assignment3

May 4, 2021

1 Assignment 3

All questions are weighted the same in this assignment. 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. All questions are worth the same number of points except question 1 which is worth 17% of the assignment grade.

Note: Questions 2-13 rely on your question 1 answer.

[1]:
```
# Import labaries
import sys
import pandas as pd
import numpy as np

%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')

import warnings
warnings.filterwarnings('ignore')

print('You\'re running python %s' % sys.version.split(' ')[0])
```

```
You're running python 3.7.3
```

1.0.1 Question 1

Load the energy data from the file `assets/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 (**Note: 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 `assets/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 `assets/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, and the rows of the DataFrame should be sorted by "Rank".

[2]:
```
'''
@author:  Steven Ponce
Date:     22 April 2021                                                               ␣
↪                                                                                    ␣
↪
'''
def answer_one():
    # YOUR CODE HERE
    #raise NotImplementedError()

    # Loading the dataset and save DataFrame with the variable name of Energy
    # Droping the first two columns and renaming the remaining columns
    Energy = pd.read_excel('assets/Energy Indicators.xls',
                           na_values=["..."],
                           header = None, skiprows=18, skipfooter= 38,
                           usecols=[2,3,4,5],
                           names=['Country', 'Energy Supply', 'Energy Supply␣
↪per Capita', '% Renewable'])

    # Convert Energy Supply to gigajoules (Note: there are 1,000,000 gigajoules␣
↪in a petajoule).
    Energy['Energy Supply'] = Energy['Energy Supply'].apply(lambda x: x *␣
↪1000000)
```

```
    # There are also several countries with numbers and/or parenthesis in their␣
↪name. Be sure to remove them.
    Energy['Country'] = Energy['Country'].str.replace(r" \(.*\)","")
    Energy['Country'] = Energy['Country'].str.replace(r"\d*","")

    # Rename the following list of countries:                                 ␣
↪
    Energy['Country'] = 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'})

    # Next, load the GDP data from the file assets/world_bank.csv. Call this␣
↪DataFrame GDP.
    GDP = pd.read_csv('assets/world_bank.csv', skiprows=4)

    # Rename the following list of countries:
    GDP['Country Name'] = GDP['Country Name'].replace({'Korea, Rep.': 'South␣
↪Korea',
                                                      'Iran, Islamic Rep.':␣
↪'Iran',
                                                      'Hong Kong SAR, China' :␣
↪'Hong Kong'})


    # Finally, load the file assets/scimagojr-3.xlsx, which ranks countries␣
↪based on their journal contributions in
    # the aforementioned area. Call this DataFrame ScimEn.
    ScimEn = pd.read_excel('assets/scimagojr-3.xlsx')

    """
    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, and␣
↪the rows of the DataFrame
    should be sorted by "Rank".

    OUTPUT: This function should return a DataFrame with 20 columns and 15␣
↪entries, and the rows of the
    DataFrame should be sorted by "Rank".
    """

    # Join the three datasets: GDP, Energy, and ScimEn into a new dataset
    Energy_ScimEn = pd.
↪merge(ScimEn,Energy,how='inner',left_on='Country',right_on='Country')
    Energy_ScimEn = Energy_ScimEn[Energy_ScimEn['Rank']<=15]

    GDP.rename(columns = {'Country Name':'Country'},inplace=True)
    GDP = GDP.loc[:,['2006', '2007', '2008', '2009', '2010', '2011', '2012',␣
↪'2013', '2014', '2015','Country']]

    Energy_ScimEn_GDP = pd.
↪merge(Energy_ScimEn,GDP,how="inner",left_on="Country",right_on="Country").
↪set_index('Country')

    return Energy_ScimEn_GDP
```

[3]:
```
answer_one().shape
```

[3]:
```
(15, 20)
```

[4]:
```
answer_one()
```

[4]:
```
                     Rank  Documents  Citable documents  Citations  \
Country
China                   1     127050             126767     597237
United States           2      96661              94747     792274
Japan                   3      30504              30287     223024
United Kingdom          4      20944              20357     206091
Russian Federation      5      18534              18301      34266
Canada                  6      17899              17620     215003
Germany                 7      17027              16831     140566
India                   8      15005              14841     128763
France                  9      13153              12973     130632
South Korea            10      11983              11923     114675
```

Italy	11	10964	10794	111850
Spain	12	9428	9330	123336
Iran	13	8896	8819	57470
Australia	14	8831	8725	90765
Brazil	15	8668	8596	60702

	Self-citations	Citations per document	H index	\
Country				
China	411683	4.70	138	
United States	265436	8.20	230	
Japan	61554	7.31	134	
United Kingdom	37874	9.84	139	
Russian Federation	12422	1.85	57	
Canada	40930	12.01	149	
Germany	27426	8.26	126	
India	37209	8.58	115	
France	28601	9.93	114	
South Korea	22595	9.57	104	
Italy	26661	10.20	106	
Spain	23964	13.08	115	
Iran	19125	6.46	72	
Australia	15606	10.28	107	
Brazil	14396	7.00	86	

	Energy Supply	Energy Supply per Capita	% Renewable	\
Country				
China	1.271910e+11	93.0	19.754910	
United States	9.083800e+10	286.0	11.570980	
Japan	1.898400e+10	149.0	10.232820	
United Kingdom	7.920000e+09	124.0	10.600470	
Russian Federation	3.070900e+10	214.0	17.288680	
Canada	1.043100e+10	296.0	61.945430	
Germany	1.326100e+10	165.0	17.901530	
India	3.319500e+10	26.0	14.969080	
France	1.059700e+10	166.0	17.020280	
South Korea	1.100700e+10	221.0	2.279353	
Italy	6.530000e+09	109.0	33.667230	
Spain	4.923000e+09	106.0	37.968590	
Iran	9.172000e+09	119.0	5.707721	
Australia	5.386000e+09	231.0	11.810810	
Brazil	1.214900e+10	59.0	69.648030	

	2006	2007	2008	2009	\
Country					
China	3.992331e+12	4.559041e+12	4.997775e+12	5.459247e+12	
United States	1.479230e+13	1.505540e+13	1.501149e+13	1.459484e+13	
Japan	5.496542e+12	5.617036e+12	5.558527e+12	5.251308e+12	

```
United Kingdom      2.419631e+12  2.482203e+12  2.470614e+12  2.367048e+12
Russian Federation  1.385793e+12  1.504071e+12  1.583004e+12  1.459199e+12
Canada              1.564469e+12  1.596740e+12  1.612713e+12  1.565145e+12
Germany             3.332891e+12  3.441561e+12  3.478809e+12  3.283340e+12
India               1.265894e+12  1.374865e+12  1.428361e+12  1.549483e+12
France              2.607840e+12  2.669424e+12  2.674637e+12  2.595967e+12
South Korea         9.410199e+11  9.924316e+11  1.020510e+12  1.027730e+12
Italy               2.202170e+12  2.234627e+12  2.211154e+12  2.089938e+12
Spain               1.414823e+12  1.468146e+12  1.484530e+12  1.431475e+12
Iran                3.895523e+11  4.250646e+11  4.289909e+11  4.389208e+11
Australia           1.021939e+12  1.060340e+12  1.099644e+12  1.119654e+12
Brazil              1.845080e+12  1.957118e+12  2.056809e+12  2.054215e+12

                            2010          2011          2012          2013  \
Country
China               6.039659e+12  6.612490e+12  7.124978e+12  7.672448e+12
United States       1.496437e+13  1.520402e+13  1.554216e+13  1.577367e+13
Japan               5.498718e+12  5.473738e+12  5.569102e+12  5.644659e+12
United Kingdom      2.403504e+12  2.450911e+12  2.479809e+12  2.533370e+12
Russian Federation  1.524917e+12  1.589943e+12  1.645876e+12  1.666934e+12
Canada              1.613406e+12  1.664087e+12  1.693133e+12  1.730688e+12
Germany             3.417298e+12  3.542371e+12  3.556724e+12  3.567317e+12
India               1.708459e+12  1.821872e+12  1.924235e+12  2.051982e+12
France              2.646995e+12  2.702032e+12  2.706968e+12  2.722567e+12
South Korea         1.094499e+12  1.134796e+12  1.160809e+12  1.194429e+12
Italy               2.125185e+12  2.137439e+12  2.077184e+12  2.040871e+12
Spain               1.431673e+12  1.417355e+12  1.380216e+12  1.357139e+12
Iran                4.677902e+11  4.853309e+11  4.532569e+11  4.445926e+11
Australia           1.142251e+12  1.169431e+12  1.211913e+12  1.241484e+12
Brazil              2.208872e+12  2.295245e+12  2.339209e+12  2.409740e+12

                            2014          2015
Country
China               8.230121e+12  8.797999e+12
United States       1.615662e+13  1.654857e+13
Japan               5.642884e+12  5.669563e+12
United Kingdom      2.605643e+12  2.666333e+12
Russian Federation  1.678709e+12  1.616149e+12
Canada              1.773486e+12  1.792609e+12
Germany             3.624386e+12  3.685556e+12
India               2.200617e+12  2.367206e+12
France              2.729632e+12  2.761185e+12
South Korea         1.234340e+12  1.266580e+12
Italy               2.033868e+12  2.049316e+12
Spain               1.375605e+12  1.419821e+12
Iran                4.639027e+11           NaN
Australia           1.272520e+12  1.301251e+12
```

```
Brazil              2.412231e+12  2.319423e+12
```

[5]:
```
assert type(answer_one()) == pd.DataFrame, "Q1: You should return a DataFrame!"

assert answer_one().shape == (15,20), "Q1: Your DataFrame should have 20␣
↪columns and 15 entries!"
```

[6]:
```
# Cell for autograder.
```

1.0.2 Question 2

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.

[7]:
```
%%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>
```

```
<IPython.core.display.HTML object>
```

[8]:
```
'''
@author:  Steven Ponce
Date:     22 April 2021                                                      ␣
↪                                                                           ␣
↪
'''
def answer_two():
    # YOUR CODE HERE
    #raise NotImplementedError()

    # Loading the dataset and save DataFrame with the variable name of Energy
    # Droping the first two columns and renaming the remaining columns
    Energy = pd.read_excel('assets/Energy Indicators.xls',
                           na_values=["..."], header = None, skiprows=18,␣
↪skipfooter= 38,
```

```
                             usecols=[2,3,4,5],
                             names=['Country', 'Energy Supply', 'Energy Supply␣
↪per Capita', '% Renewable'])

    # Convert Energy Supply to gigajoules (Note: there are 1,000,000 gigajoules␣
↪in a petajoule).
    Energy['Energy Supply'] = Energy['Energy Supply'].apply(lambda x: x *␣
↪1000000)

    # There are also several countries with numbers and/or parenthesis in their␣
↪name. Be sure to remove them.
    Energy['Country'] = Energy['Country'].str.replace(r" \(.*\)","")
    Energy['Country'] = Energy['Country'].str.replace(r"\d*","")

    # Rename the following list of countries:                                  ␣
↪
    Energy['Country'] = 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'})

    # Next, load the GDP data from the file assets/world_bank.csv. Call this␣
↪DataFrame GDP.
    GDP = pd.read_csv('assets/world_bank.csv', skiprows=4)

    # Rename the following list of countries:
    GDP['Country Name'] = GDP['Country Name'].replace({'Korea, Rep.': 'South␣
↪Korea',
                                                       'Iran, Islamic Rep.':␣
↪'Iran',
                                                       'Hong Kong SAR, China' :␣
↪'Hong Kong'})

    # Finally, load the file assets/scimagojr-3.xlsx, which ranks countries␣
↪based on their journal contributions in
    # the aforementioned area. Call this DataFrame ScimEn.
    ScimEn = pd.read_excel('assets/scimagojr-3.xlsx')

    # inner1 = Energy_ScimEn_1
    # inner2 = Energy_ScimEn_GDP_1
    # outer1 = Energy_ScimEn_2
    # outer2 = Energy_ScimEn_GDP_2
```

```
    # inner join
    Energy_ScimEn_1 = pd.
↪merge(ScimEn,Energy,how='inner',left_on='Country',right_on='Country')

    GDP.rename(columns = {'Country Name':'Country'},inplace=True)
    GDP = GDP.loc[:,['2006', '2007', '2008', '2009', '2010', '2011', '2012',␣
↪'2013', '2014', '2015','Country']]

    Energy_ScimEn_GDP_1 = pd.
↪merge(Energy_ScimEn_1,GDP,how='inner',left_on='Country',
                                  right_on='Country').set_index('Country')

    # outer join
    Energy_ScimEn_2 = pd.
↪merge(ScimEn,Energy,how='outer',left_on='Country',right_on='Country')
    Energy_ScimEn_GDP_2 = pd.
↪merge(Energy_ScimEn_2,GDP,how='outer',left_on='Country',
                                  right_on='Country').set_index('Country')

    # outer minus join
    return len(Energy_ScimEn_GDP_2)-len(Energy_ScimEn_GDP_1);
```

[9]:
```
answer_two()
```

[9]: 156

[10]:
```
assert type(answer_two()) == int, "Q2: You should return an int number!"
```

1.0.3 Question 3

What are the top 15 countries for average GDP over the last 10 years?

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

[11]:
```
'''
@author:  Steven Ponce
Date:     22 April 2021                                                    ␣
 ↪                                                                        ␣
 ↪
'''
def answer_three():
    # YOUR CODE HERE
    #raise NotImplementedError()

    df = answer_one()
    avgGDP = df[['2006','2007', '2008', '2009', '2010', '2011', '2012',
```

```
                    '2013', '2014', '2015']].mean(axis=1).rename('aveGDP').
↪sort_values(ascending=False)
    return avgGDP
```

[12]:
```
answer_three().shape
```

[12]: (15,)

[13]:
```
answer_three()
```

[13]:
```
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: aveGDP, dtype: float64
```

[14]:
```
assert type(answer_three()) == pd.Series, "Q3: You should return a Series!"
```

1.0.4 Question 4

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.

[15]:
```
'''
@author:  Steven Ponce
Date:     22 April 2021                                                        ␣
 ↪                                                                              ␣
 ↪
'''
def answer_four():
    # YOUR CODE HERE
    #raise NotImplementedError()

    df = answer_one()
    avgGDP = df[['2006','2007', '2008', '2009', '2010', '2011', '2012',
```

```
                      '2013', '2014', '2015']].mean(axis=1).rename('avgGDP').
↪sort_values(ascending=False)

    sixth_largest_avgGDP = df.iloc[3]['2015'] - df.iloc[3]['2006']

    return sixth_largest_avgGDP
```

[16]:
```
answer_four()
```

[16]: 246702696075.3999

[17]:
```
# Cell for autograder.
```

1.0.5 Question 5

What is the mean energy supply per capita?
This function should return a single number.

[18]:
```
'''
@author:  Steven Ponce
Date:     22 April 2021                                                    ␣
↪                                                                          ␣
↪
'''
def answer_five():
    # YOUR CODE HERE
    #raise NotImplementedError()

    df = answer_one()
    mean_energy_per_capita = df['Energy Supply per Capita'].mean()

    return mean_energy_per_capita
```

[19]:
```
answer_five()
```

[19]: 157.6

[20]:
```
# Cell for autograder.
```

1.0.6 Question 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.

[21]:
```
'''
@author:  Steven Ponce
Date:     22 April 2021                                                    ␣
↪                                                                          ␣
↪
'''
def answer_six():
```

```
    # YOUR CODE HERE
    #raise NotImplementedError()

    df = answer_one()
    country_max_renewable = df['% Renewable'].idxmax(), df['% Renewable'].max()

    return country_max_renewable
```

[22]:
```
answer_six()
```

[22]: ('Brazil', 69.64803)

[23]:
```
assert type(answer_six()) == tuple, "Q6: You should return a tuple!"

assert type(answer_six()[0]) == str, "Q6: The first element in your result␣
↪should be the name of the country!"
```

1.0.7 Question 7

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.

[24]:
```
'''
@author:  Steven Ponce
Date:     23 April 2021                                                    ␣
↪                                                                         ␣
↪
'''
def answer_seven():
    # YOUR CODE HERE
    #raise NotImplementedError()

    df = answer_one()
    df['Citation Ratio'] = df['Self-citations'] / df['Citations']
    max_citation_ratio = df['Citation Ratio'].idxmax(), df['Citation Ratio'].
↪max()

    return max_citation_ratio
```

[25]:
```
answer_seven()
```

[25]: ('China', 0.6893126179389422)

[26]:
```
assert type(answer_seven()) == tuple, "Q7: You should return a tuple!"

assert type(answer_seven()[0]) == str, "Q7: The first element in your result␣
↪should be the name of the country!"
```

1.0.8 Question 8

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 the name of the country

[27]:
```
'''
@author:  Steven Ponce
Date:     23 April 2021                                              ␣
↪                                                                   ␣
↪
'''
def answer_eight():
    # YOUR CODE HERE
    #raise NotImplementedError()

    df = answer_one()
    df['Population Estimate'] = df['Energy Supply'] / df['Energy Supply per␣
↪Capita']
    third_populous_country = df.sort_values(by='Population Estimate',␣
↪ascending=False).iloc[2].name

    return third_populous_country
```

[28]:
```
answer_eight()
```

[28]:
```
'United States'
```

[29]:
```
assert type(answer_eight()) == str, "Q8: You should return the name of the␣
↪country!"
```

1.0.9 Question 9

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)

[30]:
```
'''
@author:  Steven Ponce
Date:     23 April 2021                                              ␣
↪                                                                   ␣
↪
'''
def answer_nine():
    # YOUR CODE HERE
    # raise NotImplementedError()
```

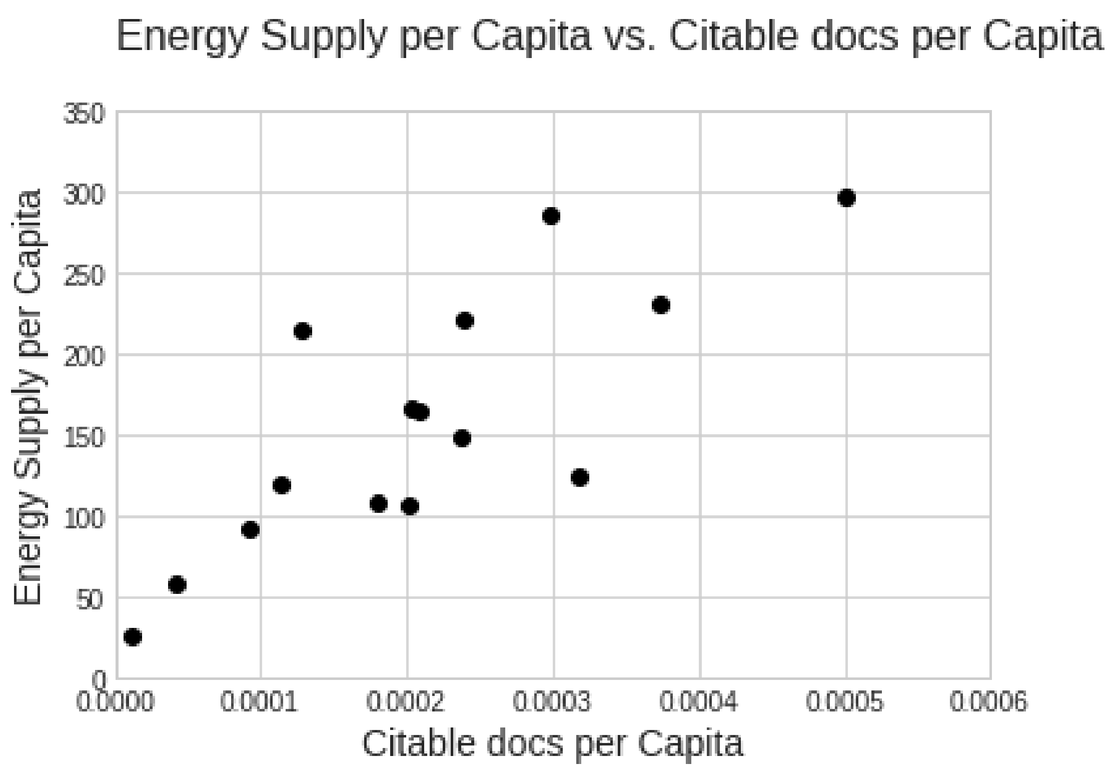
```
    df = answer_one()
    df = df.reset_index()
    df['Population Estimate'] = df['Energy Supply'] / df['Energy Supply per␣
 ↪Capita']
    df['Citable docs per Capita'] = df['Citable documents'] / df['Population␣
 ↪Estimate']

    correlation = df[['Energy Supply per Capita',
                      'Citable docs per Capita']].corr().ix['Energy Supply per␣
 ↪Capita', 'Citable docs per Capita']

    return correlation
```

[31]:
```
answer_nine()
```

[31]: 0.7940010435442943

[32]:
```
'''
@author:  Steven Ponce
Date:     23 April 2021                                                    ␣
 ↪                                                                        ␣
 ↪
'''
def plot_nine():

    df = answer_one()
    df['Population Estimate'] = df['Energy Supply'] / df['Energy Supply per␣
 ↪Capita']
    df['Citable docs per Capita'] = df['Citable documents'] / df['Population␣
 ↪Estimate']

    x_data = df['Citable docs per Capita']
    y_data = df['Energy Supply per Capita']

    plt.scatter(x_data, y_data, marker = 'o', color='black')

    title=('Energy Supply per Capita vs. Citable docs per Capita\n')
    plt.title(title, loc='left', fontsize=16)
    plt.xlabel('Citable docs per Capita', fontsize=14)
    plt.ylabel('Energy Supply per Capita',fontsize=14)
    plt.ylim(0, 350)
    plt.xlim(0.000, 0.0006)
```

[33]:
```
plot_nine()
```



[34]:
```
assert answer_nine() >= -1. and answer_nine() <= 1., "Q9: A valid correlation␣
↪should between -1 to 1!"
```

1.0.10 Question 10

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.*

[35]:
```
'''
@author:  Steven Ponce
Date:     23 April 2021                                                        ␣
↪                                                                              ␣
↪
'''
def answer_ten():
    # YOUR CODE HERE
    #raise NotImplementedError()

    df = answer_one()
    Median = df['% Renewable'].median()
```

```
    HighRenew = df['HighRenew']= df['% Renewable'].apply(lambda x:0 if x<Median␣
↪else 1 )

    return HighRenew
```

[36]:
```
answer_ten()
```

[36]:
```
Country
China                 1
United States         0
Japan                 0
United Kingdom        0
Russian Federation    1
Canada                1
Germany               1
India                 0
France                1
South Korea           0
Italy                 1
Spain                 1
Iran                  0
Australia             0
Brazil                1
Name: % Renewable, dtype: int64
```

[37]:
```
assert type(answer_ten()) == pd.Series, "Q10: You should return a Series!"
```

1.0.11 Question 11

Use the following dictionary to group the Countries by Continent, then create a DataFrame 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.

```
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']`

[38]:
```python
'''
@author:  Steven Ponce
Date:     23 April 2021
↪
↪
'''
def answer_eleven():
    # YOUR CODE HERE
    #raise NotImplementedError()

    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'}

    df = answer_one()
    df['Population Estimate'] = df['Energy Supply'] / df['Energy Supply per␣
↪Capita'].astype(float)

    stats = pd.DataFrame(columns = ['size', 'sum', 'mean', 'std'])

    for group, frame in df.groupby(ContinentDict):
        stats.loc[group] = [len(frame),
                            frame['Population Estimate'].sum(),
                            frame['Population Estimate'].mean(),
                            frame['Population Estimate'].std()]
    return stats
```

[39]:
```python
answer_eleven()
```

[39]:

	size	sum	mean	std
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
```

[40]:
```
assert type(answer_eleven()) == pd.DataFrame, "Q11: You should return a␣
↪DataFrame!"

assert answer_eleven().shape[0] == 5, "Q11: Wrong row numbers!"

assert answer_eleven().shape[1] == 4, "Q11: Wrong column numbers!"
```

1.0.12 Question 12

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.

[41]:
```
'''
@author:  Steven Ponce
Date:     23 April 2021                                                    ␣
↪                                                                         ␣
↪
'''
def answer_twelve():
    # YOUR CODE HERE
    # raise NotImplementedError()

    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'}

    df = answer_one()
    df['Continent'] = pd.Series(ContinentDict)
    df['% Renewable']=pd.cut(df['% Renewable'],5)
```

```
    output = df.groupby(['Continent','% Renewable'])['Continent'].agg(np.size).
↪dropna()

    return output
```

[42]:
```
answer_twelve()
```

[42]:
```
Continent      % Renewable
Asia           (2.212, 15.753]     4.0
               (15.753, 29.227]    1.0
Australia      (2.212, 15.753]     1.0
Europe         (2.212, 15.753]     1.0
               (15.753, 29.227]    3.0
               (29.227, 42.701]    2.0
North America  (2.212, 15.753]     1.0
               (56.174, 69.648]    1.0
South America  (56.174, 69.648]    1.0
Name: Continent, dtype: float64
```

[43]:
```
assert type(answer_twelve()) == pd.Series, "Q12: You should return a Series!"

assert len(answer_twelve()) == 9, "Q12: Wrong result numbers!"
```

1.0.13 Question 13

Convert the Population Estimate series to a string with thousands separator (using commas). Use all significant digits (do not round the results).

e.g. 12345678.90 -> 12,345,678.90

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

[44]:
```
'''
@author:  Steven Ponce
Date:     23 April 2021                                                      ␣
↪                                                                           ␣
↪
'''
def answer_thirteen():
    # YOUR CODE HERE
    # raise NotImplementedError()

    df = answer_one()
    df['Population Estimate'] = df['Energy Supply'] / df['Energy Supply per␣
↪Capita']
    PopEst = df['Population Estimate'].apply('{:,}'.format)

    return PopEst
```

[45]:
```
answer_thirteen()
```

[45]:
```
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: Population Estimate, dtype: object
```

[46]:
```
assert type(answer_thirteen()) == pd.Series, "Q13: You should return a Series!"

assert len(answer_thirteen()) == 15, "Q13: Wrong result numbers!"
```

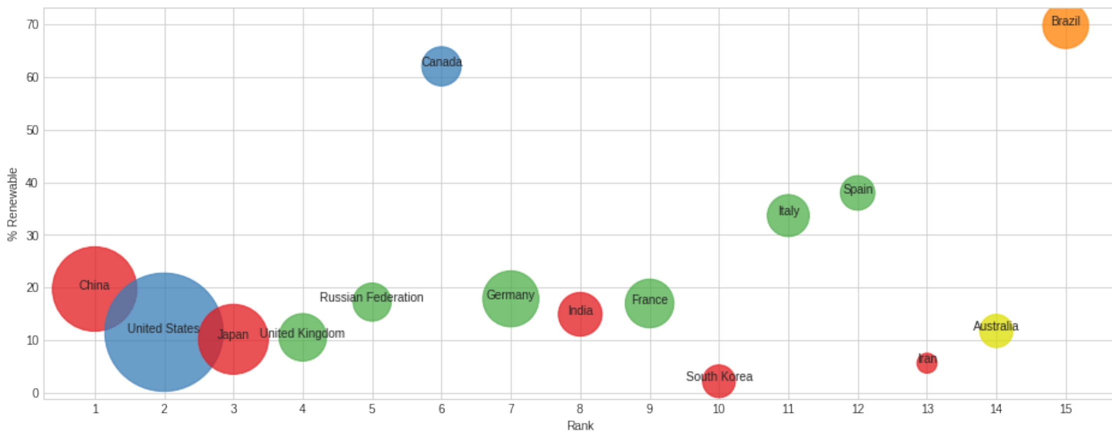
1.0.14 Optional

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

[47]:
```
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.")
```

[48]:
```
plot_optional()
```

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.



[ ]:
```
```