Measuring Income Equality with the Gini Coefficient

As we discussed in our numpy exercises, one frequently used measure of inequality is the Gini Coefficient. The Gini Coefficient takes on a value of 1 when the distribution of some property is maximally unequal across a said of entities, and a value of 0 when it is evenly distributed.

In this exercise, we will calculate the Gini Coefficient for income inequality across the countries of the world to get a sense of income inequality *across* countries.

Gradescope Autograding

Please follow all standard guidance for submitting this assignment to the Gradescope autograder, including storing your solutions in a dictionary called `results` and ensuring your notebook runs from the start to completion without any errors.

Starting with this assignment, submissions that have not been formatted with `black` will be automatically rejected.

For this assignment, please name your file `exercise_series.ipynb` before uploading.

You can check that you have answers for all questions in your `results` dictionary with this code:

```
assert set(results.keys()) == {
    "ex2_mean",
    "ex2_median",
    "ex3_highest_gdp_percap",
    "ex3_lowest_gdp_percap",
    "ex4_lessthan20_000",
    "ex5_switzerland",
    "ex6_gini_loop",
    "ex7_gini_vectorized",
    "ex8_gini_2025",
}
```

Submission Limits

Please remember that you are **only allowed three submissions to the autograder.** Your last submission (if you submit 3 or fewer times), or your third submission (if you submit more than 3 times) will determine your grade Submissions that error out will **not** count against this total.

Exercise 1

To get accustomed to Series, let's explore some data on the wealth of 10 randomly selected countries. Data below presents the GDP per capita for these countries in 2008.

Use the code below to get started:

```
gdppercap = pd.Series(
    [34605, 34493, 12393, 44200, 10041, 58138, 4709, 49284, 10109, 42536],
    index=[
        "Bahrain",
        "Belgium",
        "Bulgaria",
        "Ireland",
        "Macedonia",
        "Norway",
        "Paraguay",
        "Singapore",
        "South Africa",
        "Switzerland",
    ],
)
```

In [ ]:
```
import pandas as pd

gdppercap = pd.Series(
    [34605, 34493, 12393, 44200, 10041, 58138, 4709, 49284, 10109, 42536],
    index=[
        "Bahrain",
        "Belgium",
        "Bulgaria",
        "Ireland",
        "Macedonia",
        "Norway",
        "Paraguay",
        "Singapore",
        "South Africa",
        "Switzerland",
    ],
)
```

Exercise 2

Find the mean, median, minimum and maximum values of GDP per capita in this data.

In [ ]:
```
mean = gdppercap.mean()
median = gdppercap.median()
min = gdppercap.min()
max = gdppercap.max()
print(mean)
print(median)
```

```
print(max)
print(min)

results = {}
```

```
30050.8
34549.0
58138
4709
```

Exercise 3

Programmatically, determine which country in our data has the highest income per capita, and which has the lowest income per capita.

(Obviously, this is easier to do by just looking at the data, but that's only because this dataset is very small. With a real dataset, you would need to do it with code, so please write code to accomplish this task.)

Hint: Country names form the index for this Series, so to get country names you'll need to access the index.

Store the country names *as strings* with the keys `"ex3_highest_gdp_percap"` and `"ex3_lowest_gdp_percap"`

In [ ]:
```
# Maximum
gdppercap.head()
max_gdp = gdppercap.max()
# gdppercap.idxmax()
id_max = gdppercap[gdppercap == max_gdp].index

# Minimum
min_gdp = gdppercap.min()
id_min = gdppercap[gdppercap == min_gdp].index


results["ex3_highest_gdp_percap"] = id_max[0]
results["ex3_lowest_gdp_percap"] = id_min[0]
results
```

Out[ ]:
```
{'ex3_highest_gdp_percap': 'Norway', 'ex3_lowest_gdp_percap': 'Paraguay'}
```

Exercise 4

Get Python to print out the names of all the countries that have GDP per capita of less than $20,000.

Store these countries in a list, sorted alphabetically, and store it in `results` under the key `"ex4_lessthan20_000"`

In [ ]:
```
ex4 = gdppercap[gdppercap < 20000]
results["ex4_lessthan20_000"] = ex4[0]
results
```

```
C:\Users\Simrun Sharma\AppData\Local\Temp\ipykernel_31340\4179986109.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]`
  results["ex4_lessthan20_000"] = ex4[0]
```

Out[ ]:
```
{'ex3_highest_gdp_percap': 'Norway',
 'ex3_lowest_gdp_percap': 'Paraguay',
 'ex4_lessthan20_000': 12393}
```

Exercise 5

Get Python to print out the GDP per capita of Switzerland. Store the result as `ex5_switzerland`:

In [ ]:
```
# exercise 5
ex5 = gdppercap.loc["Switzerland"]
results["ex5_switzerland"] = ex5
results
```

Out[ ]:
```
{'ex3_highest_gdp_percap': 'Norway',
 'ex3_lowest_gdp_percap': 'Paraguay',
 'ex4_lessthan20_000': 12393,
 'ex5_switzerland': 42536}
```

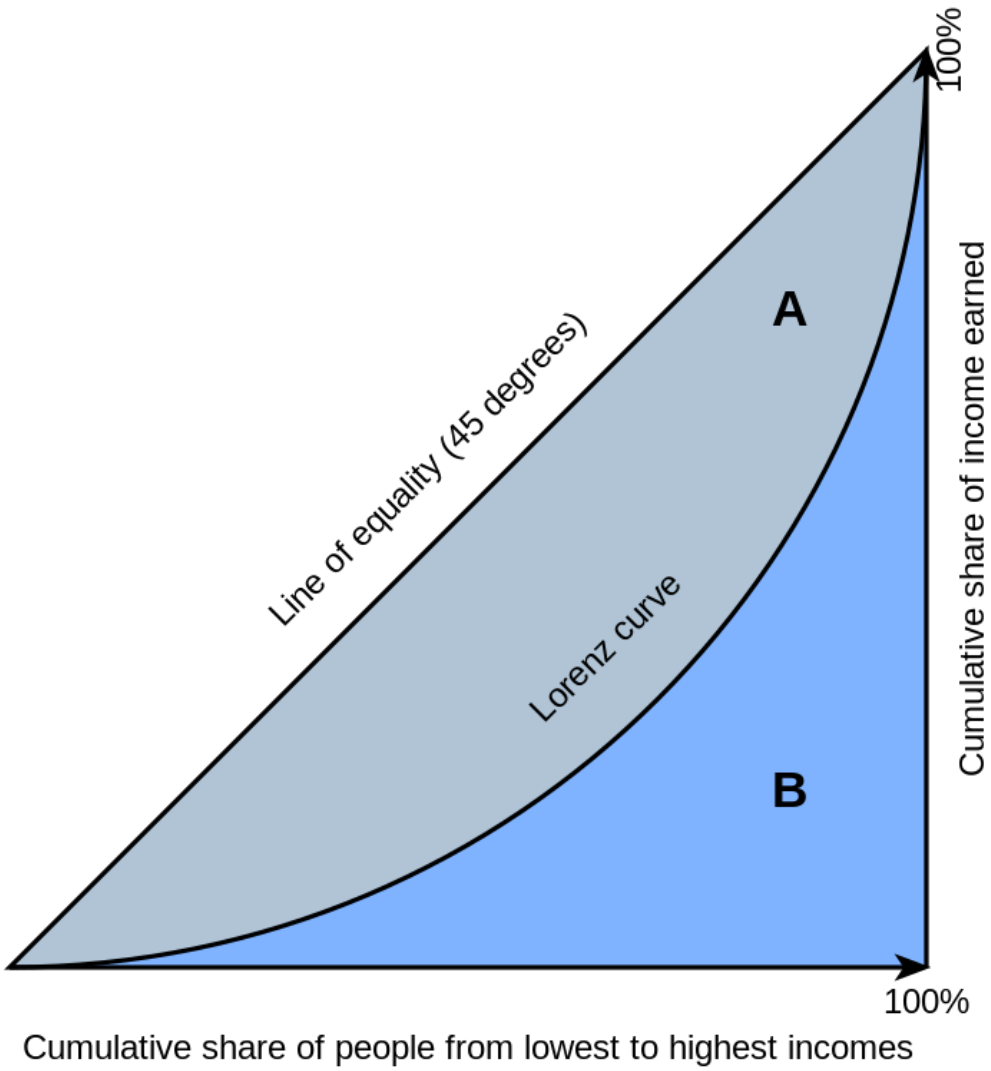
Exercise 6

One frequntly used measure of inequality is the Gini Coefficient. The Gini Coefficient takes on a value of 1 when the distribution of some variable is maximally unequal across a population, and a value of 0 when it is evenly distributed. We will calculate the Gini Coefficient for income inequality in our data.

To visualize the Gini Coefficient, we plot the cumulative share of the population (ordered from poorest to richest) on the x-axis, and cumulative share of income earned by that group on the y-axis. The Gini Coefficient is then defined as

$$\frac{A}{A+B}$$

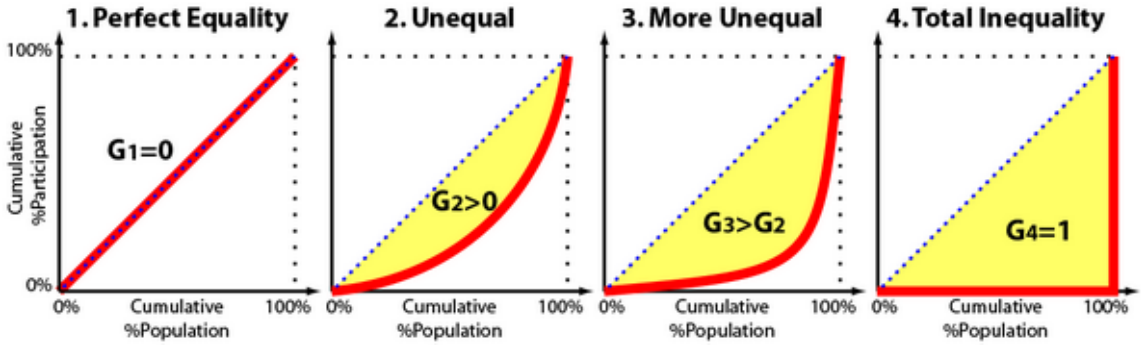
, where the areas A and B are labeled below:



If income is evenly distributed, then the poorest 20% of a population will also have 20% of the wealth; the poorest 40% will have 40% of the wealth, and so forth, resulting in a perfect 45 degree line. In this situation, there is no area between the 45% line and the actual income distribution, so $A = 0$, and the Gini Coefficient is 0.

If, by contrast, the top 10% of people hold all the wealth in a country, then there will be no wealth for the poorest 90% of people, then wealth will jump up at the far right side of the graph. This will generate a very large gap between the 45% line and actual income for most of the graph, generating a large value for the area A, creating a very high Gini Coefficient.

To illustrate, here are a few different Gini plots. These come from someone studying inequality of participation, so to adapt this to our study of income, just imagine the y-axis plots share of income):



For discrete data, the Gini Coefficient can be calculated with the following formula:

$$\frac{2\sum_{i=1}^{n} iy_i}{n\sum_{i=1}^{n} y_i} - \frac{n+1}{n}$$

Where i is each country's rank ordering from poorest to richest, and y_i is the income of country i.

Exercise 6

Using this formula, calculate the Gini coefficient for our income data.

Begin by writing a function to calculate the Gini Coefficient for our data *by looping over the entries in our Series*. In other words, try and embrace the spirit of how you might normally think about interpreting the summation notation written above.

Store the gini coefficient you calculate in `results` under the key `"ex6_gini_loop"`.

HINT: Be careful with 0-indexing! Python counts from 0, but mathematical formulas (like $\sum$) start from 1!

HINT 2: I'll probalby ask you to use this more than once, so please put it in a function.

def cal_gini_loop(data): sorted_data = sorted(data) total_income = sum(sorted_data) cumulative_sum = 0 n = len(sorted_data) for i in range(n): y_i = sorted_data[i] cumulative_sum += 2 * (i + 1) * y_i gini_coeff = (cumulative_sum / (n * total_income)) - ((n+1)/n) return gini_coeff

x = cal_gini_loop(gdppercap) print(x) results["ex6_gini_loop"] = x results

In [ ]:
```
def cal_gini_loop(data):
    sorted_data = sorted(data)
    total_income = sum(sorted_data)
    cumulative_sum = 0
    n = len(sorted_data)
    for i in range(n):
        y_i = sorted_data[i]
        cumulative_sum += 2 * (i + 1) * y_i
    gini_coeff = (cumulative_sum / (n * total_income)) - ((n + 1) / n)
```

```
    return gini_coeff


x = cal_gini_loop(gdppercap)
print(x)
results["ex6_gini_loop"] = x
results
```

```
0.3382798461272245
```

Out[ ]:
```
{'ex3_highest_gdp_percap': 'Norway',
 'ex3_lowest_gdp_percap': 'Paraguay',
 'ex4_lessthan20_000': 12393,
 'ex5_switzerland': 42536,
 'ex6_gini_loop': 0.3382798461272245}
```

Exercise 7

Excellent! But as we've seen in our readings, in data science we generally strive to *not* loop over the entries in our arrays; instead, we aspire to write *vectorized code* that naturally applies a simple operation to each observation.

So now write a new function to calculate the Gini Coefficient that *doesn't* use loops, and instead relies on vectorized code.

Store the result in `results` under the key `"ex7_gini_vectorized"`.

HINT: you will probably have to create some new series/vectors/arrays.

In [ ]:
```
import numpy as np


def cal_gini_vector(data):
    data = np.array(data.sort_values())
    n = len(data)
    i = np.arange(1, n + 1)
    numerator = np.sum(2 * data * i)
    denominator = np.sum(n * data)
    return numerator / denominator - (n + 1) / n


x = cal_gini_vector(gdppercap)
print(x)
results["ex7_gini_vectorized"] = x
results
```

```
0.3382798461272245
```

Out[ ]:
```
{'ex3_highest_gdp_percap': 'Norway',
 'ex3_lowest_gdp_percap': 'Paraguay',
 'ex4_lessthan20_000': 12393,
 'ex5_switzerland': 42536,
 'ex6_gini_loop': 0.3382798461272245,
 'ex7_gini_vectorized': 0.3382798461272245}
```

Exercise 8

The result we just generated offers a snap-shot of inequality for this subset of countries. But what are the dynamics of inequality for these countries?

There is an idea in economics called the "convergence hypothesis", which argues that poorer countries are likely to grow faster, and as a result global inequality is likely to decline. Economists advocating for this hypothesis pointed out that while rich countries had to invent new technologies in order to grow, many poor countries simply had to take advantage of innovations already developed by rich countries.

To test this hypothesis, let's do a small analysis of the dynamics of income inequality in our sample. Create the following Series in your Python session, which provides the average growth rate of GDP per capita for all the countries in our sample from 2000 to 2018.

```
avg_growth = pd.Series(
    [
        -0.29768835,
        0.980299584,
        4.52991925,
        3.686556736,
        2.621416804,
        0.775132075,
        2.015489468,
        3.345793635,
        1.349993318,
        0.982775018,
    ],
    index=[
        "Bahrain",
        "Belgium",
        "Bulgaria",
        "Ireland",
        "Macedonia",
        "Norway",
        "Paraguay",
        "Singapore",
        "South Africa",
        "Switzerland",
    ],
)
```

Using this data on average growth rates in GDP per capita, and assuming growth rates from 2000 to 2018 continue into the future, estimate what our Gini Coefficient may look like in 2025 (remembering that income in our data is from 2008, so we're extrapolating ahead 17 years)?

Hint: the formula for compound growth (i.e. value of something growing at a rate of `x` percent for t periods) is:

$$future\_value = current\_value * (1 + \frac{percentage\_growth\_rate}{100}))^t$$

Store the answer in `results` under the key `"ex8_gini_2025"`

In [ ]:
```
avg_growth = pd.Series(
    [
        -0.29768835,
        0.980299584,
        4.52991925,
        3.686556736,
        2.621416804,
        0.775132075,
        2.015489468,
        3.345793635,
        1.349993318,
        0.982775018,
    ],
    index=[
        "Bahrain",
        "Belgium",
        "Bulgaria",
        "Ireland",
        "Macedonia",
        "Norway",
        "Paraguay",
        "Singapore",
        "South Africa",
        "Switzerland",
    ],
)

t = 17
future_value = gdppercap * (1 + avg_growth / 100) ** t
ex8_gini_2025 = cal_gini_vector(future_value)
print(ex8_gini_2025)
results["ex8_gini_2025"] = ex8_gini_2025
results
```

```
0.3656264991306193
```

Out[ ]:
```
{'ex3_highest_gdp_percap': 'Norway',
 'ex3_lowest_gdp_percap': 'Paraguay',
 'ex4_lessthan20_000': 12393,
 'ex5_switzerland': 42536,
 'ex6_gini_loop': 0.3382798461272245,
 'ex7_gini_vectorized': 0.3382798461272245,
 'ex8_gini_2025': 0.3656264991306193}
```

Exercise 9

Interpret your result -- does it seem to imply that we are seeing covergence or not?

After you're done, you can see a more systematic version of this analysis here!

Answer for number 9: We are no seeing a convergence. This is because we can see that the gap is not reducing.