course resource.*

Assignment 3 - More Pandas

This assignment requires more individual learning then the last one did - you are encouraged to check out the pandas documentation to find functions or methods you might not have used yet, or ask questions on Stack Overflow and tag them as pandas and python related. And of course, the discussion forums are open for interaction with your peers and the course staff.

Question 1 (20%)

Load the energy data from the file `Energy Indicators.xls`, which is a list of indicators of energy supply and renewable electricity production from the United Nations for the year 2013, and should be put into a DataFrame with the variable name of **energy**.

Keep in mind that this is an Excel file, and not a comma separated values file. Also, make sure to exclude the footer and header information from the datafile. The first two columns are unneccessary, so you should get rid of them, and you should change the column labels so that the columns are:

`['Country', 'Energy Supply', 'Energy Supply per Capita', '% Renewable']`

Convert `Energy Supply` to gigajoules (there are 1,000,000 gigajoules in a petajoule). For all countries which have missing data (e.g. data with "...") make sure this is reflected as `np.NaN` values.

Rename the following list of countries (for use in later questions):

```
"Republic of Korea": "South Korea",
"United States of America": "United States",
"United Kingdom of Great Britain and Northern Ireland": "United Kingdom",
"China, Hong Kong Special Administrative Region": "Hong Kong"
```

There are also several countries with numbers and/or parenthesis in their name. Be sure to remove these,

e.g.

`'Bolivia (Plurinational State of)'` should be `'Bolivia'`,

`'Switzerland17'` should be `'Switzerland'`.

Next, load the GDP data from the file `world_bank.csv`, which is a csv containing countries' GDP from 1960 to 2015 from World Bank. Call this DataFrame **GDP**.

Make sure to skip the header, and rename the following list of countries:

```
"Korea, Rep.": "South Korea",
"Iran, Islamic Rep.": "Iran",
"Hong Kong SAR, China": "Hong Kong"
```

Finally, load the Sciamgo Journal and Country Rank data for Energy Engineering and Power Technology from the file `scimagojr-3.xlsx`, which ranks countries based on their journal contributions in the aforementioned area. Call this DataFrame **ScimEn**.

Join the three datasets: GDP, Energy, and ScimEn into a new dataset (using the intersection of country names). Use only the last 10 years (2006-2015) of GDP data and only the top 15 countries by Scimagojr 'Rank' (Rank 1 through 15).

The index of this DataFrame should be the name of the country, and the columns should be ['Rank', 'Documents', 'Citable documents', 'Citations', 'Self-citations', 'Citations per document', 'H index', 'Energy Supply', 'Energy Supply per Capita', '% Renewable', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014', '2015'].

This function should return a DataFrame with 20 columns and 15 entries.

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

def answer_one():
    energy = pd.read_excel('Energy Indicators.xls')
    energy = energy[16:243]
    energy.drop(energy.columns[0:2],axis=1, inplace = True)
    energy.columns=['Country', 'Energy Supply','Energy Supply per Capita', '% Renewable']
    energy = energy.replace('...', np.NaN)
    energy['Energy Supply'] = energy['Energy Supply']*1000000
    energy['Country'] = energy['Country'].str.replace(r" \(.*\)","")
    energy['Country'] = energy['Country'].str.replace(r"([0-9]+)$","")
    energy['Country'].replace({"Republic of Korea": "South Korea",
                   "United States of America": "United States",
                    "United Kingdom of Great Britain and Northern Ireland": "United Kingdom",
                    "China, Hong Kong Special Administrative Region": "Hong Kong"}, inplace=True)
    energy.reset_index()
    energy = energy.set_index('Country')
    #energy

    GDP =pd.read_csv('world_bank.csv', skiprows =4)
    GDP['Country Name'].replace({"Korea, Rep.": "South Korea",
                                 "Iran, Islamic Rep.": "Iran",
                                 "Hong Kong SAR, China": "Hong Kong"}, inplace=True)
    GDP = GDP[np.append(['Country Name'] ,np.arange(2006,2016).astype(str))]
    GDP.reset_index()
    GDP = GDP.rename(columns={'Country Name': 'Country'})
    GDP = GDP.set_index('Country')
    #GDP

    ScimEn = pd.read_excel('scimagojr-3.xlsx')
    ScimEn.reset_index()
    ScimEn = ScimEn.set_index('Country')
    #ScimEn



    energyGDP = pd.merge(energy, GDP, how='outer', left_index=True, right_index=True)
    df = pd.merge(ScimEn, energyGDP, how='outer', left_index=True, right_index=True)
    df = df.reset_index().dropna(thresh=df.shape[1]-10).set_index('Country')
    df.sort_index(by= 'Rank', ascending = True, inplace=True)
    df = df.loc[df['Rank']<=15]
    df.sort_index(axis = 1)
    return df
answer_one()
```

Out[1]:

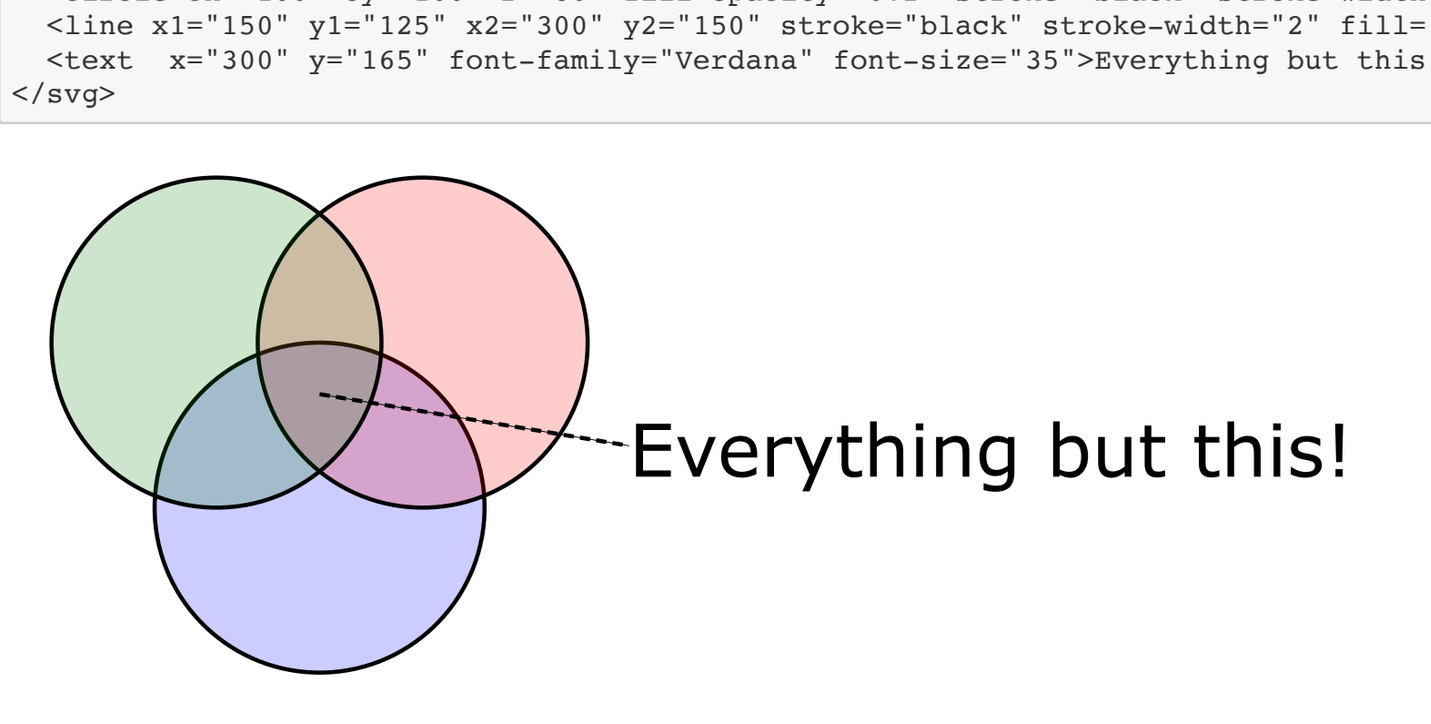
	Rank	Documents	Citable documents	Citations	Self-citations	Citations per document	H index	Energy Supply	Energy Supply per Capita	% Renewable	2006	2007
Country												
China	1.0	127050.0	126767.0	597237.0	411683.0	4.70	138.0	1.271910e+11	93.0	19.754910	3.992331e+12	4.559041e+
United States	2.0	96661.0	94747.0	792274.0	265436.0	8.20	230.0	9.083800e+10	286.0	11.570980	1.479230e+13	1.505540e+
Japan	3.0	30504.0	30287.0	223024.0	61554.0	7.31	134.0	1.898400e+10	149.0	10.232820	5.496542e+12	5.617036e+
United Kingdom	4.0	20944.0	20357.0	206091.0	37874.0	9.84	139.0	7.920000e+09	124.0	10.600470	2.419631e+12	2.482203e+
Russian Federation	5.0	18534.0	18301.0	34266.0	12422.0	1.85	57.0	3.070900e+10	214.0	17.288680	1.385793e+12	1.504071e+
Canada	6.0	17899.0	17620.0	215003.0	40930.0	12.01	149.0	1.043100e+10	296.0	61.945430	1.564469e+12	1.596740e+
Germany	7.0	17027.0	16831.0	140566.0	27426.0	8.26	126.0	1.326100e+10	165.0	17.901530	3.332891e+12	3.441561e+
India	8.0	15005.0	14841.0	128763.0	37209.0	8.58	115.0	3.319500e+10	26.0	14.969080	1.265894e+12	1.374865e+
France	9.0	13153.0	12973.0	130632.0	28601.0	9.93	114.0	1.059700e+10	166.0	17.020280	2.607840e+12	2.669424e+
South Korea	10.0	11983.0	11923.0	114675.0	22595.0	9.57	104.0	1.100700e+10	221.0	2.279353	9.410199e+11	9.924316e+
Italy	11.0	10964.0	10794.0	111850.0	26661.0	10.20	106.0	6.530000e+09	109.0	33.667230	2.202170e+12	2.234627e+
Spain	12.0	9428.0	9330.0	123336.0	23964.0	13.08	115.0	4.923000e+09	106.0	37.968590	1.414823e+12	1.468146e+
Iran	13.0	8896.0	8819.0	57470.0	19125.0	6.46	72.0	9.172000e+09	119.0	5.707721	3.895523e+11	4.250646e+
Australia	14.0	8831.0	8725.0	90765.0	15606.0	10.28	107.0	5.386000e+09	231.0	11.810810	1.021939e+12	1.060340e+
Brazil	15.0	8668.0	8596.0	60702.0	14396.0	7.00	86.0	1.214900e+10	59.0	69.648030	1.845080e+12	1.957118e+

Question 2 (6.6%)

The previous question joined three datasets then reduced this to just the top 15 entries. When you joined the datasets, but before you reduced this to the top 15 items, how many entries did you lose?

This function should return a single number.

In [2]:
```
%%HTML
<svg width="800" height="300">
  <circle cx="150" cy="180" r="80" fill-opacity="0.2" stroke="black" stroke-width="2" fill="blue" />
  <circle cx="200" cy="100" r="80" fill-opacity="0.2" stroke="black" stroke-width="2" fill="red" />
  <circle cx="100" cy="100" r="80" fill-opacity="0.2" stroke="black" stroke-width="2" fill="green" />
  <line x1="150" y1="125" x2="300" y2="150" stroke="black" stroke-width="2" fill="black" stroke-dasharray="5,3"/>
  <text  x="300" y="165" font-family="Verdana" font-size="35">Everything but this!</text>
</svg>
```



In [3]:
```python
import pandas as pd
import numpy as np

def answer_two():
    energy = pd.read_excel('Energy Indicators.xls')
    energy = energy[16:243]
    energy.drop(energy.columns[0:2],axis=1, inplace = True)
    energy.columns=['Country', 'Energy Supply','Energy Supply per Capita', '% Renewable']
    energy = energy.replace('...', np.NaN)
    energy['Energy Supply'] = energy['Energy Supply']*1000000
    energy['Country'] = energy['Country'].str.replace(r" \(.*\)","")
    energy['Country'] = energy['Country'].str.replace(r"([0-9]+)$","")
    energy['Country'].replace({"Republic of Korea": "South Korea",
                   "United States of America": "United States",
                    "United Kingdom of Great Britain and Northern Ireland": "United Kingdom",
                    "China, Hong Kong Special Administrative Region": "Hong Kong"}, inplace=True)
    energy.reset_index()
    energy = energy.set_index('Country')
    #energy

    GDP =pd.read_csv('world_bank.csv', skiprows =4)
    GDP['Country Name'].replace({"Korea, Rep.": "South Korea",
                                 "Iran, Islamic Rep.": "Iran",
                                 "Hong Kong SAR, China": "Hong Kong"}, inplace=True)
    GDP = GDP[np.append(['Country Name'] ,np.arange(2006,2016).astype(str))]
    GDP.reset_index()
    GDP = GDP.rename(columns={'Country Name': 'Country'})
    GDP = GDP.set_index('Country')
    #GDP

    ScimEn = pd.read_excel('scimagojr-3.xlsx')
    ScimEn.reset_index()
    ScimEn = ScimEn.set_index('Country')
    #ScimEn

    energyGDP_outer = pd.merge(energy, GDP, how='outer', left_index=True, right_index=True)
    df_outer = pd.merge(ScimEn, energyGDP_outer, how='outer', left_index=True, right_index=True)

    energyGDP_inner = pd.merge(energy, GDP, how='inner', left_index=True, right_index=True)
    df_inner = pd.merge(ScimEn, energyGDP_inner, how='inner', left_index=True, right_index=True)


    #df = df.reset_index().dropna(thresh=df.shape[1]-10).set_index('Country')
    #df.sort_index(by= 'Rank', ascending = True, inplace=True)
    #df = df.loc[df['Rank']<=15]
    #df.sort_index(axis = 1)
    return len(df_outer) - len(df_inner)

answer_two()
```

Out[3]: 156

Answer the following questions in the context of only the top 15 countries by Scimagojr Rank (aka the DataFrame returned by `answer_one()`)

Question 3 (6.6%)

What is the average GDP over the last 10 years for each country? (exclude missing values from this calculation.)

This function should return a Series named `avgGDP` with 15 countries and their average GDP sorted in descending order.

In [20]:
```python
import pandas as pd
import numpy as np

def answer_three():
    import numpy as np
    Top15 = answer_one()
    years_to_keep = np.arange(2006, 2016).astype(str)
    Top15['avgGDP'] = Top15[years_to_keep].mean(axis=1)
    return Top15['avgGDP'].sort_values(ascending=False)

answer_three()
```

Out[20]:
```
Country
United States         1.536434e+13
China                 6.348609e+12
Japan                 5.542208e+12
Germany               3.493025e+12
France                2.681725e+12
United Kingdom        2.487907e+12
Brazil                2.189794e+12
Italy                 2.120175e+12
India                 1.769297e+12
Canada                1.660647e+12
Russian Federation    1.565459e+12
Spain                 1.418078e+12
Australia             1.164043e+12
South Korea           1.106715e+12
Iran                  4.441558e+11
Name: avgGDP, dtype: float64
```

Question 4 (6.6%)

By how much had the GDP changed over the 10 year span for the country with the 6th largest average GDP?

This function should return a single number.

In [5]:
```python
def answer_four():
    import numpy as np
    Top15 = answer_one()
    years_to_keep = np.arange(2006, 2016).astype(str)
    Top15['avgGDP'] = Top15[years_to_keep].mean(axis=1)
    Top15 = Top15.sort_values(['avgGDP'], ascending=False)
    Top15['deltaGDP'] = Top15['2015'] - Top15['2006']
    Top15 = Top15.reset_index()
    return Top15.loc[5, 'deltaGDP']
answer_four()
```

Out[5]: 246702696075.3999

Question 5 (6.6%)

What is the mean `Energy Supply per Capita`?

This function should return a single number.

In [6]:
```python
def answer_five():
    Top15 = answer_one()
    mESpC = Top15['Energy Supply per Capita'].mean(axis=0)
    return float(round(mESpC,2))
answer_five()
```

Out[6]: 157.6

Question 6 (6.6%)

What country has the maximum % Renewable and what is the percentage?

This function should return a tuple with the name of the country and the percentage.

In [7]:
```python
import numpy as np
def answer_six():
    Top15 = answer_one()
    result = Top15['% Renewable'].argmax()
    return (result, Top15.loc[result,'% Renewable'])
answer_six()
```

Out[7]: ('Brazil', 69.648030000000006)

Question 7 (6.6%)

Create a new column that is the ratio of Self-Citations to Total Citations. What is the maximum value for this new column, and what country has the highest ratio?

This function should return a tuple with the name of the country and the ratio.

In [8]:
```python
import numpy as np
def answer_seven():
    Top15 = answer_one()
    Top15['raito']= Top15['Self-citations']/ Top15['Citations']
    result = Top15['raito'].argmax()

    return (result, Top15.loc[result,'raito'])
answer_seven()
```

Out[8]: ('China', 0.68931261793894216)

Question 8 (6.6%)

Create a column that estimates the population using Energy Supply and Energy Supply per capita. What is the third most populous country according to this estimate?

This function should return a single string value.

In [9]:
```python
def answer_eight():
    Top15 = answer_one()
    Top15['population'] = Top15['Energy Supply']/Top15['Energy Supply per Capita']
    popula = Top15['population'].sort_values(ascending=False)
    return popula.index[2]
answer_eight()
```

Out[9]: 'United States'

Question 9 (6.6%)

Create a column that estimates the number of citable documents per person. What is the correlation between the number of citable documents per capita and the energy supply per capita? Use the `.corr()` method, (Pearson's correlation).

This function should return a single number.

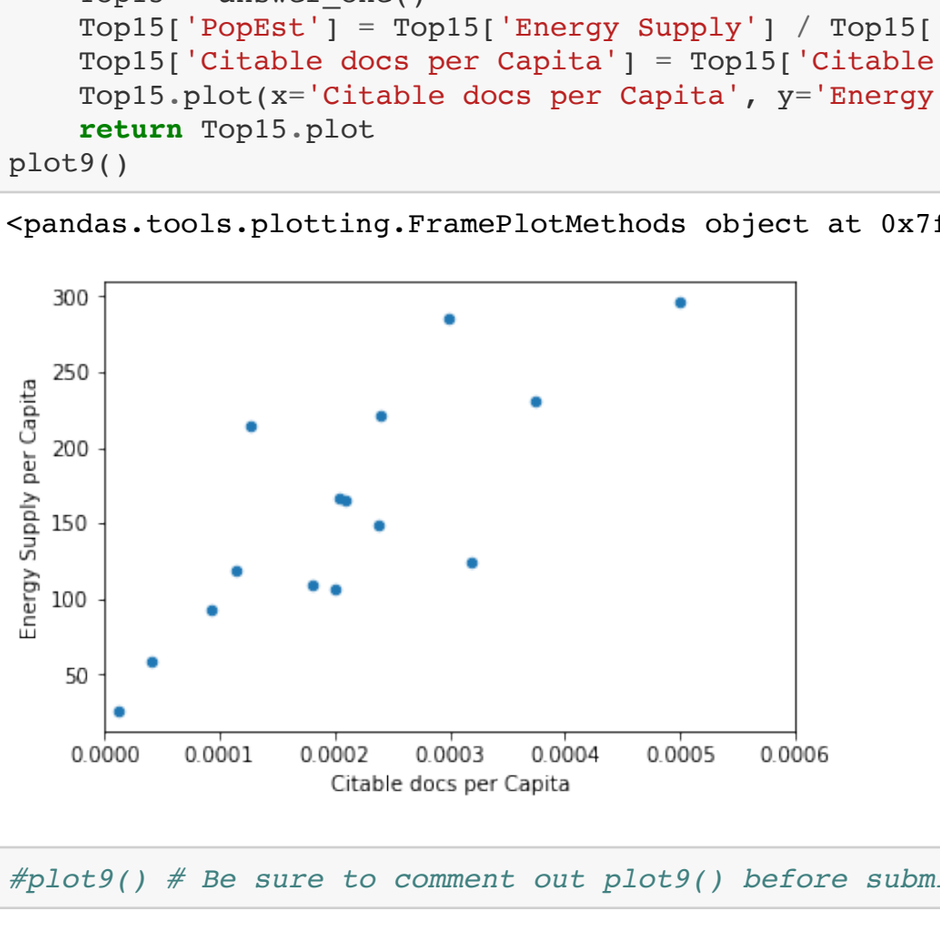
(Optional: Use the built-in function `plot9()` to visualize the relationship between Energy Supply per Capita vs. Citable docs per Capita)

In [10]:
```python
import numpy as np
def answer_nine():
    Top15 = answer_one()
    se= Top15['Citable documents']/ (Top15['Energy Supply']/Top15['Energy Supply per Capita'])
    ans = se.corr(Top15['Energy Supply per Capita'])
    return ans
answer_nine()
```

Out[10]: 0.79400104354429457

In [22]:
```python
def plot9():
    import matplotlib as plt
    %matplotlib inline

    Top15 = answer_one()
    Top15['PopEst'] = Top15['Energy Supply'] / Top15['Energy Supply per Capita']
    Top15['Citable docs per Capita'] = Top15['Citable documents'] / Top15['PopEst']
    Top15.plot(x='Citable docs per Capita', y='Energy Supply per Capita', kind='scatter', xlim=[0, 0.0006])
    return Top15.plot
plot9()
```

Out[22]: <pandas.tools.plotting.FramePlotMethods object at 0x7f2b0d9df320>



In [ ]:
```python
#plot9() # Be sure to comment out plot9() before submitting the assignment!
```

Question 10 (6.6%)

Create a new column with a 1 if the country's % Renewable value is at or above the median for all countries in the top 15, and a 0 if the country's % Renewable value is below the median.

This function should return a series named `HighRenew` whose index is the country name sorted in ascending order of rank.

In [19]:
```python
import numpy as np

#def answer_ten():
    #Top15 = answer_one()
    #medavg = Top15['% Renewable'].median(axis=0)
    #Top15['HighRenew'] = None
    #for i in range(len(Top15)):
        #if Top15['% Renewable'][i] >= medavg:
             #Top15['HighRenew'][i] = 1
        #else:
             #Top15['HighRenew'][i] = 0

   #Top15.sort_values(['Rank'], ascending= True)
    #
    #return Top15['HighRenew']
#answer_ten()
def answer_ten():
    import numpy as np
    Top15 = answer_one()
    Top15['median % Renewable'] = Top15['% Renewable'].median()
    Top15['HighRenew'] = Top15['% Renewable'] >= Top15['median % Renewable']
    return Top15['HighRenew'].sort_values(ascending=True)

answer_ten()
```

Out[19]:
```
Country
United States         False
Japan                 False
United Kingdom        False
India                 False
South Korea           False
Iran                  False
Australia             False
China                  True
Russian Federation     True
Canada                 True
Germany                True
France                 True
Italy                  True
Spain                  True
Brazil                 True
Name: HighRenew, dtype: bool
```

Question 11 (6.6%)

Use the following dictionary to group the Countries by Continent, then create a dateframe that displays the sample size (the number of countries in each continent bin), and the sum, mean, and std deviation for the estimated population of each country.

```python
ContinentDict  = {'China':'Asia',
                  'United States':'North America',
                  'Japan':'Asia',
                  'United Kingdom':'Europe',
                  'Russian Federation':'Europe',
                  'Canada':'North America',
                  'Germany':'Europe',
                  'India':'Asia',
                  'France':'Europe',
                  'South Korea':'Asia',
                  'Italy':'Europe',
                  'Spain':'Europe',
                  'Iran':'Asia',
                  'Australia':'Australia',
                  'Brazil':'South America'}
```

This function should return a DataFrame with index named Continent `['Asia', 'Australia', 'Europe', 'North America', 'South America']` and columns `['size', 'sum', 'mean', 'std']`

In [13]:
```python
import numpy as np
import pandas as pd

def answer_eleven():
    Top15 = answer_one()
    ContinentDict= {'China':'Asia',
                  'United States':'North America',
                  'Japan':'Asia',
                  'United Kingdom':'Europe',
                  'Russian Federation':'Europe',
                  'Canada':'North America',
                  'Germany':'Europe',
                  'India':'Asia',
                  'France':'Europe',
                  'South Korea':'Asia',
                  'Italy':'Europe',
                  'Spain':'Europe',
                  'Iran':'Asia',
                  'Australia':'Australia',
                  'Brazil':'South America'}
    Top15= Top15.reset_index()
    Top15['Continent'] = Top15['Country'].map(ContinentDict)
    Top15['Estimate Population'] = Top15['Energy Supply'] / Top15['Energy Supply per Capita']
    Top15 = Top15[['Continent', 'Estimate Population']]


    Top15 = Top15.groupby('Continent')['Estimate Population'].agg({'size': np.size,'sum': np.sum,'mean': np.mean,'std
': np.std})

    return Top15

answer_eleven()
```

Out[13]:

	size	sum	mean	std
Continent				
Asia	5.0	2.898666e+09	5.797333e+08	6.790979e+08
Australia	1.0	2.331602e+07	2.331602e+07	NaN
Europe	6.0	4.579297e+08	7.632161e+07	3.464767e+07
North America	2.0	3.528552e+08	1.764276e+08	1.996696e+08
South America	1.0	2.059153e+08	2.059153e+08	NaN

Question 12 (6.6%)

Cut % Renewable into 5 bins. Group Top15 by the Continent, as well as these new % Renewable bins. How many countries are in each of these groups?

*This function should return a **Series** with a MultiIndex of `Continent`, then the bins for `% Renewable`. Do not include groups with no countries.*

In [21]:
```python
import numpy as np
import pandas as pd

def answer_twelve():
    import numpy as np
    import pandas as pd
    Top15 = answer_one()
    ContinentDict  = {'China':'Asia',
                  'United States':'North America',
                  'Japan':'Asia',
                  'United Kingdom':'Europe',
                  'Russian Federation':'Europe',
                  'Canada':'North America',
                  'Germany':'Europe',
                  'India':'Asia',
                  'France':'Europe',
                  'South Korea':'Asia',
                  'Italy':'Europe',
                  'Spain':'Europe',
                  'Iran':'Asia',
                  'Australia':'Australia',
                  'Brazil':'South America'}
    Top15 = Top15.reset_index()
    Top15['Continent'] = Top15['Country'].map(ContinentDict)
    Top15['% Renewable'] = pd.cut(Top15['% Renewable'], 5)
    result = Top15.groupby(['Continent', '% Renewable'])['Country'].count()
    result = result.reset_index()
    #result.drop('Country', axis=1, inplace=True)

    result = result.set_index(['Continent', '% Renewable'])
    return result['Country']

answer_twelve()
```

Out[21]:
```
Continent      % Renewable
Asia           (2.212, 15.753]     4
               (15.753, 29.227]    1
Australia      (2.212, 15.753]     1
Europe         (2.212, 15.753]     1
               (15.753, 29.227]    3
               (29.227, 42.701]    2
North America  (2.212, 15.753]     1
               (56.174, 69.648]    1
South America  (56.174, 69.648]    1
Name: Country, dtype: int64
```

Question 13 (6.6%)

Convert the Population Estimate series to a string with thousands separator (using commas). Do not round the results.

e.g. 317615384.61538464 -> 317,615,384.61538464

This function should return a Series `PopEst` whose index is the country name and whose values are the population estimate string.

In [15]:
```python
import numpy as np
def answer_thirteen():
    Top15 = answer_one()
    Top15['PopEst'] = Top15['Energy Supply'] / Top15['Energy Supply per Capita']
    Top15['PopEst'] = Top15['PopEst'].apply('{:,}'.format)
    return Top15['PopEst']
answer_thirteen()
```

Out[15]:
```
Country
China                 1,367,645,161.2903225
United States          317,615,384.61538464
Japan                  127,409,395.97315437
United Kingdom         63,870,967.741935484
Russian Federation            143,500,000.0
Canada                  35,239,864.86486486
Germany                 80,369,696.96969697
India                 1,276,730,769.2307692
France                  63,837,349.39759036
South Korea            49,805,429.864253394
Italy                  59,908,256.880733944
Spain                    46,443,396.2264151
Iran                    77,075,630.25210084
Australia              23,316,017.316017315
Brazil                 205,915,254.23728815
Name: PopEst, dtype: object
```

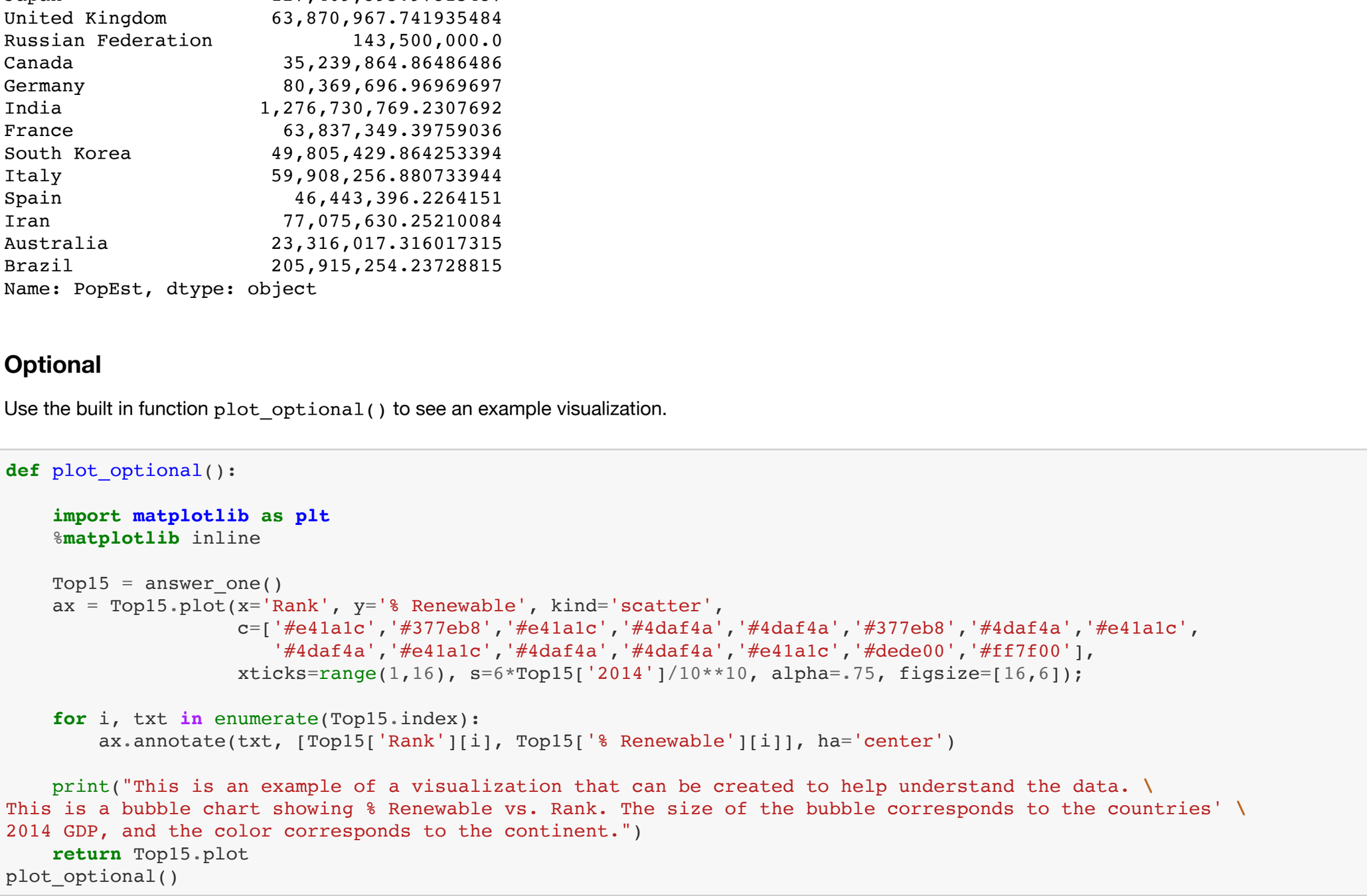
Optional

Use the built in function `plot_optional()` to see an example visualization.

In [23]:
```python
def plot_optional():

    import matplotlib as plt
    %matplotlib inline

    Top15 = answer_one()
    ax = Top15.plot(x='Rank', y='% Renewable', kind='scatter',
                    c=['#e41a1c','#377eb8','#e41a1c','#4daf4a','#4daf4a','#377eb8','#4daf4a','#e41a1c',
                       '#4daf4a','#e41a1c','#4daf4a','#4daf4a','#e41a1c','#dede00','#ff7f00'],
                    xticks=range(1,16), s=6*Top15['2014']/10**10, alpha=.75, figsize=[16,6]);

    for i, txt in enumerate(Top15.index):
        ax.annotate(txt, [Top15['Rank'][i], Top15['% Renewable'][i]], ha='center')

    print("This is an example of a visualization that can be created to help understand the data. \
This is a bubble chart showing % Renewable vs. Rank. The size of the bubble corresponds to the countries' \
2014 GDP, and the color corresponds to the continent.")
    return Top15.plot
plot_optional()
```

```
This is an example of a visualization that can be created to help understand the data. This is a bubble chart showin
g % Renewable vs. Rank. The size of the bubble corresponds to the countries' 2014 GDP, and the color corresponds to
the continent.
```

Out[23]: <pandas.tools.plotting.FramePlotMethods object at 0x7f2b13484cf8>



In [ ]:
```python
#plot_optional() # Be sure to comment out plot_optional() before submitting the assignment!
```