

Lab6

October 15, 2024

1 STOR 320: Introduction to Data Science

1.1 Lab 6

Name: Ivy Nangalia

PID: 730670491

```
[ ]: from datetime import datetime
from bs4 import BeautifulSoup
from io import StringIO
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import requests
```

```
[ ]: %pip install html5lib
```

```
Requirement already satisfied: html5lib in
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages
(1.1)
Requirement already satisfied: six>=1.9 in
/Users/ivynangalia/Library/Python/3.12/lib/python/site-packages (from html5lib)
(1.16.0)
Requirement already satisfied: webencodings in
/Users/ivynangalia/Library/Python/3.12/lib/python/site-packages (from html5lib)
(0.5.1)
Note: you may need to restart the kernel to use updated packages.
```

2 Scraping, Merging, and Analyzing Datasets for Countries (25 points)

Background: Many times in data science, your data will be split between many different sources, some of which may be online. In this analysis assignment, we will webscrape country level data from multiple websites, clean the data individually, and merge the data. The website [Worldometers](#) contains very interesting country level data that when connected may allow us to learn interesting things about the wonderful world in which we exist.

2.1 0. GDP by Country (7 Points)

Information at [Worldometer GDP](https://www.worldometers.info/gdp/gdp-by-country/) contains GDP data from 2022 published by the world bank. GDP is the monetary value of goods and services produced within a country over a period of time. On this website, GDP is presented in dollars.

2.1.1 0.0 Scraping the Data

We will walk through webscraping the data from <https://www.worldometers.info/gdp/gdp-by-country/> using Pandas into a DataFrame called GDP. You should end up with a new object called GDP which is a DataFrame with 177 observations and 8 variables.

```
[ ]: URL_GDP = "https://www.worldometers.info/gdp/gdp-by-country/"

# Send a GET request to the URL
response = requests.get(URL_GDP)

# Parse the HTML content
soup = BeautifulSoup(response.content, 'html.parser')

# Find all tables and read into pandas DataFrame
tables = soup.find_all('table')

table_I0 = StringIO(str(tables))
GDP = pd.read_html(table_I0, flavor='bs4', header=0)[0] # Read the first table

GDP.shape
```

```
[ ]: (177, 8)
```

```
[ ]: GDP.head(5)
```

```
[ ]: #          Country GDP (nominal, 2022)      GDP (abbrev.) GDP growth \
0  1  United States $25,462,700,000,000  $25.463 trillion      2.06%
1  2           China $17,963,200,000,000  $17.963 trillion      2.99%
2  3           Japan $4,231,140,000,000   $4.231 trillion      1.03%
3  4         Germany $4,072,190,000,000   $4.072 trillion      1.79%
4  5           India $3,385,090,000,000   $3.385 trillion      7.00%

      Population (2022) GDP per capita Share of World GDP
0          341534046      $74,554          25.32%
1          1425179569     $12,604          17.86%
2          124997578      $33,850           4.21%
3           84086227      $48,429           4.05%
4          1425423212      $2,375           3.37%
```

2.1.2 0.1 Cleaning the Data (7 points)

Now that we scraped our data into a DataFrame, we need to clean it up. Perform the following tasks:

1. Remove the first ('#') and fourth ('GDP (abbrev.)') columns from the DataFrame.
2. Rename the columns 'GDP (nominal, 2022)', 'GDP growth', 'Population (2022)', 'GDP per capita', and 'Share of World GDP' to 'GDP', 'Growth', 'Population', 'PerCapita', and 'Share', respectively.
3. Remove all dollar signs, percent signs, and commas from 'GDP', 'Growth', 'PerCapita', and 'Share'.
4. Update column data type of "Country" to be a string dtype and the remaining columns to be numeric. Hint: use `pd.to_numeric`
5. Rewrite over the original 'GDP' variable with a new variable called 'GDP' that is in trillions of dollars rather than in actual dollars. Rewrite over the original 'Population' variable with a new variable of the same name that is in millions of people rather than in actual people. You are scaling the original variables to change the units without changing the variable names.

Be careful of the formatting and spacing in the original column names! Display the first five rows of the cleaned GDP DataFrame and the dtype info for GDP.

```
[ ]: # Code Solution Here

# 1
GDP = GDP.drop('#', axis=1)
GDP = GDP.drop('GDP (abbrev.)', axis=1)

# 2
GDP = GDP.rename(columns={
    'GDP (nominal, 2022)': 'GDP',
    'GDP growth': 'Growth',
    'Population (2022)': 'Population',
    'GDP per capita': 'PerCapita',
    'Share of World GDP': 'Share'
})

# 3
columns_to_clean = ['GDP', 'Growth', 'PerCapita', 'Share']
GDP[columns_to_clean] = GDP[columns_to_clean].replace({'\$', ',', '%': ''},
    ↵ '', regex=True)

# 4
GDP['Country'] = GDP['Country'].astype(str)
GDP[columns_to_clean] = GDP[columns_to_clean].apply(pd.to_numeric)

# 5
GDP['GDP'] = GDP['GDP'] / 1000000000000
GDP['Population'] = GDP['Population'] / 1000000
```

```
GDP
```

```
<>:18: SyntaxWarning: invalid escape sequence '\$'
<>:18: SyntaxWarning: invalid escape sequence '\$'
/var/folders/nw/5zcrqdxs7c57b12ptv8284p80000gn/T/ipykernel_67865/1712976710.py:1
8: SyntaxWarning: invalid escape sequence '\$'
    GDP[columns_to_clean] = GDP[columns_to_clean].replace({'\$': '', ',': '', '%':
''}, regex=True)
```

```
[ ]:
      Country      GDP  Growth  Population  PerCapita  Share
0   United States  25.462700    2.06   341.534046    74554  25.32
1      China     17.963200    2.99  1425.179569    12604  17.86
2      Japan      4.231140    1.03   124.997578    33850   4.21
3    Germany      4.072190    1.79    84.086227   48429   4.05
4      India      3.385090    7.00  1425.423212    2375   3.37
..      ...      ...      ...      ...      ...
172 Sao Tome & Principe  0.000547    0.93    0.226305    2416   0.00
173      Micronesia    0.000427   -0.62    0.523477     816   0.00
174  Marshall Islands  0.000280    1.50    0.040077    6978   0.00
175      Kiribati      0.000223    1.56    0.130469    1712   0.00
176      Tuvalu       0.000060    0.68    0.009992    6040   0.00
```

```
[177 rows x 6 columns]
```

2.2 1. Education Index Data by Country (3 Points)

Check out the [Wikipedia page](#), which contains the education index for all countries from 1990 to 2019.

2.2.1 1.0 Scraping the Education Index Data

The code provided scrapes the data from (https://en.wikipedia.org/wiki/Education_Index) into a data frame called EDU.

```
[ ]: # URL to fetch data from
URL_EDU = "https://en.wikipedia.org/wiki/Education_Index"

# Fetch the HTML content
response = requests.get(URL_EDU)
soup = BeautifulSoup(response.content, 'html.parser')

# Find the table and read it into a DataFrame
table = soup.find_all('table')[0] # Assuming the first table is the one we want
table_IO = StringIO(str(table))
EDU = pd.read_html(table_IO, flavor='bs4', header=0)[0]

EDU_preclean = EDU.copy()
```

```
EDU.head(5)
```

```
[ ]:      Country  1990  1991  1992  1993  1994  1995  1996  1997  1998 \
0  Afghanistan  0.122  0.133  0.145  0.156  0.168  0.179  0.190  0.202  0.213
1    Albania  0.583  0.588  0.557  0.542  0.528  0.550  0.557  0.569  0.579
2    Algeria  0.385  0.395  0.405  0.414  0.424  0.431  0.443  0.458  0.473
3    Andorra   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
4    Angola   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN

...  2010  2011  2012  2013  2014  2015  2016  2017  2018  2019
0  ...  0.372  0.374  0.390  0.398  0.403  0.405  0.406  0.408  0.413  0.414
1  ...  0.671  0.714  0.739  0.749  0.758  0.753  0.745  0.747  0.743  0.746
2  ...  0.626  0.644  0.639  0.639  0.652  0.659  0.660  0.665  0.668  0.672
3  ...  0.670  0.671  0.724  0.714  0.725  0.718  0.722  0.713  0.720  0.720
4  ...  0.398  0.423  0.435  0.447  0.460  0.472  0.487  0.498  0.500  0.500
```

```
[5 rows x 31 columns]
```

```
[ ]:
```

2.2.2 1.1 Cleaning the Education Data (3 points)

Perform the following tasks to clean the EDU DataFrame:

1. Modify the resulting DataFrame EDU to only keep 2 variables: 1) the country's name and 2) its education index from 2019.
2. Rename the variable named "2019" to "EDIndex".
3. Update the dtype of 'Country' to a string.

Display the first 5 rows of EDU and the info of EDU after making these changes.

```
[ ]: # Code Solution Here

# 1
EDU = EDU[['Country', '2019']]

# 2
EDU = EDU.rename(columns={'2019': 'EDIndex'})

# 3
EDU['Country'] = EDU['Country'].astype(str)

EDU.head(5)
```

```
[ ]:      Country  EDIndex
0  Afghanistan    0.414
1    Albania    0.746
2    Algeria    0.672
```

```
3      Andorra    0.720
4      Angola    0.500
```

```
[ ]: EDU.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 189 entries, 0 to 188
Data columns (total 2 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Country     189 non-null    object
1   EDIndex      189 non-null    float64
dtypes: float64(1), object(1)
memory usage: 3.1+ KB
```

2.3 2: Merging the Datasets (8 points)

Now, we are going to merge the datasets for maximum gains. Make sure you carefully read the instructions for each question. Be very careful in this part of the assignment.

2.3.1 2.0 Joining GDP and EDU (2 Points)

The dataset GDP is our primary dataset. Create a new DataFrame `GDP_EDU` that brings the the education data from `EDU` into the dataset `GDP` using a left join only. Display the first 12 rows of `GDP_EDU`.

```
[ ]: # Code Solution Here
GDP_EDU = pd.merge(GDP,EDU, how="left")
GDP_EDU
```

```
[ ]:
```

	Country	GDP	Growth	Population	PerCapita	Share \
0	United States	25.462700	2.06	341.534046	74554	25.32
1	China	17.963200	2.99	1425.179569	12604	17.86
2	Japan	4.231140	1.03	124.997578	33850	4.21
3	Germany	4.072190	1.79	84.086227	48429	4.05
4	India	3.385090	7.00	1425.423212	2375	3.37
..
172	Sao Tome & Principe	0.000547	0.93	0.226305	2416	0.00
173	Micronesia	0.000427	-0.62	0.523477	816	0.00
174	Marshall Islands	0.000280	1.50	0.040077	6978	0.00
175	Kiribati	0.000223	1.56	0.130469	1712	0.00
176	Tuvalu	0.000060	0.68	0.009992	6040	0.00

	EDIndex
0	0.900
1	0.862
2	0.851
3	0.943

```

4      0.555
..      ...
172     NaN
173     NaN
174     0.707
175     0.594
176     NaN

```

```
[177 rows x 7 columns]
```

2.3.2 2.1 Missing Education Index (2 Points)

How many countries in GDP_EDU have missing values for Education Index? Show code that can be used to answer this question and then write your answer in complete sentences.

```
[ ]: # Code Solution Here

missing_edu = GDP_EDU['EDIndex'].isnull().sum()
missing_edu

```

```
[ ]: 19
```

Answer: There are 19 missing values for Education Index. This was found by slicing the GDP_EDU dataframe by the EDIndex and summing all the null (NaN) values.

2.3.3 2.2 Data Inspection (3 Points)

Closely inspect the original datasets and answer the following questions about GDP_EDU in complete sentences. You can use the code if needed, but it is not required. Please show all work. If you don't reference the appropriate dataset or you are not specific in your answers, you will get 0 points.

2.2.0 Why is there no education index for Iran in the dataset G_EDU? (1 Point)

```
[ ]: # assuming G_EDU is referring to GDP_EDU
EDU_preclean[EDU_preclean['Country'] == 'Iran']

```

```
[ ]: Empty DataFrame
Columns: [Country, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999,
2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012,
2013, 2014, 2015, 2016, 2017, 2018, 2019]
Index: []

```

```
[0 rows x 31 columns]
```

```
[ ]: EDU_preclean[EDU_preclean['Country'] == 'Iran (Islamic Republic of)']

```

```
[ ]:
      Country  1990  1991  1992  1993  1994  1995 \
79  Iran (Islamic Republic of)  0.397  0.414  0.432  0.449  0.466  0.483

```

	1996	1997	1998	...	2010	2011	2012	2013	2014	2015	2016	\
79	0.493	0.504	0.514	...	0.662	0.684	0.731	0.738	0.738	0.739	0.747	

	2017	2018	2019
79	0.749	0.75	0.756

[1 rows x 31 columns]

Answer: Looking at the original dataset, Iran has an education index of 0.756 in 2019, but it's listed as "Iran (Islamic Republic of)", which likely caused the error.

2.2.1 Why is there no education index for State of Palestine in the dataset GDP_EDU? (1 Point)

```
[ ]: EDU_preclean[EDU_preclean['Country'] == 'State of Palestine']
```

```
[ ]: Empty DataFrame
Columns: [Country, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999,
2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012,
2013, 2014, 2015, 2016, 2017, 2018, 2019]
Index: []
```

[0 rows x 31 columns]

```
[ ]: EDU_preclean[EDU_preclean['Country'] == 'Palestine, State of']
```

```
[ ]:
      Country  1990  1991  1992  1993  1994  1995  1996  1997 \
131  Palestine, State of   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN

      1998  ...  2010  2011  2012  2013  2014  2015  2016  2017  2018 \
131   NaN  ...  0.656  0.662  0.671  0.66  0.663  0.668  0.67  0.675  0.676

      2019
131  0.678
```

[1 rows x 31 columns]

Answer: There is no "State of Palestine" in the original dataset. There is, however, a "Palestine, State of" value with an education index of 0.678 in 2019.

2.2.2 Why is there no education index for Laos in the dataset GDP_EDU? (1 point)

Answer: Same reason as the other two, it's listed as "Lao People's Democratic Republic" in the original dataset, and so since the names don't match up the join did not work.

2.3.4 2.2 Removing NA Values (1 point)

Instead of replacing or dropping all the countries with missing values by hand, we will just drop all rows that are missing the Education Index to move forward with the analysis portion. Drop all rows from GDP_EDU that are null for EDIndex.


```
[ ]: # Code Solution Here
```

```
GDP_EDU = GDP_EDU.dropna(subset=['EDIndex'])
```

2.4 3. Analyzing the Merged Dataset (12 points)

In these questions, find the answer using code, and then answer the question using complete sentences below the code.

2.4.1 3.0 Above Average GDP PerCapita (2 Points)

How many countries have a GDP per capita above the global average?

```
[ ]: # Code Solution Here
```

```
GDP_EDU[GDP_EDU["PerCapita"] > GDP_EDU["PerCapita"].mean()].count()[0]
```

```
/var/folders/nw/5zcrqdxs7c57b12ptv8284p80000gn/T/ipykernel_67865/229476441.py:2:  
FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a  
future version, integer keys will always be treated as labels (consistent with  
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`  
GDP_EDU[GDP_EDU["PerCapita"] > GDP_EDU["PerCapita"].mean()].count()[0]
```

```
[ ]: 49
```

Answer: There are 49 countries with an above-average GDP per capita.

3.1 Highest GDP Growth Rate (4 Points)

- Of the countries that have above average GDP PerCapita, what country has the highest GDP growth rate?
- Of the countries that have below average GDP PerCapita, what country has the highest GDP growth rate?

```
[ ]: # Code Solution Here
```

```
higher = GDP_EDU[GDP_EDU["PerCapita"] > GDP_EDU["PerCapita"].mean()]  
highest_above_idx = higher.loc[higher['Growth'].idxmax()]  
pd.DataFrame(highest_above_idx) # for prettier display
```

```
[ ]:  
          127  
Country    Guyana  
GDP         0.015358  
Growth       57.8  
Population  0.821637  
PerCapita   18691  
Share       0.02  
EDIndex     0.601
```

```
[ ]: lower = GDP_EDU[GDP_EDU["PerCapita"] < GDP_EDU["PerCapita"].mean()]  
lowest_above_idx = lower.loc[lower['Growth'].idxmax()]  
pd.DataFrame(lowest_above_idx)
```

```
[ ]:
Country      Cabo Verde
GDP           0.002315
Growth        17.71
Population    0.519741
PerCapita     4454
Share         0.0
EDIndex       0.562
```

Answer: * Of the countries that have above average GDP Per Capita, Guyana has the highest growth rate. * For the below-average countries, Cabo verde has the highest growth rate.

3.2 Lowest Education Index (4 Points)

- Of the countries that have above average GDP PerCapita, what country has the lowest education index?
- Of the countries that have below average GDP PerCapita, what country has the lowest education index?

```
[ ]: GDP_EDU
```

```
[ ]:
Country      GDP  Growth  Population  PerCapita  Share  \
0      United States  25.462700   2.06   341.534046    74554  25.32
1           China  17.963200   2.99  1425.179569   12604  17.86
2           Japan   4.231140   1.03   124.997578   33850   4.21
3          Germany   4.072190   1.79    84.086227   48429   4.05
4           India   3.385090   7.00  1425.423212    2375   3.37
..          ...
167        Vanuatu   0.000984   1.85    0.313046    3142   0.00
170         Samoa   0.000832  -6.02    0.215261    3867   0.00
171        Dominica  0.000612   5.94    0.066826    9159   0.00
174  Marshall Islands  0.000280   1.50    0.040077    6978   0.00
175         Kiribati  0.000223   1.56    0.130469    1712   0.00

EDIndex
0      0.900
1      0.862
2      0.851
3      0.943
4      0.555
..      ...
167     0.561
170     0.713
171     0.632
174     0.707
175     0.594
```

```
[158 rows x 7 columns]
```

```
[ ]: # Code Solution Here
highest_lowest_EDU_idx = higher.loc[higher['EDIndex'].idxmin()]
pd.DataFrame(highest_lowest_EDU_idx)
```

```
[ ]:
Country      Guyana
GDP          0.015358
Growth       57.8
Population   0.821637
PerCapita    18691
Share        0.02
EDIndex      0.601
```

```
[ ]: lowest_lowest_EDU_idx = lower.loc[lower['EDIndex'].idxmin()]
pd.DataFrame(lowest_lowest_EDU_idx)
```

```
[ ]:
Country      Niger
GDP          0.01397
Growth       11.5
Population   25.311973
PerCapita    552
Share        0.01
EDIndex      0.249
```

Answer: For countries with above-average GDP, Guyana has the lowest EDIndex, but for below-average GDP countries, Niger has the lowest.

3.3 Critical Thinking (2 points) State two additional questions you could answer with the merged dataset. Be creative. You do not need to find the answer, but are welcome to if you are curious.

Answer: 1. Is there a correlation between GDP per capita and EDIndex? 2. Is there a correlation between GDP Growth and EDIndex?