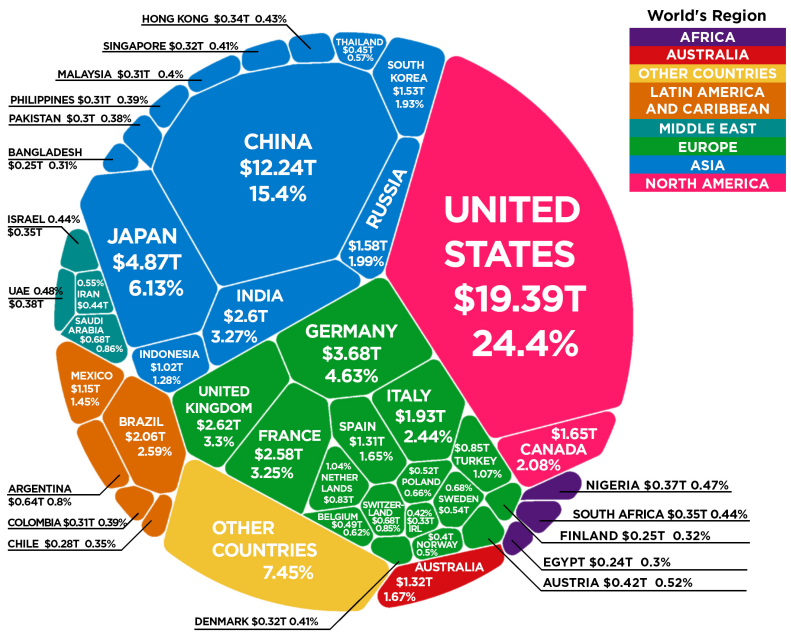
Day 9 In-Class: Cleaning and Analyzing Economic Data



Article and Sources:
https://howmuch.net/articles/the-world-economy-2017
http://databank.worldbank.org/data/download/GDP.pdf

howmuch.net

✅ Put your name here

✅ Put your group member names here.

Goals for today's in-class project

- Load in data and clean Pandas dataframes
- Learn different ways to index Pandas dataframes
- Analyze different countries GDP data
- Practice using online research to learn new programming skills

Assignment instructions

Work with your group to complete this assignment. The assignment is due at the end of class and should be uploaded to the appropriate submission folder on D2L.

Part 1: Practice calculating statistics using Python

1.1 Computing standard deviation by hand

✅ **Fix a function** that takes in a *list of values* and calculates the *standard deviation* using only basic python functions. The function is already written but it doesn't *quite* work. Run the cell to see. Here's the equation for standard deviation:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i-\mu)^2}{N}}$$

where the symbols in this equation represent the following:

- σ: Standard Deviation
- μ: Mean
- N: Number of observations
- x_i: the value of dataset at position i

```
# Fix the function here
def std(vals):
    length = len(vals)
    mean = sum(vals)
    diffs = []
    for i in range(length):
        diffs.append(vals[i] - mean)
    return sum(diffs) ** 0.5
```

✅ **Check your function for accuracy**

Call your function using the variable `test_list` (provided below) as the input and compare your function's output with that of `np.std()` to make sure you calculated standard deviation correctly.

Contents

```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
```

```
test_list = [1,3,5,10,15,5]

# Put your code for comparing the answers here
```

1.2 Next, we will apply stats to a distribution *visually*

But first, let's cover how to visualize the distribution of a one-dimensional data set. We begin with a random distribution of numbers from a random number generator in the NumPy library.

```
# You might not be familiar this with random number generator, that's OK,
# This is one of _many_ that are available in NumPy.
random_distribution = np.random.wald(200,500,size=1000)
```

This is an array holding 1000 random numbers, generated from a statistical distribution called the "Wald distribution".

Let's look at the first 10 numbers:

```
random_distribution[0:10]
```

And plot all of the elements:

```
plt.plot(random_distribution,'o')
plt.xlabel('Index')
plt.ylabel('Value')
```

What are some other ways we can analyze and visualize this data? One visualization is a box plot, which shows where the ***quartiles*** of the data set are, as well as outliers.

```
box = plt.boxplot(random_distribution, vert=False)
ylabel = plt.ylabel("Frequency")
xlabel = plt.xlabel("Value")
title = plt.title("Wald Distribution")
```

Another visualization is a histogram, which splits the data set into a bunch of equally sized intervals, and then graphs the number of data points that fall into each interval. The higher the bar on a histogram, the more data points in that interval.

```
hist = plt.hist(random_distribution, bins=50, color="k", alpha=0.5) #what's the alpha
argument doing?
ylabel = plt.ylabel("Frequency")
xlabel = plt.xlabel("Value")
title = plt.title("Wald Distribution")
```

✅ **Compare the representations above**

What are the similarities between how the boxplot represents the data set versus the histogram? What does the boxplot do a better job of showing? What does the histogram do a better job of showing?

✎ *Put your answer here*

1.3 Compute and Compare

✅ **Now lets actually compute the mean and median and visualize them on the distribution graph.**

Add **two vertical lines** with different colors where the **mean** and **median** are using Matplotlib's `plt.axvline()` function – this might be new to you, so make sure you understand how it works!

Make sure you label your lines and include a legend.

```
hist = plt.hist(random_distribution,bins=50,color="k",alpha=0.5)
# Add your additional plotting commands here
#plt.axvline(x=?,linewidth=2, color='r',label = "Median")
```

✅ **What is larger for this data set, mean or median? Explain why you think that is.**

✎ *Put your answer here*

Part 2: Loading in and cleaning economic data

The next part we will focus on transforming and manipulating a dataset using Pandas. As data scientist/computational professional in training, one of the goals we want you to accomplish is to be comfortable searching through online resources to try and solve problems. There are far too many functions and concepts in programming to remember everything so in practice it's essential to utilize package documentation, stack overflow, etc. Some of the questions you will see below will ask you to use or look for a function you've never used before to get you to practice Googling questions that help you accomplish your task.

We will be analyzing a dataset from the World Bank containing yearly GDP data for countries from 1960-2020. The GDP numbers have been converted to USD for all countries by the exchange rate at the time. Which is important to note because depending on the exchange rates at the time this could over/under value the non US countries numbers or increase the variance of GDP.

Link to dataset: https://data.worldbank.org/indicator/ny.gdp.mktp.cd

GDP stands for **Gross Domestic Product** and it is equal to the market value of all the finished goods and services produced within a country's borders in a specific time period.

GDP = Consumer Spending + Private Investment + Government Expenditure + Net Exports

---

2.1 Cleaning data is an important part analyzing data.

First, we're going to load in the .csv dataset into a Pandas Dataframe and explore the original structure of the data and think about if it could be formatted in a more useful way. **There might be multiple .csv's in the download. You might have to figure out which file has the GDPs before you load it into your notebook.**

Make sure you import the Pandas module before moving on!

```
# put your Pandas import command here
```

✅ **Load in the insert_filename_here.csv file using `pd.read_csv()`. Skip the first 4 rows and use a comma as the delimiter. Then display the first few lines using `.head()`.**

Use `gdp` as the variable name for storing your dataframe as indicated in the code cell below.

```
# Load in GDP.csv
#gdp = # Finish this line to load in the data!
```

As a first step to cleaning a data set, it can be helpful to get rid of rows that have a lot of "NaN" values. NaN means "Not a Number," and it is a value that sometimes takes the place of a blank entry. Countries that did not track GDP as far back as 1960 will have some NaN values, such as Aruba and Angola. You may want to keep these rows in your own data sets, but for this assignment, we are going to "drop" them from the data set, using a handy pandas function called "dropna".

```
gdp = gdp.dropna(axis="columns", how="all") # drop empty columns, like the last column
in gdp
gdp = gdp.dropna() # drop rows with NaNs, like Aruba and Angola
gdp.head()
```

Note that the `dropna` function was accessed from the dataframe itself (`gdp.dropna()`). These functions are included with each dataframe object. We've already seen this with functions like `describe()`, and even with numpy array functions like `my_array.sum()`. Many of the functions you'll be using today are included with the dataframe objects.

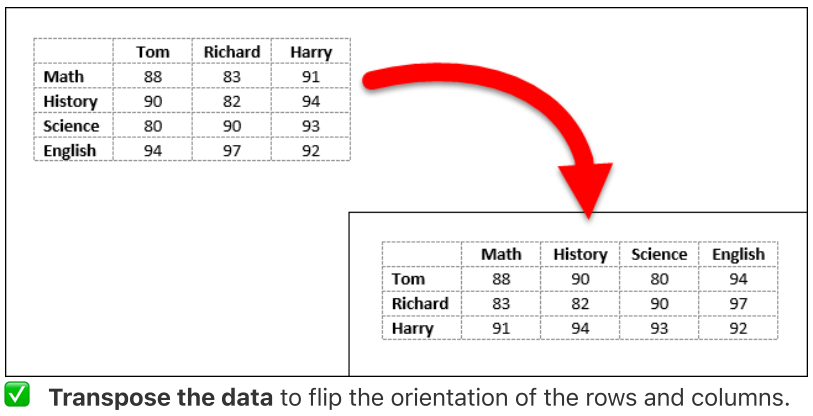
You can browse through these built-in functions by typing the name of a dataframe followed by `.` and then hitting the tab key. Try it out below!

```
# uncomment the line below, then go to the end of the line and hit tab
#gdp.
```

If you want to learn more about something, select it (or type it out), add a question mark, and then run the cell.

Reflecting on the data: Typically when we are looking at data over time we represent each time step as a row rather than a column. Let's transpose the dataset to get years as rows. Pandas dataframes have a built in transpose function – see if you can figure out what it is!

Example of Transposing:



	Tom	Richard	Harry
Math	88	83	91
History	90	82	94
Science	80	90	93
English	94	97	92

	Math	History	Science	English
Tom	88	90	80	94
Richard	83	82	90	97
Harry	91	94	93	92

✅ **Transpose the data** to flip the orientation of the rows and columns.

```
# Transpose the dataframe here and check to see if it worked
```

One of the benefits of Pandas Dataframe is being able to index a column by name rather than a number.

✅ **Modify the dataframe so that each country name is used as the column headers by assigning the first row of the dataframe to be the column headers.**

Think back to the Pre-Class Assignment. What information are you looking to retreive? What tools do you have to access this information? You may want to use `.iloc` to do this. If done correctly, you then should be able to index a column out of our dataframe using `gdp['United States']`, for example. Make sure to test this out!

```
# Change the column headers to be the country names here.
```

This is looking pretty good!

Of course, now we have a few redundant rows: “Country Name”, “Country Code”, “Indicator Name”, and “Indicator Code”. We don’t really need these any more now that we’ve change the column labels.

✅ **Remove these four rows, since they don’t contain yearly GDP data.** There’s more than one way to do this. The best option is to use the `gdp.drop()` function. Figure out how it works with `gdp.drop?`.

```
# Try to remove the rows that don't represent years here
```

Now our dataset should be in an easier format. The next step is to examine the structure of our data.

✅ **Review the following code and comment what each line is doing.**

```
print(gdp.index) #comment here


print(type(gdp.index[0])) #comment here
```

We can see the index column is made up of strings representing years, which isn’t ideal!

The code below will change the data type from strings to integers. This will be helpful for when we begin plotting because when you try to plot strings as numbers it doesn’t usually work out very well!

```
gdp.index = gdp.index.astype(int)
gdp.index
```

2.2 Exploring the Data

✅ **Now pick 2 countries and print the GDP for year 1975 using `.loc` and the column name.** Again, look back at the dataset and think about what information we are looking to retreive. What information is stored in our columns? What information is stored in our rows?

```
# Put your code here
```

✅ Now, **plot those two countries GDP *in billions of dollars* from 1960-2020, make sure to have proper labels and legends.**

```
# Put your plotting commands here
```

✅ **Question** Is this a good way visual comparison for the two countries? If one country has a much larger GDP or much larger population than the other country what would be a better way to normalize or compare the data? This might involve doing some sort of calculation or visualizing the data differently.

✎ *Put your answer here*

2.3 Exploring the log-linear plot

During the COVID-19 pandemic, some of the visualizations floating around that show the numbers of confirmed cases in various places around the world have been “log-linear” plots which uses a logarithmic scale (tick marks indicate powers of 10) on the y-axis and a linear scale on the x-axis. Some folks have even written papers about how these sort of plots may or may not impact how people perceive the need for confinement to stop the spread of the virus.

You can change the scaling of an axis using `plt.yscale('log')`.

✅ Try using a log scale for the GDP in the previous plot for the two different countries to see if it facilitates a better comparison!

```
# Try making a "semilogy" plot here
```

✅ **Question** Do you find this to be a better way to visualize the data, yes or no? Explain your reasoning!

✎ *Put your answer here*

2.4 Analyzing growth rates

One way to compare GDP between different countries in a way that is unit free would be to consider the countries *growth rates*. The growth rate for a year would be equal to the **percent change** going from one year to another, defined like so:

- Growth Rate in 1961 = (GDP in year 1961 - GDP in year 1960) / GDP in year 1960

✅ **Lets plot two countries growth rates on the same plot over time making sure to properly label our graph.**

(Pandas dataframes might have a function that can compute the percent change for you – time to consult the internet again!)

Try using the Pandas plotting functions for this part: https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html

```
# Calculate and plot the growth rates as a function of time
```

✅ **Question**: Why might comparing growth rates be a better comparison for countries that have GDP's of very different magnitudes?

✎ *Put your answer here*

Time Permitting: More Data Manipulation (time permitting or if you're interested in exploring the data further on your own time!)

Filtering, sorting, and calculating new quantities

You've been able to clean, transform, and visualize the data, but for an extra challenge let's use any time you have remaining to work on filtering and sorting your data.

The below analyses are going to focus on data for individual countries for the year 2020. To get started, we're going to:

1. create a new dataset for only the year 2020, and
2. drop columns that don't correspond to individual countries

```
gdp2020 = gdp.loc[2020]
gdp2020 = gdp2020.drop(['World', 'High income', 'OECD members', 'Post-demographic
dividend', 'IDA & IBRD total',
                        'Low & middle income', 'Middle income', 'IBRD only', 'Upper
middle income',
                        'North America', 'Late-demographic dividend',
                        'East Asia & Pacific (excluding high income)',
                        'East Asia & Pacific (IDA & IBRD countries)', 'Euro area',
'Early-demographic dividend',
                        'Lower middle income', 'Latin America & Caribbean',
                        'Latin America & the Caribbean (IDA & IBRD countries)',
                        'Latin America & Caribbean (excluding high income)', 'South
Asia',
                        'South Asia (IDA & IBRD)', 'IDA total', 'Fragile and conflict
affected situations',
                        'Sub-Saharan Africa', 'Sub-Saharan Africa (IDA & IBRD
countries)',
                        'Sub-Saharan Africa (excluding high income)', 'IDA only'])
```

✅ **Great! Now filter the top 10% of countries in the cleaned up data set by their 2020 GDP, print their names, and store the names in a list in ordered by their GDP ranking.**

(Hint Pandas has a `quantile` function that could be useful to find the value for the 10% cut off as well as a function for sorting the values)

```
# Put your code for finding the countries with the highest 10 GDP values and sorting
them here
```

✅ **Let's take a closer look at how the countries rank by plotting a horizontal bar graph of the top 10% countries GDP in billions by ranking order starting with the highest GDP.**

Pandas dataframes have a horizontal bar graph function as well (`.plot.barh()`) – isn't Pandas handy?

```
# Make your horizonal bar graph here
```

✅ **With Pandas, we can pull multiple columns at the same time. Using that list of the top 10% of countries, create a subset of the original GDP dataframe that has data for only the last 20 years for countries in your list of top 10%.**

We can create a subset by setting a new variable to equal the subset of the Dataframe.

(something like: `Subset = DataFrame[list_of_columns_headers]`)

```
# Put your code here and create additional code cells as needed
```

✅ Next, **calculate the standard deviation of GDP for each country in the subset dataframe over the last 20 years. Recreate the Horizontal Bargraph above with but using the standard deviation.**

✅ **Are there similiarities between the GDP graph and the standard deviation graph? Explain why you think they look similiar and what the limitation is with using standard deviation to compare the variation of the GDP for different contries.**

✎ *Put your answer here*

Let's pause and think about the following example:

```
X = np.array([2,4,10,15,30,50])
print(np.std(X))

X2 = X*50
print(np.std(X2))
```

The idea here is the spread between the numbers in the datasets `X` & `X2` are the same in when considered as a percentage of the total, but the standard deviation will be proportionally higher for `X2`. This means the difference between each observation and the mean from a percentage basis is the same, but because the values in `X2` are 50 times larger, the standard deviation will be 50 times larger.

The take away is if we want to compare how much a countries GDP growth varies relative to another country, we want an apples to apples comparison. For example, taking the standard deviation of the United States compared to Thailand we would expect United States to have a higher standard deviation because the US GDP is much higher. When in reality, Thailand's GDP varies relatively more than the United States GDP varies.

In order to compare the variation in GDP by countries of different magnitudes we want to be looking at the change in GDP from a percentage view.

✅ **Recreate the horizontal bar graph again, but this time take the standard deviation of the percent changes, or growth rates, of GDP for the last 20 years.**

```
# Put your code here
```

✅ **What do you observe? Why is taking the standard deviation of growth rates a better assessment of volatility than standard deviation of normal GDP for this data?**

✎ *Put your answer here*

Congratulations, you're done!

Submit this assignment by uploading your notebook to the course Desire2Learn web page. Go to the "In-Class Assignments" folder, find the appropriate submission link, and upload everything there. Make sure your name is on it!
