

# Assignment 3

January 2, 2020

---

*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.*

---

## 1 Assignment 3 - More Pandas

This assignment requires more individual learning than 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.

### 1.0.1 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 unnecessary, 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]: import pandas as pd
        energy = pd.read_excel("Energy Indicators.xls", header = 15)
        energy = energy.iloc[1:228]

In [2]: energy = energy[['Unnamed: 2', 'Energy Supply', 'Energy Supply per capita', 'Renewable Elec

In [3]: energy = energy.rename(columns = {'Unnamed: 2' : 'Country', 'Energy Supply' : 'Energy Sup

In [4]: def energy_conv(row):
        row['Energy Supply'] = row['Energy Supply']*1000000
        return row

In [5]: energy[['Energy Supply', 'Energy Supply per Capita', '% Renewable']] = energy[['Energy S

In [6]: energy = energy.apply(energy_conv,axis=1)

In [7]: import re

In [8]: def charac_change(row):
        country = row['Country']
        country = re.sub(' \((.*?)\)', '', country)
        country = re.sub('\d', '', country)
        row['Country'] = country
        return row

In [9]: energy = energy.apply(charac_change,axis=1)

In [10]: energy.loc[energy["Country"]=="Republic of Korea","Country"]= "South Korea"
         energy.loc[energy["Country"]=="United States of America","Country"] = "United States"
         energy.loc[energy["Country"]=="United Kingdom of Great Britain and Northern Ireland","C
         energy.loc[energy["Country"]=="China, Hong Kong Special Administrative Region","Country

In [11]: GDP = pd.read_csv("world_bank.csv",header=4)
```

```

In [12]: GDP.loc[GDP["Country Name"]=="Korea, Rep.", "Country Name"] = "South Korea"
         GDP.loc[GDP["Country Name"]=="Iran, Islamic Rep.", "Country Name"] = "Iran"
         GDP.loc[GDP["Country Name"]=="Hong Kong SAR, China", "Country Name"] = "Hong Kong"

In [13]: ScimEn = pd.read_excel("scimagojr-3.xlsx")

In [14]: merge1 = pd.merge(ScimEn, energy, how='outer' , left_on = 'Country', right_on = 'Country')

In [15]: merge2 = pd.merge(merge1, GDP[['Country Name', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014', '2015']],
                           left_on = 'Country', right_on = 'Country Name', how='outer')

In [16]: answer1 = merge2[['Country', 'Rank', 'Documents', 'Citable documents', 'Citations', 'Self Citations', 'H Index']]

In [17]: answer1 = answer1.set_index('Country', drop=True)

In [18]: def answer_one():
         return answer1[answer1['Rank'] <= 15]

```

### 1.0.2 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 [19]: def answer_two():
         union = pd.merge(pd.merge(energy, GDP, left_on='Country', right_on = "Country Name"),
                           ScimEn, left_on='Country', right_on = "Country Name", how='outer')
         intersect = pd.merge(pd.merge(energy, GDP, left_on='Country', right_on = "Country Name"),
                               ScimEn, left_on='Country', right_on = "Country Name", how='inner')
         return len(union)-len(intersect)

```

## 1.1 Answer the following questions in the context of only the top 15 countries by Scimagojr Rank (aka the DataFrame returned by answer\_one())

### 1.1.1 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]: import numpy as np

In [21]: def avg_gdp(row):
         avg1 = np.mean(row[['2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014', '2015']])
         return pd.Series({'average GDP': avg1})

In [62]: def answer_three():
         Top15 = answer_one()
         Top15.apply(avg_gdp, axis=1).sort_values("average GDP", ascending=False)
         return Top15.apply(avg_gdp, axis=1).sort_values("average GDP", ascending=False).iloc[:, 0]

```

### 1.1.2 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 [23]: def answer_four():
         Top15 = answer_one()
         return Top15.loc['United Kingdom'][19] - Top15.loc['United Kingdom'][10]
```

### 1.1.3 Question 5 (6.6%)

What is the mean Energy Supply per Capita?

*This function should return a single number.*

```
In [24]: def answer_five():
         Top15 = answer_one()
         return np.mean(Top15['Energy Supply per Capita'])
```

### 1.1.4 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 [25]: def answer_six():
         Top15 = answer_one()
         answer6 = Top15.sort_values(by='% Renewable', ascending=False).iloc[0]
         return answer6.name, answer6['% Renewable']
```

### 1.1.5 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 [26]: def answer_seven():
         Top15 = answer_one()
         Top15['Ratio-citations'] = Top15['Self-citations'] / Top15['Citations']
         answer6 = Top15.sort_values(by='Ratio-citations', ascending=False).iloc[0]
         return (answer6.name, answer6['Ratio-citations'])
```

### 1.1.6 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 [27]: def answer_eight():
         Top15 = answer_one()
         Top15['Pop-estimate'] = Top15['Energy Supply'] / Top15['Energy Supply per Capita']
         answer8 = Top15.sort_values(by="Pop-estimate", ascending = False).iloc[2]
         return answer8.name
```

### 1.1.7 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 [28]: def answer_nine():
        Top15 = answer_one()
        Top15['citabledoc-per-capita'] = Top15['Citable documents']/(Top15['Energy Supply'])
        answer9 = Top15[['citabledoc-per-capita', 'Energy Supply per Capita']].corr(method =
        return answer9.iloc[0][1]
```

```
In [29]: 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'
```

```
In [30]: #plot9() # Be sure to comment out plot9() before submitting the assignment!
```

### 1.1.8 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 [31]: def answer_ten():
        Top15 = answer_one()
        mid = Top15['% Renewable'].median()
        Top15['HighRenew'] = Top15['% Renewable']>=mid
        Top15['HighRenew'] = Top15['HighRenew'].apply(lambda x:1 if x else 0)
        Top15.sort_values(by='Rank', inplace=True)
        return Top15['HighRenew']
```

### 1.1.9 Question 11 (6.6%)

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']*

```

In [32]: 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'};

```

```

In [33]: def continent(row):
    row['Continent'] = ContinentDict[row.name]
    return row

```

```

In [34]: def answer_eleven():
    Top15 = answer_one()
    Top15['Country'] = Top15.index
    Top15['Continent'] = None
    Top15['PopEst'] = Top15['Energy Supply'] / Top15['Energy Supply per Capita']
    Top15 = Top15.apply(continent, axis=1)
    bins = Top15[['Country', 'Continent']].groupby('Continent').agg({'Country': 'count'})
    x = Top15[['PopEst', 'Continent']].groupby('Continent').agg({'PopEst': 'sum'})
    y = Top15[['PopEst', 'Continent']].groupby('Continent').agg({'PopEst': 'mean'})
    z = Top15[['PopEst', 'Continent']].groupby('Continent').agg({'PopEst': 'std'})
    xy = pd.merge(x,y,left_index=True,right_index = True)

```

```

xyz = pd.merge(xy,z,left_index= True,right_index = True)
xyzb = pd.merge(bins,xyz,left_index= True,right_index = True)
xyzb.columns = ['size', 'sum', 'mean', 'std']
return xyzb

```

### 1.1.10 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 [63]: def answer_twelve():
        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'] = [ContinentDict[country] for country in Top15['Country']]
        Top15['bins'] = pd.cut(Top15['% Renewable'],5)
        return Top15.groupby(['Continent','bins']).size()

```

```

In [64]: answer_twelve()

```

```

Out[64]: Continent      bins
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
dtype: int64

```

```

In [81]: def answer_twelve():
        Top15 = answer_one()

```

```

Top15['bins'] = pd.cut(Top15['% Renewable'],5)
Top15['Continent'] = None
Top15['Country'] = Top15.index
Top15 = Top15.apply(continent, axis=1)
bins2 = Top15[['Country', 'Continent', 'bins']].groupby(['Continent', 'bins']).agg('count')
bins2.sort(['Continent', 'bins'])
return bins2.iloc[:,0]

```

```
In [ ]: df.sort(['Group1', 'C'], ascending=False)
```

### 1.1.11 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 [37]: def answer_thirteen():
        Top15 = answer_one()
        Top15['PopEst'] = (Top15['Energy Supply'] / Top15['Energy Supply per Capita']).astype(float)
        return Top15['PopEst'].apply(lambda x: '{0:,}'.format(x))

```

### 1.1.12 Optional

Use the built in function plot\_optional() to see an example visualization.

```

In [38]: 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',
                            '#4daf4a', '#e41a1c', '#4daf4a', '#4daf4a', '#e41a1c', '#d9d9d9', '#f0f0f0' ],
                        xticks=range(1,16), s=6*Top15['2014']/10**10, alpha=.75, figsize=[16,10])

        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 country's 2014 GDP, and the color corresponds to the continent.")

```

```
In [39]: #plot_optional() # Be sure to comment out plot_optional() before submitting the assignment
```