assignment3

June 12, 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 3-13 rely on your question 1 answer.

[1]:

```
import pandas as pd
import numpy as np
import re
import string

# Filter all warnings. If you would like to see the warnings, please comment␣
 ↪the two lines below.
import warnings
warnings.filterwarnings('ignore')
```

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".

[29]:

```
import pandas as pd
import numpy as np

def answer_one():
    # YOUR CODE HERE
    # loadenergy data
    Energy = pd.read_excel("assets/Energy Indicators.xls",na_values= ["...
 ↪"],header=None, skiprows=18,
                           skipfooter = 38,use_cols = [2,3,4,5],
                           names = ['Country', 'Energy Supply', 'Energy Supply␣
 ↪per Capita', '% Renewable'])



    # missing data (e.g. data with "...") make sure this is reflected as np.NaN␣
 ↪values
    #Energy = Energy.replace("...", np.nan)

    #Convert Energy Supply to gigajoules
    Energy["Energy Supply"] = Energy["Energy Supply"].apply(lambda x: x␣
 ↪*1000000)
    #remove_digit (There are also several countries with numbers and/or␣
 ↪parenthesis )
    def remove_numbers_parenthesis(data):
        if re.search(r'\d', data)!=None:
            data = data.strip(string.digits)
        if data.find("(")!=-1:
```

```
        data = data[:data.find("(")-1]
    return data
Energy["Country"] = Energy["Country"].apply(remove_numbers_parenthesis)
print(Energy['Country'])

# Rename some 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
# load GDP data
GDP = pd.read_csv('assets/world_bank.csv', skiprows=4)
# Rename Country Name column
GDP = GDP.rename(columns={"Country Name":"Country"})
# Rename some Countries
GDP["Country"] = GDP["Country"].replace({"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.
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".
    """
    #merge
    ScimEn_Energy = pd.
↪merge(ScimEn,Energy,how="inner",left_on="Country",right_on="Country")
    ScimEn_Energy = ScimEn_Energy[ScimEn_Energy["Rank"]<=15]

    GDP = GDP.loc[:,['2006', '2007', '2008', '2009', '2010', '2011', '2012',␣
↪'2013', '2014', '2015',"Country"]]
    ScimEn_Energy_GDP= pd.
↪merge(ScimEn_Energy,GDP,how="inner",left_on="Country",right_on="Country").
↪set_index("Country")
    print(ScimEn_Energy_GDP.shape)
    return ScimEn_Energy_GDP


    raise NotImplementedError()
```

[ ]:

```
```

[30]:

```
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!"
```

```
NaN  Afghanistan                                 Afghanistan
     Albania                                         Albania
     Algeria                                         Algeria
     American Samoa                           American Samoa
     Andorra                                         Andorra
                                                ...
     Viet Nam                                       Viet Nam
     Wallis and Futuna Islands    Wallis and Futuna Islands
     Yemen                                             Yemen
     Zambia                                           Zambia
     Zimbabwe                                       Zimbabwe
Name: Country, Length: 227, dtype: object
(15, 20)
NaN  Afghanistan                                 Afghanistan
     Albania                                         Albania
     Algeria                                         Algeria
     American Samoa                          American Samoa
     Andorra                                        Andorra
                                             ...
     Viet Nam                                      Viet Nam
     Wallis and Futuna Islands    Wallis and Futuna Islands
     Yemen                                            Yemen
     Zambia                                          Zambia
     Zimbabwe                                      Zimbabwe
Name: Country, Length: 227, dtype: object
(15, 20)
```

[92]:

```
# 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.

[ ]:

```
%%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>
```

[52]:

```
import pandas as pd
import numpy as np

def answer_two():
    # YOUR CODE HERE
    Energy = pd.read_excel("assets/Energy Indicators.xls",na_values= ["...
↪"],header=None, skiprows=18,
                           skipfooter = 38,use_cols = [2,3,4,5],
                           names = ['Country', 'Energy Supply', 'Energy Supply␣
↪per Capita', '% Renewable'])



    # missing data (e.g. data with "...") make sure this is reflected as np.NaN␣
↪values
```

```
    #Energy = Energy.replace("...", np.nan)

    #Convert Energy Supply to gigajoules
    Energy["Energy Supply"] = Energy["Energy Supply"].apply(lambda x: x␣
↪*1000000)

    #remove_digit (There are also several countries with numbers and/or␣
↪parenthesis )
    def remove_numbers_parenthesis(data):
        if re.search(r'\d', data)!=None:
            data = data.strip(string.digits)
        if data.find("(")!=-1:
            data = data[:data.find("(")-1]
        return data
    Energy["Country"] = Energy["Country"].apply(remove_numbers_parenthesis)
    print(Energy['Country'])

    # Rename some 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
    # load GDP data
    GDP = pd.read_csv('assets/world_bank.csv', skiprows=4)
    # Rename Country Name column
    GDP = GDP.rename(columns={"Country Name":"Country"})
    # Rename some Countries
    GDP["Country"] = GDP["Country"].replace({"Korea, Rep.": "South Korea",
                                                       "Iran, Islamic Rep.":␣
↪"Iran",
                                                       "Hong Kong SAR, China":␣
↪"Hong Kong"})

    ScimEn = pd.read_excel("assets/scimagojr-3.xlsx")

    df1 = pd.
↪merge(ScimEn,Energy,how="inner",left_on="Country",right_on="Country")
```

```
    GDP = GDP.loc[:,['2006', '2007', '2008', '2009', '2010', '2011', '2012',␣
↪'2013', '2014', '2015',"Country"]]
    df2 = pd.merge(df1,GDP,how="inner",left_on="Country",right_on="Country").
↪set_index("Country")
    df11 = pd.
↪merge(ScimEn,Energy,how="outer",left_on="Country",right_on="Country")
    df21 = pd.merge(df11,GDP,how="outer",left_on="Country",right_on="Country").
↪set_index("Country")

    return len(df21)-len(df2)

    raise NotImplementedError()
```

[53]:

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

```
NaN  Afghanistan                                     Afghanistan
     Albania                                             Albania
     Algeria                                             Algeria
     American Samoa                               American Samoa
     Andorra                                             Andorra
                                                  ...
     Viet Nam                                           Viet Nam
     Wallis and Futuna Islands         Wallis and Futuna Islands
     Yemen                                                 Yemen
     Zambia                                               Zambia
     Zimbabwe                                           Zimbabwe
Name: Country, Length: 227, dtype: object
```

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.

[54]:

```
import pandas as pd
import numpy as np

def answer_three():
    # YOUR CODE HERE
    df = answer_one()
    avgGDP =␣
↪df[["2006","2007","2008","2009","2010","2011","2012","2013","2014","2015"]].
↪mean(axis=1).rename('avgGDP').sort_values(ascending=False)
    return avgGDP

    raise NotImplementedError()
```

[55]:
```
assert type(answer_three()) == pd.Series, "Q3: You should return a Series!"
```

```
NaN  Afghanistan                                Afghanistan
     Albania                                        Albania
     Algeria                                        Algeria
     American Samoa                          American Samoa
     Andorra                                        Andorra
                                            ...
     Viet Nam                                      Viet Nam
     Wallis and Futuna Islands    Wallis and Futuna Islands
     Yemen                                            Yemen
     Zambia                                          Zambia
     Zimbabwe                                      Zimbabwe
Name: Country, Length: 227, dtype: object
(15, 20)
```

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.

[56]:
```
import pandas as pd
import numpy as np

def answer_four():
    # YOUR CODE HERE
    df = answer_one()
    return df.loc[answer_three().index[5]]["2015"]-df.loc[answer_three().
 ↪index[5]]["2006"]
    raise NotImplementedError()
```

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

1.0.5 Question 5

What is the mean energy supply per capita?

This function should return a single number.

[57]:
```
import pandas as pd
import numpy as np

def answer_five():
    # YOUR CODE HERE
    df = answer_one()
    return float(df["Energy Supply per Capita"].mean())
    raise NotImplementedError()
```

[ ]:
```
# 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.

[59]:

```
import pandas as pd
import numpy as np

def answer_six():
    # YOUR CODE HERE
    df = answer_one()
    return df['% Renewable'].idxmax(), df['% Renewable'].max()

    raise NotImplementedError()
```

[ ]:

```
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.

[60]:

```
import pandas as pd
import numpy as np

def answer_seven():
    # YOUR CODE HERE
    #idxmax() function returns index of first occurrence of maximum over␣
 ↪requested axis
    df = answer_one()
    df["Ratio"] = df["Self-citations"] / df["Citations"]
    return df["Ratio"].idxmax(), df["Ratio"].max()
    raise NotImplementedError()
```

[ ]:

```
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

```
[63]: import pandas as pd
      import numpy as np

      def answer_eight():
          # YOUR CODE HERE
          df = answer_one()
          df["Population"] = df['Energy Supply'] / df['Energy Supply per Capita']
          final = df.sort_values("Population", ascending=False)
          return final.iloc[2].name
          raise NotImplementedError()
```

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

```
NaN  Afghanistan                                Afghanistan
     Albania                                        Albania
     Algeria                                        Algeria
     American Samoa                          American Samoa
     Andorra                                        Andorra
                                            ...
     Viet Nam                                      Viet Nam
     Wallis and Futuna Islands    Wallis and Futuna Islands
     Yemen                                            Yemen
     Zambia                                          Zambia
     Zimbabwe                                      Zimbabwe
Name: Country, Length: 227, dtype: object
(15, 20)
```

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)

```
[65]: import pandas as pd
      import numpy as np

      def answer_nine():
          # YOUR CODE HERE
          df = answer_one()

          # calculate population estimation
          df["population"] = df["Energy Supply"] / df["Energy Supply per Capita"]
          df["Citable docts per capita"] = df["Citable documents"] / df["population"]
          return df["Citable docts per capita"].corr(df["Energy Supply per Capita"])
```

```
    raise NotImplementedError()
```

[ ]:
```
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])
```

[ ]:
```
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.

[66]:
```
import pandas as pd
import numpy as np

def answer_ten():
    # YOUR CODE HERE
    df = answer_one()
    df["HighRenew"]= df["% Renewable"].apply(lambda x:0 if x<(df["% Renewable"].
↪median()) else 1 )
    return df["HighRenew"]
    raise NotImplementedError()
```

[ ]:
```
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']`

[ ]:

```
import pandas as pd
import numpy as np

def answer_eleven():
    # YOUR CODE HERE
    df = 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'}

    df["population"] = df["Energy Supply"] / df["Energy Supply per Capita"]
    return df["population"].groupby(ContinentDict).agg(['size', 'sum', 'mean',␣
 ↪'std'])

    raise NotImplementedError()
```

[ ]:

```
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.*

[67]:

```
import pandas as pd
import numpy as np
import re


def answer_twelve():
    # YOUR CODE HERE
    df = 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'}

    df["bins"] = pd.cut(df["% Renewable"], 5)
    return df.groupby([ContinentDict, df['bins']]).size()
    raise NotImplementedError()
```

[ ]:

```
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*

[ ]:
```
import pandas as pd
import numpy as np

def answer_thirteen():
    # YOUR CODE HERE
    df = answer_one()
    df["Population"] = df['Energy Supply'] / df['Energy Supply per Capita']
    return df['Population'].apply(lambda x: '{0:,}'.format(x))
    raise NotImplementedError()
```

[ ]:
```
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.

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