Project: Eternal Growth, Immortal Inequality

Table of Contents

1. Introduction

1a. Data Source: Gapminder

Gapminder is, in their own words, "an independent educational non profit fighting global misconceptions." More specifically they collect myriads of indicators about the health, happiness, and economic welfare of nations as globally and as far back in time as possible. I will be analyzing their data related to wealth, inequality, human flourishing, and CO2 emissions.

1b. Data Bases

Real Gross Domestic Product per capita using Purchasing Power Parity (in 2017 US Dollars)

What: GDP stands for gross domestic product and is a measure of the total amount of economic value that a state produces in a year. GDP is important because it is almost synonymous with gross national income (GNI) or how much total money that state's population can spend that year.

Dividing GDP by population gives you the GDP per capita, which tells you roughly much money the average individual in that state has at their disposal.

Real GDP means that we have adjusted the GDP of each year for changes in inflation which makes it easier to compare the **real** changes in GDP across periods of time.

Finally, Purchasing Power Parity measure how much a dollar in each country can buy relative to how much a dollar can buy in the US. This is done by measuring the price of a batch of goods in each country, then adjusting it for that difference. For example, if that batch of goods is **cheaper** in India than it would be in the US, we would modify the GDP per capita

number **upwards** to represent that the dollar buys more in that country. This is useful for understanding how rich the average person feels in a country, which is what I was interested in investigating so I used it for this analysis.

In summary: The higher the Real GDP per capita PPP of a nation, the richer the average citizens of that nation.

Sources: This dataset is collected from a many sources including the World Bank, see documentation.

Gini Coefficient (0-100)

What: The Gini Coefficient is an economic metric for quantifying how much income inequality exists in a society with 0% representing 'perfectly equal' and 100% being 'completely unequal'. Briefly, the Gini Coefficient is the difference between a hypothetical society with perfectly equal income distribution (where for instance each quartile of the population has a 25% of the total income of that country) vs the actual distribution of incomes in that country (where for instance the highest quartile control receive 70% of that nations total income, and the lowest quartile control 1%). How to calculate the Gini Coefficient is described in footnote 1.

Sources: This dataset is collected by the World Bank.

Happiness (0-100)

What: From Gapminder:

> This is the national average response to the question of life evaluations asking the following “Please imagine a ladder, with steps numbered from 0 at the bottom to 10 at the top. The top of the ladder represents the best possible life for you and the bottom of the ladder represents the worst possible life for you. On which step of the ladder would you say you personally feel you stand at this time?” This measure is also referred to as Cantril life ladder. Gapminder has converted this indicator's scale from 0 to 100 to easily communicate it in terms of percentage.

Sources: This dataset is collected by the Gallup World Poll organization.

Murder Rate per 100,000 people

What: Mortality or deaths due to interpersonal violence per 100,000 people in a population.

Sources: This dataset is collected by Global Burden of Disease

The Freedom Index (1-7)

What: From Gapminder

> Freedom index is the average of political rights and civil liberties ratings and is used to determine countries' freedom statuses. It's range on a scale of 1 (most free) to 7 (least free).

Sources: This dataset is collected by Freedom in the World of Freedom House.

CO2 Emissions per capita (in metric tonnes)

What: This dataset is a measure of the total metric tonnes of carbon dioxide emissions from a nation divided by its population.

Sources: This dataset is collected and provided by the Carbon Dioxide Information Analysis Center.

1c. Questions

The Welfare of Nations

Question 1. Is an increase in GDP per capita is associated with increases in happiness, safety, and freedom?

Historically, GDP per capita has robustly correlated with improvements in health outcomes, education, and consumption of products, but what about other measures of human flourishing like happiness, safety, and freedom? The markers I choose to evaluate this are self-reported happiness, deaths to violence per 100,000 people, and the freedom index from the Gapminder's databases. It is important to better understand the role of stimulating economic growth as a means of improving peoples' lives.

Rich Man, Poor World

Question 2. Is an increase in GDP per capita associated with an increase in CO2 emissions per capita?

Even if GDP is often associated with improvements in human welfare, it is also important to understand the costs of increased productivity and wealth. One cost that has become especially salient in recent years is the impact of human activity on global warming in the form of anthropomorphic climate change due to greenhouse gas emissions. Does increasing GDP per capita meaning increasing the CO2 emissions per capita?

Progress and Poverty

Question 3. Is an increase in GDP per capita associated with an increase in economic inequality?

Another potential problem with the exponential growth of wealth in recent years is the seeming growth of inequality. Are the wealthiest nations in the world also the most unequal? To answer this question, I will explore the relationship between economic wealth and its evil twin, economic inequality, as measured by the Gini Coefficient (described in the data bases section).

The Gini's Curse

Question 4. Is an increase in economic inequality associated with a decrease in happiness, safety, and freedom?

Many people think that economic inequality is related to many of societies woes, but is it? Here is where I will look at the association between the Gini Coefficient and human welfare metrics like happiness, safety, and freedom.

1d. Disclaimer

I will only be looking at correlations and doing only basic statistical assessments of the data. No causal conclusions can be made from this analysis.

For 'safety' I will be using declining murder rates as a proxy.
For 'economic inequality' I will be using the Gini Coefficient as a proxy.

2. Data Wrangling

2a. Initialization and Assessment

In this subsection I imported all the libraries that I would use and loaded in my six datasets. I used for loops to explore the six dataframes and would often look at the excess output in a text file.

In [ ]:
```
# Libraries to use
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set_style('darkgrid')

import warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```
# Economic Indicators
gdp_df = pd.read_csv("Data/gdppercap.csv", header=1)
gini_df = pd.read_csv('Data/inequality_index_gini.csv')
```

```
# Wellbeing Indicators
hap_df = pd.read_csv("Data/happiness.csv")
murder_df = pd.read_csv("Data/murder_per_100000_people.csv")
free_df = pd.read_csv("Data/freedix_fh.csv")

# Environmental Indicators
co2_df = pd.read_csv("Data/co2_emissions_tonnes_per_person.csv")
```

Adding New Dataframe Considerations:

This wasn't specifically written with massive generalizability in mind --- however I would like it to generalize somewhat. Thus, I have denoted all the places in the code that require edits/checks/add-ons here before adding any other dfs to this project.

1. Add df to df_ls list, and the other title lists
2. check dtypes, make sure the df being added ONLY has float or int data,
3. Make sure the columns are ONLY years (as strings), be aware of what years you add!
4. check the overlap of country names in each df
5. make sure the new df can be broken down into quintiles!

1. **There are a couple functions that are very picky about dataframe setup, make sure to check all the dots and comments**

In [ ]:
```
# new dfs
# Sociological Indicators
diversity_df = pd.read_csv("Data/diversity.csv", index_col=4)
```

In [ ]:
```
diversity_df = diversity_df.iloc[:,13].to_frame().rename(columns={'ethnicFract'

diversity_df.head()
```

Out[ ]:

	2005
country	
Uganda	0.9302
Liberia	0.9084
Madagascar	0.8791
DR Congo	0.8747
Republic of the Congo	0.8747

In [ ]:
```
# Commonly used Lists
df_ls = [gdp_df, gini_df, hap_df, murder_df, free_df, co2_df, diversity_df] # a
colors = ['black', 'brown', 'darkgoldenrod', 'blue', 'green']
figure_number = 1

# commonly title lists
pretty_title = ['GDP per person', # add df here
 'Gini coefficient',
 'Happiness',
 'Murder rate',
 'Freedom Index',
```

```
 'Co2 emissions per person',
 'Diversity']

column_title = ['gdp_per_person',
 'gini',
 'happiness',
 'murder_rate',
 'freedom_index',
 'co2_emissions_per_person',
 'diversity']

axis_title = ['GDP per person (USD $)',
 'Gini Coefficient (lower is better)',
 'Happiness (higher is better)',
 'Murders per 100,000 people',
 'Freedom Index (lower is better)',
 'CO2 Emissions per person (in metric tons)',
 'Diversity (Ethnic Fractionalization)']
```

In [ ]:

```
# Using head to look at all of the dataFrames
for i in range(0, len(pretty_title)):
    print(f'{pretty_title[i]} df head():\n', df_ls[i].head(1))
```

```
GDP per person df head():
    geo Country Name   1800   1801   1802   1803   1804   1805   1806   1807
\
0  afg  Afghanistan  683.0  683.0  683.0  683.0  683.0  683.0  683.0  683.0

   ...    2041    2042    2043    2044    2045    2046    2047    2048  \
0  ...  2689.0  2747.0  2806.0  2866.0  2928.0  2991.0  3056.0  3122.0

     2049    2050
0  3189.0  3257.0

[1 rows x 253 columns]
Gini coefficient df head():
   country  1966  1967  1968  1969  1970  1971  1972  1973  1974  ...  2011  \
0  Angola    NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN  ...   NaN

   2012  2013  2014  2015  2016  2017  2018  2019  2020
0   NaN   NaN   NaN   NaN   NaN  51.3   NaN   NaN   NaN

[1 rows x 56 columns]
Happiness df head():
        country  2004  2005  2006  2007  2008  2009  2010  2011  2012  2013  \
0  Afghanistan   NaN   NaN   NaN  37.2  44.0  47.6  38.3  37.8  35.7  31.3

   2014  2015  2016  2017  2018  2019  2020
0  39.8  42.2  26.6  26.9  23.8   NaN  24.0
Murder rate df head():
    country  1949  1950  1951  1952  1953  1954  1955  1956  1957  ...  2006
\
0  Albania   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN  ...  3.07

   2007  2008  2009  2010  2011  2012  2013  2014  2015
0  2.96   NaN   2.3   NaN   NaN   NaN   NaN   NaN   NaN

[1 rows x 68 columns]
Freedom Index df head():
        country  1971  1972  1973  1974  1975  1976  1977  1978  1979  ...  \
0  Afghanistan   4.5   6.5   6.5   6.5   6.5   6.0   7.0   7.0   7.0  ...

   2010  2011  2012  2013  2014  2015  2016  2017  2018  2019
0   6.0   6.0   6.0   6.0   6.0   6.0   5.5   5.5   5.5   5.5

[1 rows x 50 columns]
Co2 emissions per person df head():
        country  1799  1800  1801  1802  1803  1804  1805  1806  1807  ...  \
0  Afghanistan   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN  ...

    2008  2009   2010   2011  2012   2013   2014   2015   2016   2017
0  0.238  0.29  0.406  0.345  0.28  0.253  0.262  0.245  0.247  0.254

[1 rows x 220 columns]
Diversity df head():
            2005
country
Uganda   0.9302
```

In [ ]:
```
# for loop for looking at all of the info()s
# for i in range(0, len(pretty_title)):
```

```
#     print(f'\n{pretty_title[i]} df info():')
#     df_ls[i].info()
```

2b. Cleaning up Extra Columns and Names

Drops:
This brief look at the datasets showed me that the gdp_df and co2_df had dates going all the way back to 1799 and even as far out as 2050, however none of the other datasets went back or forward that far. Also after looking through some of the documentation of gapminder, I also realized that the more recent data would likely be more accurate. Due to these two reasons I decided to drop the data before 1900 from both datasets and the data beyond 2020.

Renaming:
I wanted to make sure that each dataset had closely matching indexes because I would often be looking to compare the data about countries between them. To do this I had to rename the country column of each dataset to 'country' and set that as the index. Then I had to find countries that didn't overlap with each other, and try to make them match up as much as possible.

In [ ]:
```
# First I got rid of some of the extra columns in GDP and CO2 df that aren't ir
gdp_df.drop(columns='geo', inplace=True)
gdp_df.drop(columns=gdp_df.columns[222:], inplace=True)
gdp_df.drop(columns=gdp_df.columns[1:101], inplace=True)

co2_df.drop(columns=co2_df.columns[1:102], inplace=True)

df_ls = [gdp_df, gini_df, hap_df, murder_df, free_df, co2_df, diversity_df] # a
```

In [ ]:
```
# # Then I cleaned up the indexes of the dataframes to make them uniform - i us

for l in df_ls[:6]: # will need to make sure that the columns of your already s
    l.rename(columns={l.columns[0]:'country'}, inplace=True)
    l.set_index('country', inplace=True)

    l.astype(np.number)
    print(l.shape, l.dtypes.unique(), l.count().sum())
```

```
(204, 121) [dtype('float64')] 24321
(168, 55) [dtype('float64')] 1833
(163, 17) [dtype('float64')] 2067
(118, 67) [dtype('float64')] 3166
(194, 49) [dtype('float64')] 8534
(194, 118) [dtype('float64')] 15866
```

```
C:\Users\Samsickle\AppData\Roaming\Python\Python39\site-packages\pandas\core\d
types\common.py:1691: DeprecationWarning: Converting `np.inexact` or `np.float
ing` to a dtype is deprecated. The current result is `float64` which is not st
rictly correct.
  npdtype = np.dtype(dtype)
C:\Users\Samsickle\AppData\Roaming\Python\Python39\site-packages\pandas\core\d
types\common.py:1691: DeprecationWarning: Converting `np.inexact` or `np.float
ing` to a dtype is deprecated. The current result is `float64` which is not st
rictly correct.
  npdtype = np.dtype(dtype)
C:\Users\Samsickle\AppData\Roaming\Python\Python39\site-packages\pandas\core\d
types\common.py:1691: DeprecationWarning: Converting `np.inexact` or `np.float
ing` to a dtype is deprecated. The current result is `float64` which is not st
rictly correct.
  npdtype = np.dtype(dtype)
C:\Users\Samsickle\AppData\Roaming\Python\Python39\site-packages\pandas\core\d
types\common.py:1691: DeprecationWarning: Converting `np.inexact` or `np.float
ing` to a dtype is deprecated. The current result is `float64` which is not st
rictly correct.
  npdtype = np.dtype(dtype)
C:\Users\Samsickle\AppData\Roaming\Python\Python39\site-packages\pandas\core\d
types\common.py:1691: DeprecationWarning: Converting `np.inexact` or `np.float
ing` to a dtype is deprecated. The current result is `float64` which is not st
rictly correct.
  npdtype = np.dtype(dtype)
C:\Users\Samsickle\AppData\Roaming\Python\Python39\site-packages\pandas\core\d
types\common.py:1691: DeprecationWarning: Converting `np.inexact` or `np.float
ing` to a dtype is deprecated. The current result is `float64` which is not st
rictly correct.
  npdtype = np.dtype(dtype)
```

This went smoothly. Now I wanted to attempt to have as much of each dataframes index of countries match as much as possible. The GDP index had the largest index of all the dataframes, so I checked it against the other indexes.

In [ ]:
```
# Assessing Index of GDP
gdp_df.head()
```

Out[ ]:

	1900	1901	1902	1903	1904	1905	1906	1907	1908	1909	...
country											
Afghanistan	1054.0	1063.0	1071.0	1080.0	1089.0	1097.0	1104.0	1112.0	1120.0	1127.0	...
Albania	1269.0	1286.0	1303.0	1320.0	1337.0	1354.0	1372.0	1390.0	1408.0	1426.0	...
Algeria	1861.0	1882.0	1903.0	1925.0	1947.0	1969.0	1991.0	2014.0	2037.0	2060.0	...
Andorra	3617.0	3660.0	3704.0	3747.0	3792.0	3838.0	3883.0	3929.0	3976.0	4023.0	...
Angola	779.0	778.0	778.0	777.0	777.0	776.0	776.0	775.0	774.0	774.0	...

5 rows × 121 columns

In [ ]:
```
# for i in range(1,len(pretty_title)):
#     print(f"Not Overlapping in Both GDP index and {pretty_title[i]}:", set(di
#     print(f"Things not in GDP index that are in {pretty_title[i]}:", set(df_i

# you should check this output too for your new df.
```

This assessment showed me that there were several extra indexes in the GDP dataframe compared to the others. For fixing the indexes I would remove the indexes "NaN" and "Name" from the gdp index. Also a brief google revealed that I should update the names 'Swaziland' to 'Eswatini' and 'Macedonia, FYR' to 'North Macedonia' to make them more current and accurate. Finally, I renamed 'West Bank and Gaza' to 'Palestine' in the GDP dataframe for simple uniformity and since they refer to the same area.

I decided to keep the extra indexes of 'Europe', 'The Americas', 'World', 'Hong Kong, China', 'Africa', 'Taiwan', 'Asia' in the GDP dataframe because they could provide interesting information later, even if they had no analogous entry in the other datasets so I wouldn't be able to look at their correlation with the other dataframes.

In [ ]:
```
# Before renaming the indexes, I wanted to set a variable to the one of the row
before = gdp_df.loc['Swaziland'].copy()
```

In [ ]:
```
# I dropped some of the rows from the gdp_df that were not in any other df: "Na
gdp_df.drop(index=gdp_df.index[197:199], inplace=True)

# I went about renaming some of the discrepancies in the indexes
gdp_df.rename(index={'Swaziland':'Eswatini', 'Macedonia, FYR':'North Macedonia'
gini_df.rename(index={'Swaziland':'Eswatini', 'Macedonia, FYR':'North Macedonia
```

In [ ]:
```
# Here is my check to see if the row I changed was changed correctly
print((before == gdp_df.loc['Eswatini', :]).all())
```

```
True
```

In [ ]:
```
# I ran my check again to see if the indexes were now uniform
# for i in range(1,len(pretty_title)):
#     print(f"Not Overlapping in Both GDP index and {pretty_title[i]}:", set(di
#     print(f"Things not in GDP index that are in {pretty_title[i]}:", set(df_i

df_ls = [gdp_df, gini_df, hap_df, murder_df, free_df, co2_df, diversity_df] # a

# you should check this output too for your new df
```

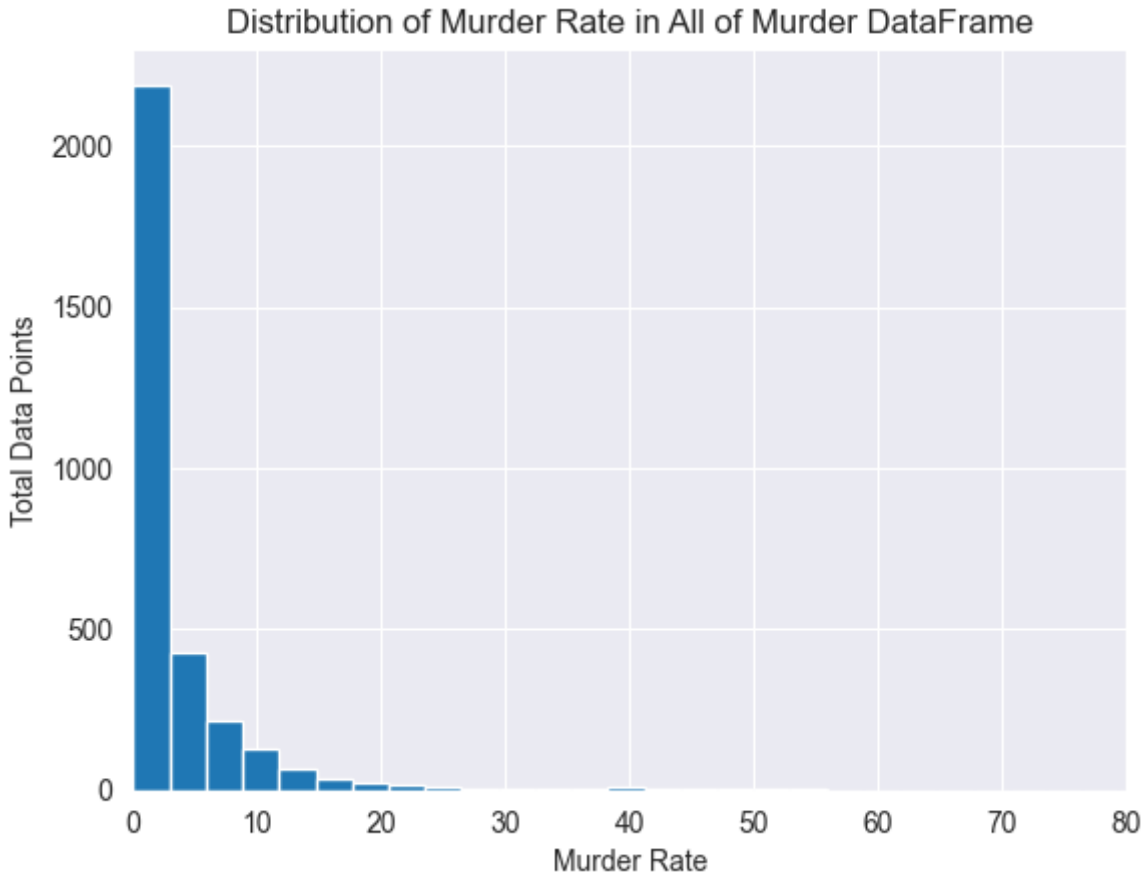
Kosovo still remained unnamed in the GDP dataset, but that seemed ok since there was only one measure and it only had data in the gini_df. However, there was a large set of non intersecting indexes between the GDP dataframe and the Murder Rate dataframe. I wanted to assess how much that data deviated from the other data that I would be able to assess.

In [ ]:
```
# check the distribution of the unmatched columns in murder_df vs the other nat
# make all the plots share a y axis
# plt 1
plt.hist(murder_df.stack().reset_index().iloc[:,2], bins=30)

plt.xlim(0, 80)
```

```
plt.title('Distribution of Murder Rate in All of Murder DataFrame')
plt.xlabel('Murder Rate')
plt.ylabel('Total Data Points')
```

Out[ ]:
```
Text(0, 0.5, 'Total Data Points')
```

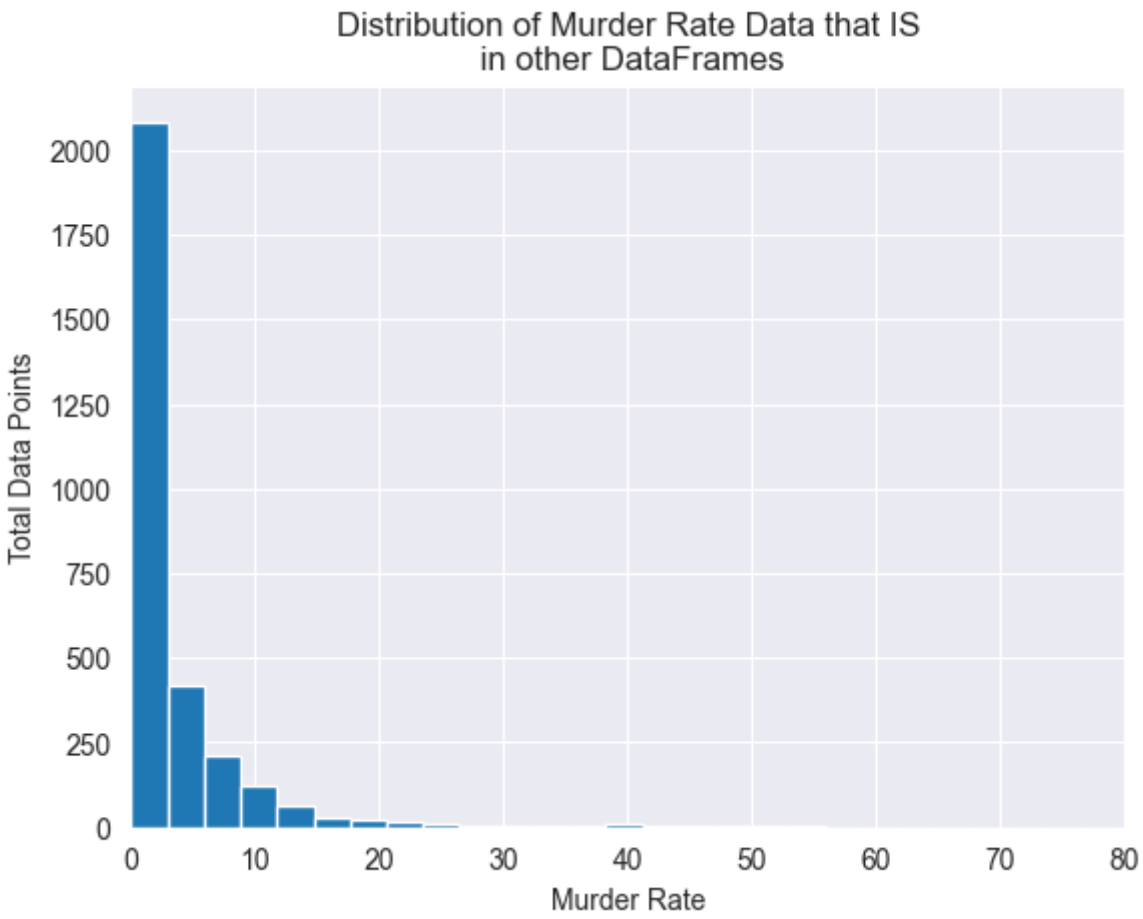


In [ ]:
```
# plt 2
unmatched = {'Cayman Islands', 'Martinique', 'Virgin Islands (U.S.)', 'Bermuda'
murder_df.loc[list(set(murder_df.index)-unmatched)].stack().reset_index().iloc[

plt.xlim(0, 80)


plt.title('Distribution of Murder Rate Data that IS\n in other DataFrames')
plt.xlabel('Murder Rate')
plt.ylabel('Total Data Points')
```

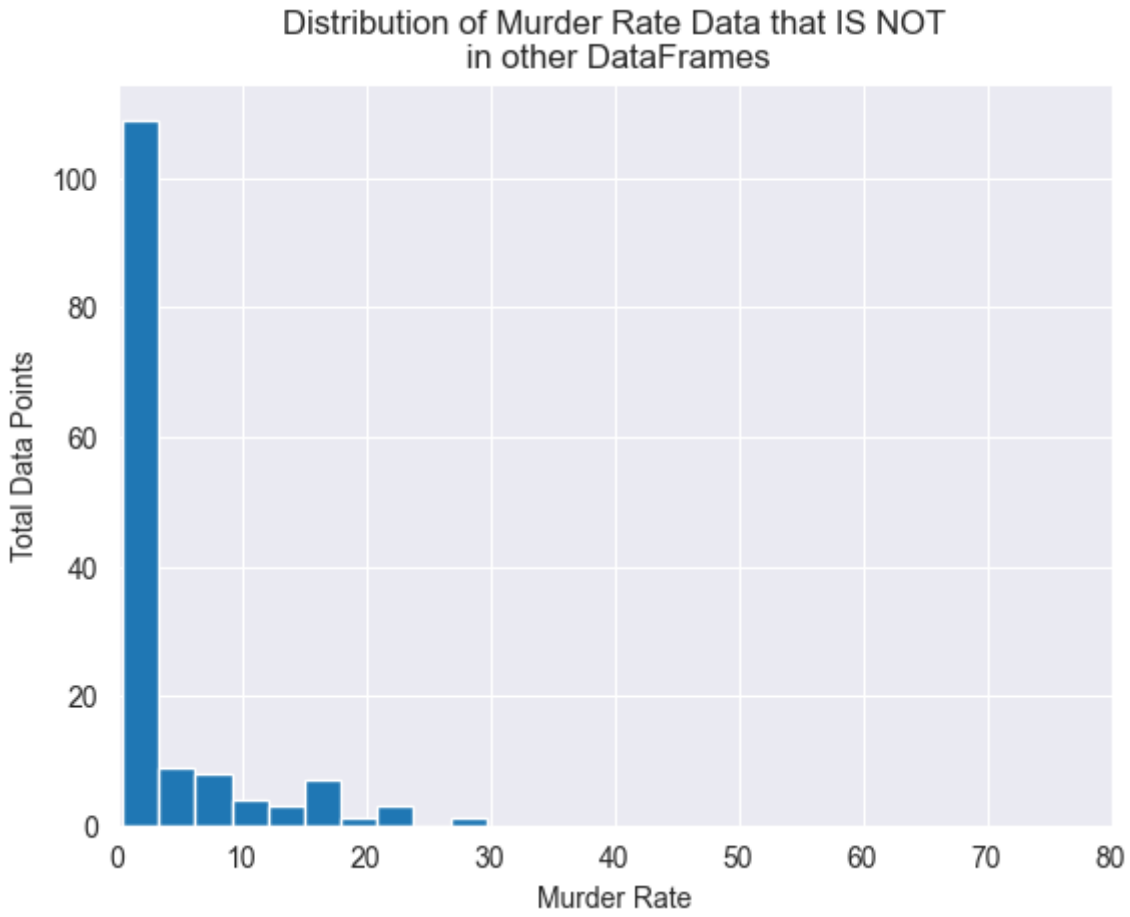
Out[ ]:
```
Text(0, 0.5, 'Total Data Points')
```



In [ ]:
```
# plt 3
murder_df.loc[list(unmatched)].stack().reset_index().iloc[:,2].hist(bins=10)
plt.xlim(0, 80)

plt.title('Distribution of Murder Rate Data that IS NOT\n in other DataFrames')
plt.xlabel('Murder Rate')
plt.ylabel('Total Data Points')
```

Out[ ]:
```
Text(0, 0.5, 'Total Data Points')
```



In [ ]:
```
# I saw that the USSR and Russia were both in the Murder data set and wanted to
murder_df.loc[['USSR', 'Russia']].T.plot();

plt.title('Times Series Line Plot of USSR vs Russian Murder Rate')
plt.xlabel('Years')
plt.ylabel('Murder Rate')
```

Out[ ]:
```
Text(0, 0.5, 'Murder Rate')
```

These visualizations led me to conclude that the relative similarity of these indexes to the included indexes would not impact my final evaluation of the correlation between GDP and murder rates. Also the difference between 'Russia' and 'USSR' were both very small and likely due to the inclusion of other non-russian territories in the 'USSR'. The GDP index already had these territories though in the form of independent states, so I didn't think it would greatly impact my final conclusions.

With this I had finished my cleaning and data type fixing. However I still needed to put the dataframes into a format that would be make it slightly easier to manipulate and compare factors.

2c. Restructuring and New Dataframes

Structure:

After doing all these steps, I then had to restructure my data to be easier to analyze. This involved making a dataframe with all of the information from the other dataframes.

In [ ]:
```
# # Create stacked multi-df
# stack each dataframe, and reset columns, and make a copy
stacked_df = pd.DataFrame(df_ls[0].stack().reset_index().copy())

# add every other stacked dataframe to this original one
for i in range(1, len(pretty_title)):
```

```
    stack = pd.DataFrame(df_ls[i].stack().reset_index().copy())

    stacked_df = pd.merge(stacked_df, stack, on=['country', 'level_1'], \
        how='outer', suffixes=[column_title[i-1], column_title[i]])
```

In [ ]:
```
# if you don't have a year column in the new df, your merges will fail here
stack_column_title = ['country','year']
stack_column_title.extend(column_title)

stacked_df.columns = stack_column_title

stacked_df.head()
```

Out[ ]:

	country	year	gdp_per_person	gini	happiness	murder_rate	freedom_index	co2_emiss
0	Afghanistan	1900	1054.0	NaN	NaN	NaN	NaN	
1	Afghanistan	1901	1063.0	NaN	NaN	NaN	NaN	
2	Afghanistan	1902	1071.0	NaN	NaN	NaN	NaN	
3	Afghanistan	1903	1080.0	NaN	NaN	NaN	NaN	
4	Afghanistan	1904	1089.0	NaN	NaN	NaN	NaN	

In [ ]:
```
stacked_df['year'] = pd.to_numeric(stacked_df['year'])

stacked_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24424 entries, 0 to 24423
Data columns (total 9 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   country                 24424 non-null  object 
 1   year                    24424 non-null  int64  
 2   gdp_per_person          24200 non-null  float64
 3   gini                    1833 non-null   float64
 4   happiness               2067 non-null   float64
 5   murder_rate             3166 non-null   float64
 6   freedom_index           8534 non-null   float64
 7   co2_emissions_per_person  15866 non-null  float64
 8   diversity               216 non-null    float64
dtypes: float64(7), int64(1), object(1)
memory usage: 1.7+ MB
```

Quintiles

I also wanted to sort each nation into five different quintiles for each metric. This would let me notice if there was a specific trend in the top 80th percentile of happy nations or the lowest 20th percentile of unequal nations. To do this I averaged each country's total score over all time, then broke them into quintiles, then columns to denote these quintiles in each dataframe.

After showing this to a couple people I realized that peoples intuitions for richest countries actually diverge a little. Some people thought the bins should be sorted;

1. According to how they were doing from long ago (starting rank)
2. According to how they are doing now (ending rank)
3. **According to their absolute average over all time (which essentially weighs more recent GDP growth more than past GDP growth)**
4. According to their relative average position compared to other countries over all time

I originally only did 3. I realized after a discussion that 4 would actually match some people's intuitions better, and some people just were curious about 1 and 2.

In [ ]:
```
# Create quintiles for each country
quintile_labels = ['0-20th', '20-40th','40-60th', '60-80th', '80-100th']

pd.qcut(df_ls[1].mean(axis=1, numeric_only=True), q=5, labels=quintile_labels).
```

Out[ ]:
```
country
Angola                  80-100th
Albania                   0-20th
United Arab Emirates      0-20th
Argentina                60-80th
Armenia                   0-20th
dtype: category
Categories (5, object): ['0-20th' < '20-40th' < '40-60th' < '60-80th' < '80-10
0th']
```

In [ ]:
```
# label each country with a quintile of each df
# then merge those labels onto the stacked df, and the original dfs

for i in range(len(pretty_title)):
    # find the mean over all time of each country in each dataframe
    mean = df_ls[i].mean(axis=1, numeric_only=True)

    # use pd.qcut to make a column with all the country names
    quintile_series = pd.Series(pd.qcut(mean, q=5, labels=quintile_labels), nam

    # merge or add each series to every dataframe and the stacked_df
    for j in df_ls:
        j[f'{column_title[i]}_quintile'] = quintile_series

    stacked_df = pd.merge(stacked_df, quintile_series, on='country', how='left'

# I also reassigned df_ls in case there were any changes I didn't account for i
# make sure to add df here also
df_ls = [gdp_df, gini_df, hap_df, murder_df, free_df, co2_df, diversity_df]

# then finally
stacked_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24424 entries, 0 to 24423
Data columns (total 16 columns):
 #   Column                              Non-Null Count  Dtype   
---  ------                              --------------  -----   
 0   country                             24424 non-null  object  
 1   year                                24424 non-null  int64   
 2   gdp_per_person                      24200 non-null  float64 
 3   gini                                1833 non-null   float64 
 4   happiness                           2067 non-null   float64 
 5   murder_rate                         3166 non-null   float64 
 6   freedom_index                       8534 non-null   float64 
 7   co2_emissions_per_person            15866 non-null  float64 
 8   diversity                           216 non-null    float64 
 9   gdp_per_person_quintile             24200 non-null  category
 10  gini_quintile                       20219 non-null  category
 11  happiness_quintile                  19723 non-null  category
 12  murder_rate_quintile                12735 non-null  category
 13  freedom_index_quintile              23389 non-null  category
 14  co2_emissions_per_person_quintile   23389 non-null  category
 15  diversity_quintile                  22361 non-null  category
dtypes: category(7), float64(7), int64(1), object(1)
memory usage: 1.8+ MB
```

3. Exploratory Data Analysis

3a. Making Functions for Graphing

I wanted to make three functions that would take in the variables I wanted to compare and output some type of graph investigating the relationship between all the variables. It was important to keep track of which columns I would be manipulating from my stacked_df so I put a list of them here to keep a reference. I also used this list as an index for each factor I was investigating.

1. gdp_per_person
2. gini
3. happiness
4. murder_rate
5. freedom_index
6. co2_emissions_per_person

I would eventually need to be able to find the first and last years between two dataframes I was working with. So I created three functions to find all the years in a df, the last year between two dfs, and the first year between two dfs

In [ ]:
```
# create list of years in each df
def years_only(df):
    yrs = []
    for i in df:
        try:
```

```
            yrs.append(int(i))
        except:
            continue
    return np.array(yrs)

# check your new df here too
years_only(df_ls[6])
```

Out[ ]:
```
array([2005])
```

In [ ]:
```
# find the last year that is in both dfs
def max_year(df1, df2):
    years1 = years_only(df1).max()
    years2 = years_only(df2).max()
    last_yr = np.array([years1, years2]).min()
    return last_yr

# check your new df here too
max_year(df_ls[0], df_ls[6]) # right answer is 2017
```

Out[ ]:
```
2005
```

In [ ]:
```
# find the first year that is in both dfs
def min_year(df1, df2):
    years1 = years_only(df1).min()
    years2 = years_only(df2).min()
    first_yr = np.array([years1, years2]).max()
    return first_yr

min_year(df_ls[2], df_ls[1]) # right answer is 2004

stacked_df.head()
```

Out[ ]:

	country	year	gdp_per_person	gini	happiness	murder_rate	freedom_index	co2_emiss
0	Afghanistan	1900	1054.0	NaN	NaN	NaN	NaN	
1	Afghanistan	1901	1063.0	NaN	NaN	NaN	NaN	
2	Afghanistan	1902	1071.0	NaN	NaN	NaN	NaN	
3	Afghanistan	1903	1080.0	NaN	NaN	NaN	NaN	
4	Afghanistan	1904	1089.0	NaN	NaN	NaN	NaN	

First Function:

scatter_wellbeing(independent_variable=0, dependent_variables=[2,3,4], high_or_low=True)

The first function would take one independent variable and any amount of comparison dependent variables, then output a scatter plot of their relationship. This function also categorizes each country into quintiles of the independent variable and colors each point in the scatter plot with that country's respective quintile. The high or low parameter lets the user specify whether it is good for the independent variable to be high (the default), or

whether it was good for the independent variable to be low in which case one would have to set the parameter of `high_or_low = False`.

In [ ]:
```
# Plot the scatter plot relationship between a chosen metric and another metric

def scatter_wellbeing(independent_variable=0, dependent_variables=[2,3,4], high
    # setting the ordering of colors
    if high_or_low:
        scatter_colors = colors
    else:
        scatter_colors = colors[::-1]

    # figsize dependent on the amount of dependent_variables
    plt.figure(figsize=[7*len(dependent_variables),6])

    # assign variables to represent the independent_variable's stacked_df colum
    # quintile column, and a list of each unique quintile in the quintile colum
    quintile_column_number =  independent_variable + len(column_title) + 2
    stack_column_number = independent_variable + 2
    quintiles_list = sorted(stacked_df.iloc[:, quintile_column_number].dropna()

    # set up subplots dependent on the amount of comparison variables
    for plot_number, comparison_v in enumerate(dependent_variables):
        plt.subplot(1, len(dependent_variables), (plot_number+1))

        # plot each quintile SEPARATELY on the same subplot using this for loop
        for color_num, single_quintile in enumerate(quintiles_list):
            quintile = stacked_df[stacked_df.iloc[:, quintile_column_number]==s
            plt.scatter(x=quintile.iloc[:,stack_column_number], \
                y= quintile.iloc[:, comparison_v+2], marker='.', \
                alpha=.3, color=scatter_colors[color_num])

        # get min and max year of each subplot
        last_year = max_year(df_ls[comparison_v], df_ls[independent_variable])
        first_year = min_year(df_ls[comparison_v], df_ls[independent_variable])

        # decorate each subplot
        plt.title(f'Relationship between {pretty_title[independent_variable]} a

        plt.ylabel(f'{axis_title[comparison_v]}', size=12)
        plt.xlabel(f'{axis_title[independent_variable]}\nFigure {figure_number}
        plt.legend([f'{q} quintile' for q in quintiles_list], title=f'Countries

        globals()['figure_number'] += 1
```

Second Function:

time_series_quintile(independent_variable=0, dependent_variables=[2,3,4], high_or_low=True)

The second function takes one independent variable, and any amount of dependent variables and outputs a line chart. This function categorizes each country into different quintiles of the independent variable, then averages the dependent variables of all the countries in each quintile into one number per year. The line chart finally plots each

quintiles' average over time. The high or low parameter lets the user specify whether it is good for the independent variable to be high (the default), or whether it was good for the independent variable to be low in which case one would have to set the parameter of `high_or_low = False`.

In [ ]:
```
# Plot the relationship between a chosen metric's quintile range and those quir

def time_series_quintile(independent_variable=0, dependent_variables=[2,3,4], h
    # setting the ordering of colors
    if high_or_low:
        line_colors = colors
    else:
        line_colors = colors[::-1]

    # figsize dependent on the amount of dependent_variables
    plt.figure(figsize=[7*len(dependent_variables),6])

    # assign variables to represent the independent_variable's stacked_df quint
    # and a list of each unique quintile in the quintile column
    quintile_column_number =  independent_variable + 2 + len(pretty_title)
    quintiles_list = sorted(stacked_df.iloc[:, quintile_column_number].dropna()

    # cycle through the dependent_variables' dataframes and find the column of
    for plot_number, comparison_v in enumerate(dependent_variables):
        df = df_ls[comparison_v]

        # column of interest in each cycled through df
        quintile_column = df.columns[-len(column_title):][independent_variable]

        # the mean dependent_variables values for each quintile for each year
        # to make the years appear on the x axis (rows), a transpose was requir
        wellbeing_value = df.groupby(quintile_column).mean(numeric_only=True).T

        # plotting each quintiles progression through time from each dependent_
        plt.subplot(1, len(dependent_variables), (plot_number+1))
        plt.plot(wellbeing_value, alpha=.8)

        # change the color of each line using this for loop
        ax = plt.gcf().axes[plot_number]
        for color_num, lines in enumerate(ax.get_lines()):
            lines.set_color(line_colors[color_num])

        # get min and max year of each subplot
        last_year = max_year(df_ls[comparison_v], df_ls[independent_variable])
        first_year = min_year(df_ls[comparison_v], df_ls[independent_variable])

        # decorate each subplot
        plt.title(f'Average {pretty_title[comparison_v]} over time \nby Quintil
        plt.ylabel(f'Average {axis_title[comparison_v]}', size=12)
        plt.xlabel(f'Years\nFigure {figure_number}', size=12)
        plt.xticks(ticks=range(0, len(df.columns)-len(column_title), 5), labels
        plt.legend([f'{q} quintile' for q in quintiles_list], title = f'Countri

        globals()['figure_number'] += 1
```

Third Function:

quintile_wellbeing(independent_variable=0, dependent_variables=[2,3,4], high_or_low=True)

The third function takes one independent variable, and any amount of dependent variables and outputs a bar chart. It then categorizes all the countries into quintiles of the independent variable, then displays each quintiles' average of the dependent comparison metrics in a bar chart. The high or low parameter lets the user specify whether it is good for the independent variable to be high (the default), or whether it was good for the independent variable to be low in which case one would have to set the parameter of `high_or_low = False`.

In [ ]:
```
# Plot the relationship between a chosen metric's quintile ranges and those qui

def quintile_wellbeing(independent_variable=0, dependent_variables=[2,3,4], hig
    # setting the ordering of colors
    if high_or_low:
        quintile_colors = colors
    else:
        quintile_colors = colors[::-1]

    # figsize determined by amount of dependent_variables
    plt.figure(figsize=[7*len(dependent_variables),6])

    # assign variables to represent the independent_variable's quintile column
    # and a list of each unique quintile in the quintile column
    quintile_column_number =  independent_variable + 2 + len(column_title)
    quintiles_list = sorted(stacked_df.iloc[:, quintile_column_number].dropna()

    # assign variables to the average of all columns for each quintile
    quintile_grouping = stacked_df.groupby(stacked_df.iloc[:, quintile_column_r
    quintile_grouping = quintile_grouping.drop(columns='year')  # I needed to d

    # set up subplots and plot quintile_grouping on each subplot
    for plot_number, comparison_v in enumerate(dependent_variables):
        plt.subplot(1, len(dependent_variables), (plot_number+1))

        # plot each quintile SEPARATELY on the same subplot using this for loop
        for color_num, single_quintile in enumerate(quintiles_list):

            # when a single quintile is selected, the quintile_grouping is filt
            quintile_mean = quintile_grouping.loc[single_quintile].iloc[compari
            plt.bar(x=single_quintile, height= quintile_mean, alpha=.8, color=c

        # get min and max year of each subplot
        last_year = max_year(df_ls[comparison_v], df_ls[independent_variable])
        first_year = min_year(df_ls[comparison_v], df_ls[independent_variable])

        # decorate each subplot
        plt.title(f'Average {pretty_title[comparison_v]} over all time\nby Quir
        plt.ylabel(f'{axis_title[comparison_v]}', size=12)
        plt.xlabel(f'Quintile by {axis_title[independent_variable]}\nFigure {fi
        plt.xticks(ticks=range(len(quintiles_list)), labels= [f'{q}' for q in q

        globals()['figure_number'] += 1
```

3b. Question 1: GDP and Welfare Metrics

Scatter plot, time series line plot, bar chart

I explored the relationships the data using a scatter plot to look at the correlation of GDP with these wellbeing metrics, a time series line plot to observe how each GDP quintile's metrics on average have changed over time, and a bar chart to illustrate the average values of the richest to poorest quintiles over all time.

In [ ]:
```
scatter_wellbeing(0)
time_series_quintile(0)
quintile_wellbeing(0) # use quintiles consistently
```

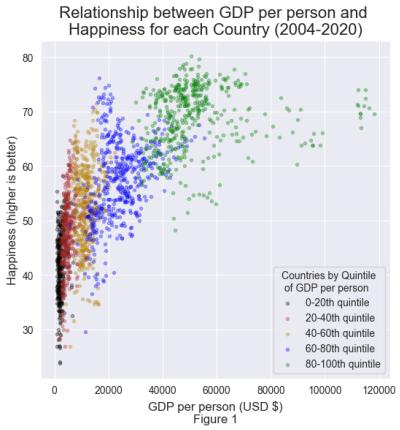


Figure 1

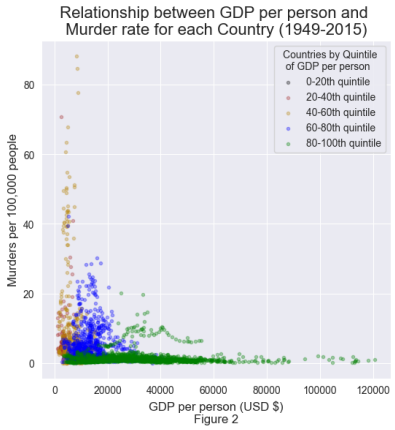


Figure 2

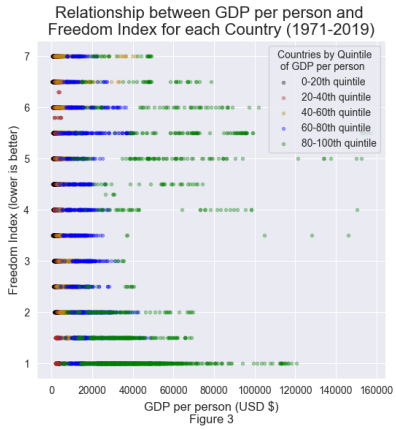


Figure 3

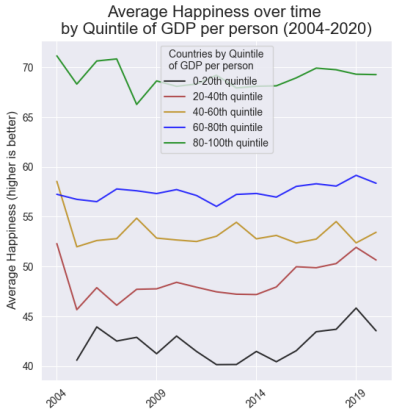


Figure 4

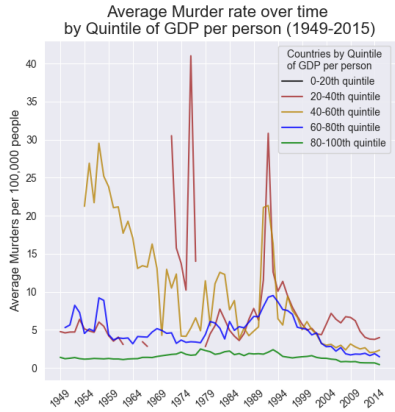


Figure 5

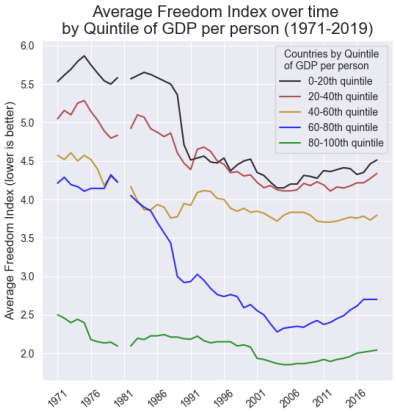


Figure 6

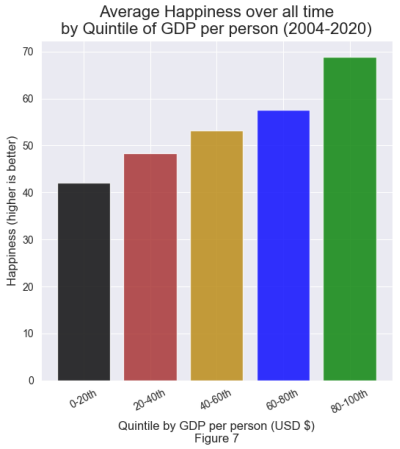


Figure 7

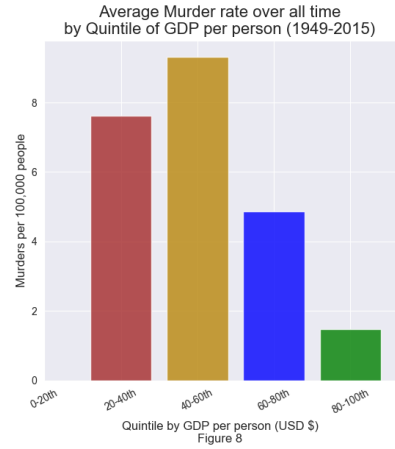


Figure 8

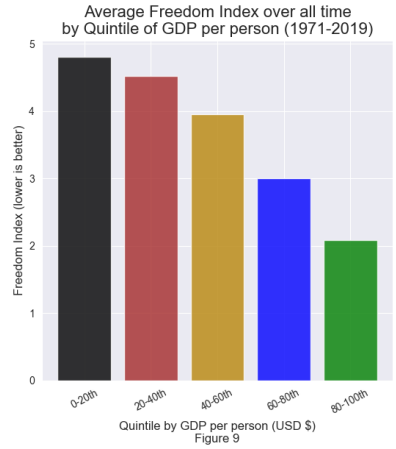


Figure 9

Comments:

This is pretty much what I wanted to explore. The scatter plot helped me visualize some of the relationships between GDP per capita and the other metrics, especially happiness. The freedom dataframe does not look very good or really give any intuitions in scatter plot form however.

The time series plot helped me visualizes changes in each metric over time, such as spikes in the data from one year to the next, or global trends shared between each quintile of countries. I originally tried grouping countries by deciles, however the time series line plot looked like a mess and didn't really help me understand what was happening in the different levels of wealth.
The time series data also revealed that there was not a single data point for murder rate in the lowest quintile of GDP countries. This surprised me. I wrote a little bit of code to better investigate what was happening here.

In [ ]:
```
# find number of years in murder df
murder_years = len(years_only(murder_df))

gdp_quint_murders = murder_df.groupby('gdp_per_person_quintile')\
    .count().iloc[:, :murder_years].sum(axis=1)

# find amount of murder rates data reported per quintile
print("Total data per", gdp_quint_murders)

# find average amount of murder rate data per year and per quintile
print("Average data per year in each", gdp_quint_murders/murder_years)
```

```
Total data per gdp_per_person_quintile
0-20th          0
20-40th       117
40-60th       420
60-80th       952
80-100th     1532
dtype: int64
Average data per year in each gdp_per_person_quintile
0-20th        0.000000
20-40th       1.746269
40-60th       6.268657
60-80th      14.208955
80-100th     22.865672
dtype: float64
```

This confirmed that there is not a single data point in the lowest quintile. Yikes! Also, it looks like even the 20-40th quintile also had very low rates of reporting. This explained some of the big spikes in the times series data of murder rate.

The bar charts were a nice way to visualize the average welfare metrics for each quintile over all time. In a way this was a dangerous simplification of the varying trends in the data, but it seemed like an important counter point to the exceptionally noisy time series plots

that let us get an idea of the general relationship each GDP quintile had with each metric of interest.

Before moving on to look at the actual correlations of the data I also wanted to do this same scatter, time series line plot, and bar chart of averages over all time for every other metric in question.

3c. Question 2: GDP and CO2

Scatter plot, time series line plot, bar chart

I used the same scatter plot, time series line plot, and bar chart of total averages for this too. Quintiles of GDP per capita are still created by using the average GDP per capita of each nation over all 120 years, then binning them into five equal subsections or quintiles.

In [ ]:
```
scatter_wellbeing(0, [5])
```

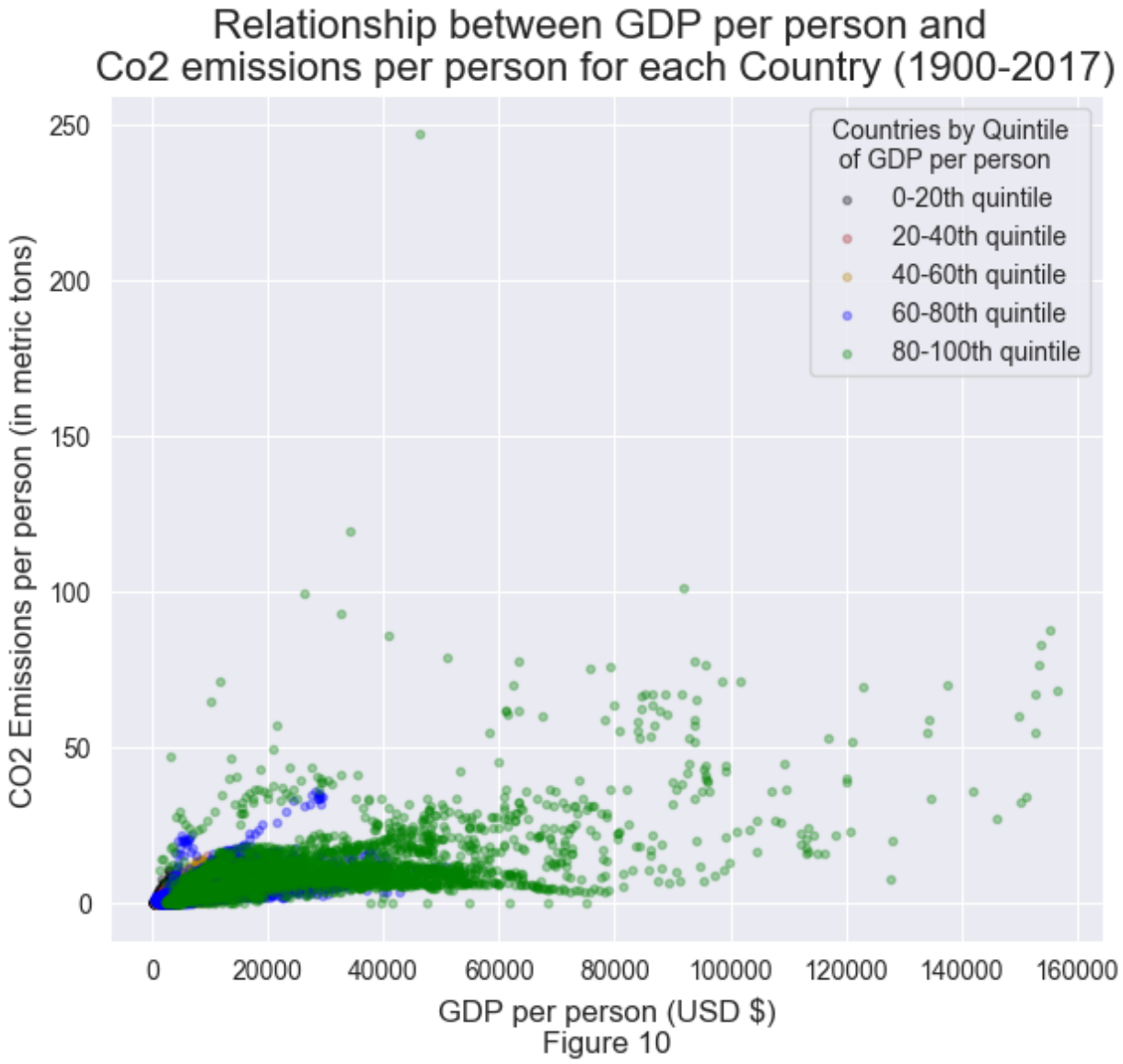


Figure 10

This scatter plot shows a pretty strong relationship but very clustered relationship. I figured that the outliers we making it hard to assess what was happening in the data (specifically the 250 CO2 emissions and the very rich countries above $120,000 GDP per capita), so I did the function again with a slightly trimmed down dataframes.

Unfortunately the functions I had written earlier made it somewhat difficult to change which dataframes were being inputted to the scatter plot function without changing the stacked dataframe itself. I did a somewhat hokey fix for this by setting a variable to hold the original stacked_df dataframe, then changing the stacked_df dataframe, calling the function, then resetting the stacked_df dataframe to its original values.

In [ ]:
```
# set original to a variable and use copy
stacked_df_original = stacked_df.copy()

# edit stacked df with query calls
stacked_df = stacked_df.query('gdp_per_person < 100000 and co2_emissions_per_pe

# call function
scatter_wellbeing(0, [5])
```

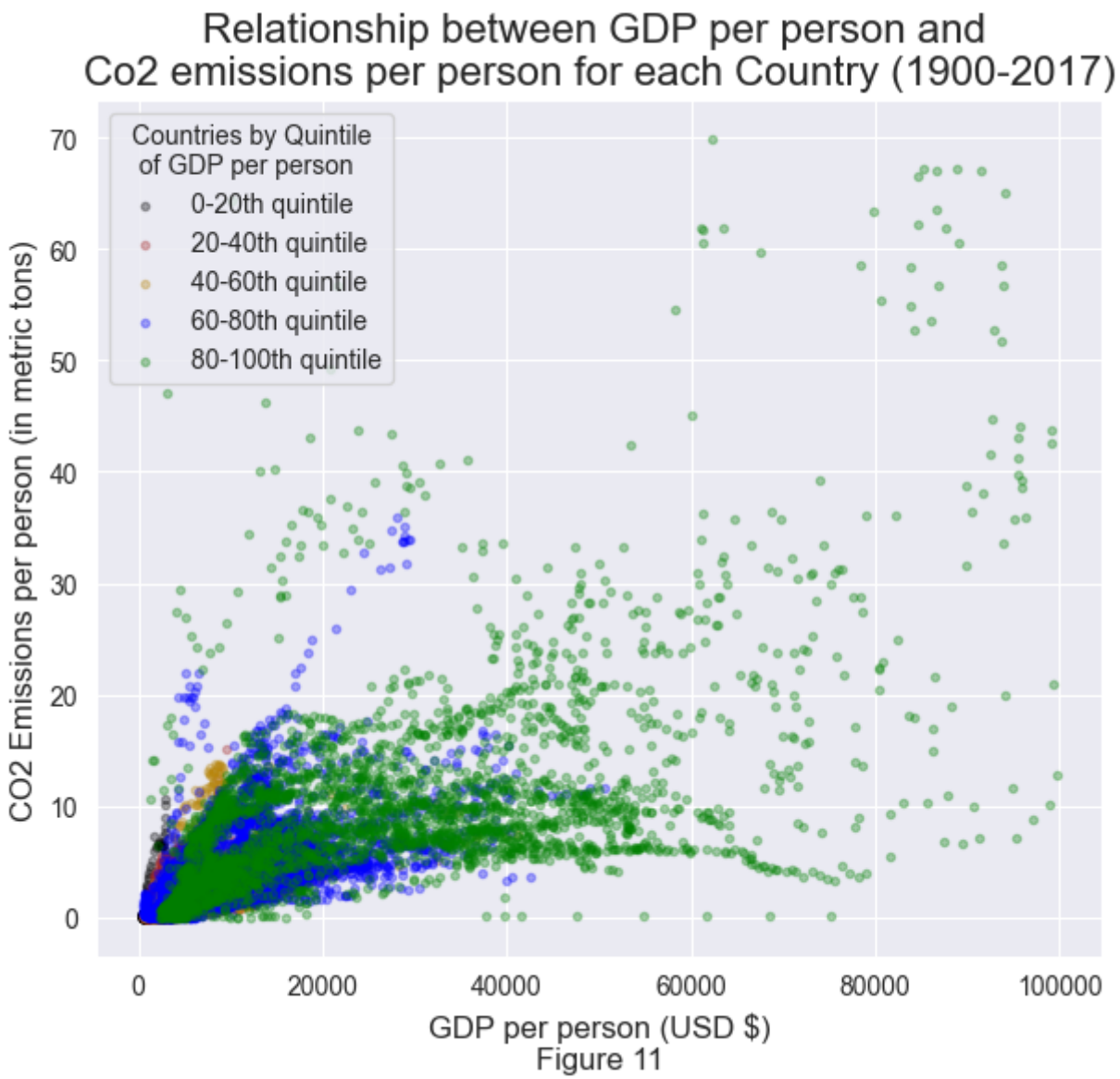


Figure 11

In [ ]:
```
# reset stacked df to itself
stacked_df = stacked_df_original
```

In [ ]:
```
# look at remaining plots
time_series_quintile(0, [0, 5])
quintile_wellbeing(0, [5])
```

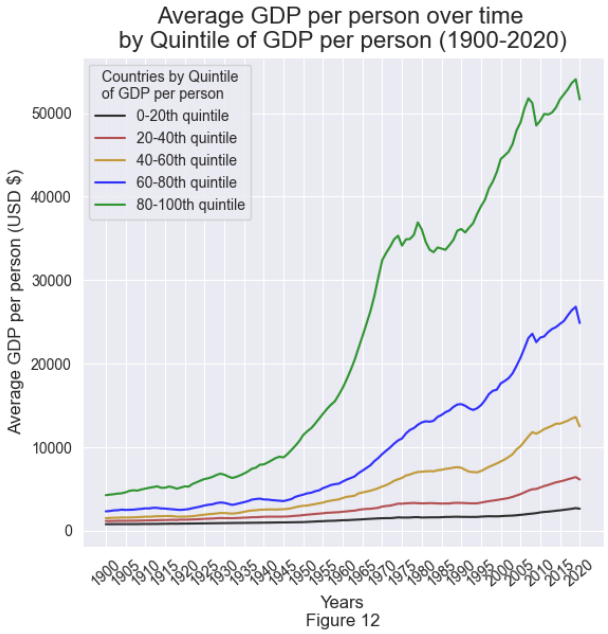


Figure 12

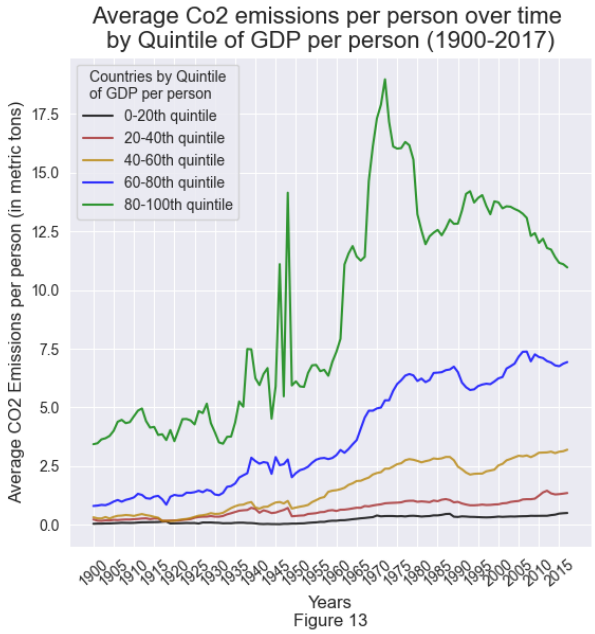


Figure 13

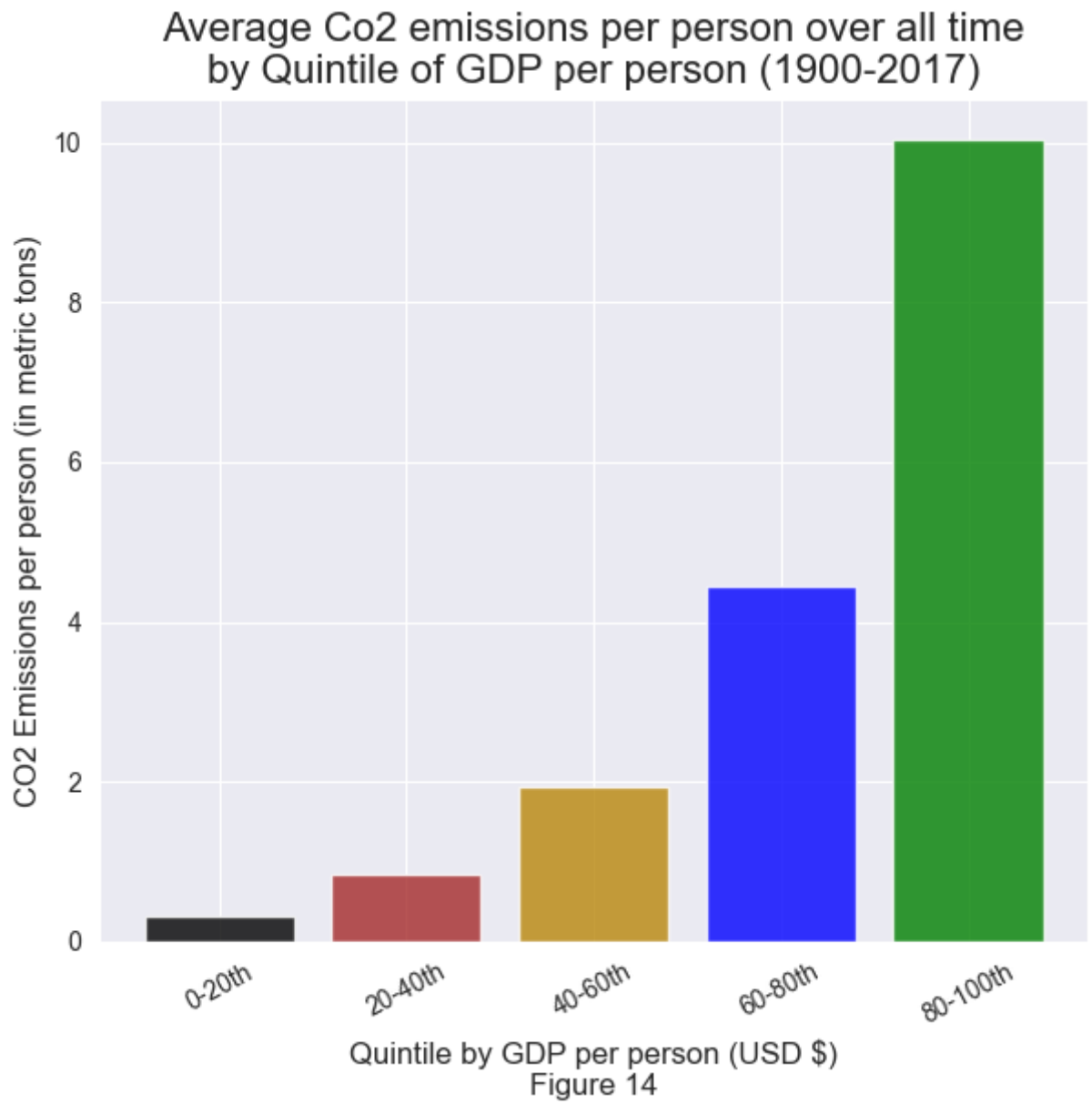


Figure 14

Comments

This scatter plot and bar chart show a potentially very strong positive correlation between CO2 and GDP per capita. The time series line is very interesting. There seems to have been a major inflection in trends of CO2 emissions around 1970 and the highest quintile has had steadily decreasing emissions since the 1990's. I decided to investigate this further by placing the growth of gdp next to it in a time series also for comparison in Figure 12 and 13. This shows that GDP has potentially decoupled from CO2 in recent years in the richest quintiles of countries.

I decided to also quickly plot the worlds average GDP per capita over a time series also just to see if we really did reach peak CO2 emissions per capita in the 1990's.

In [ ]:
```
# co2_df.mean().plot(color='black', title='Global CO2 emissions per capita');

# print(co2_df.mean().idxmax(), co2_df.mean().max())

# plt.ylabel('CO2 per capita (in Metric Tonnes)')
# plt.xlabel(f'Years\nFigure {figure_number}')

# figure_number+=1
```

In [ ]:
```
# gdp_df.iloc[-1,:-6].plot(color='blue', title='Global GDP per capita')

# plt.ylabel('GDP per capita (USD $)')
# plt.xlabel(f'Years\nFigure {figure_number}')

# figure_number+=1
```

In [ ]:
```
# gdp_df.loc['World', '1972':].to_frame().T
```

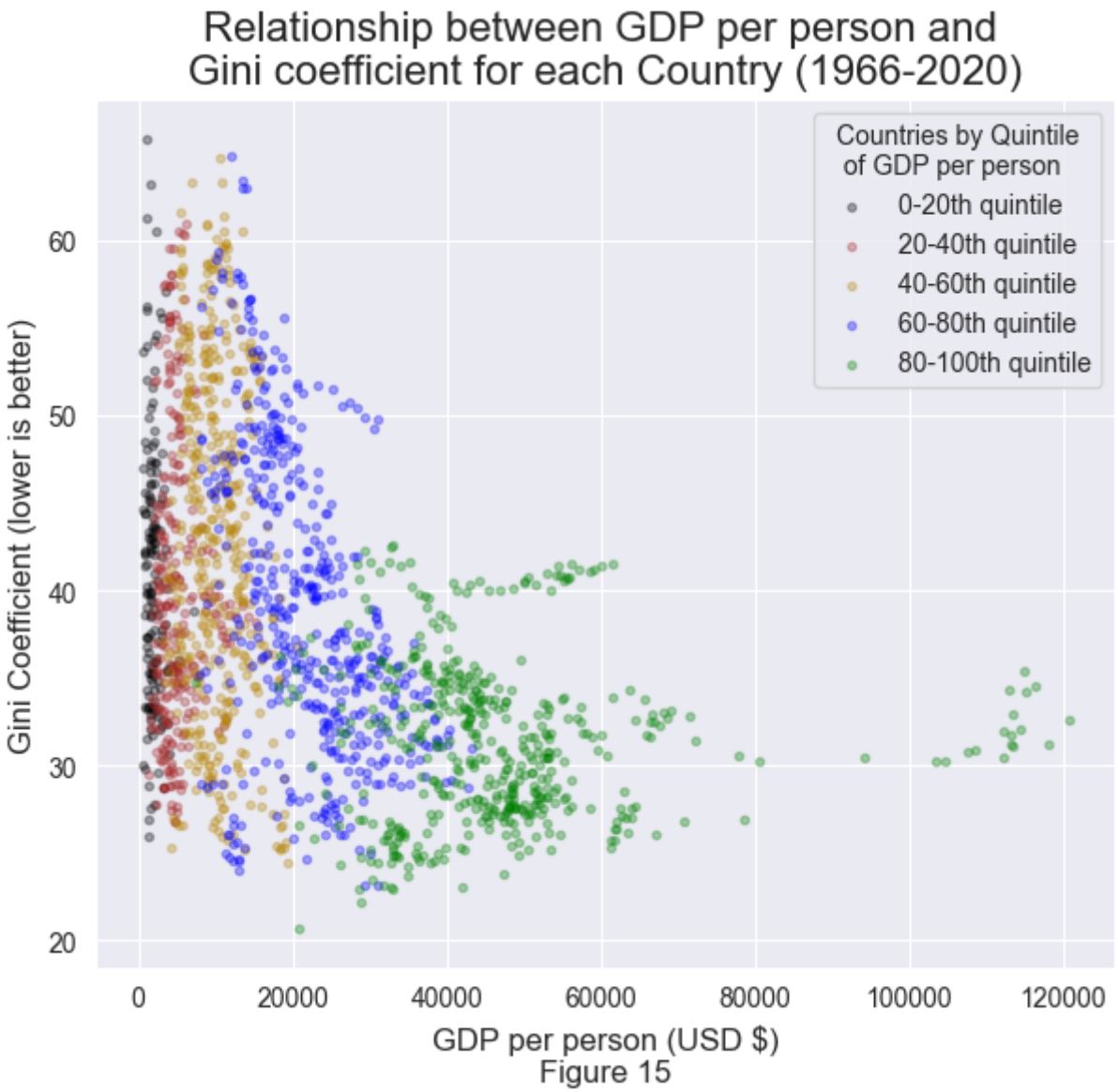
Yes, there was definitely a peak in 1972 at 5.42 metric tonnes of CO2 per person, and yet it appears that GDP per person has continued to increase.

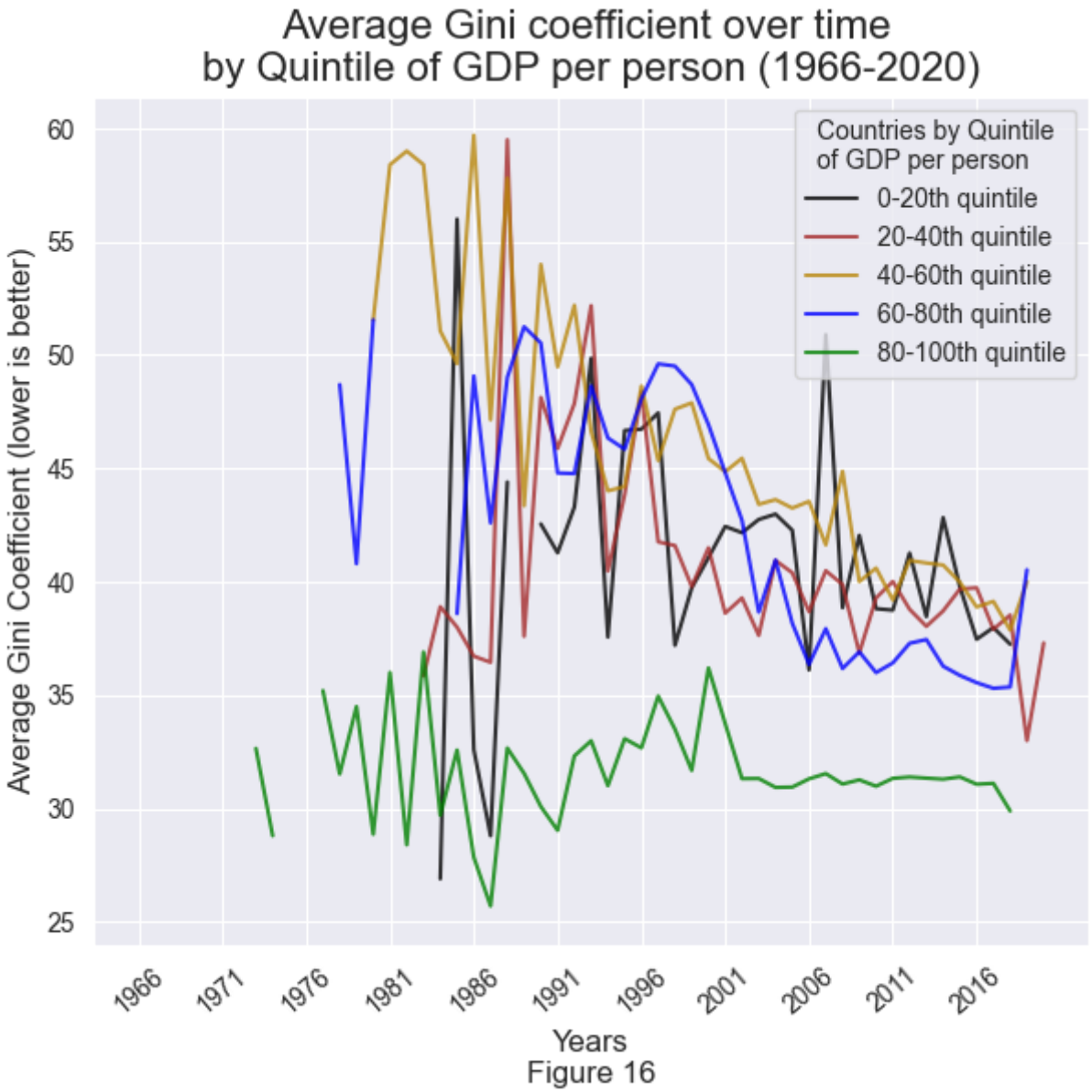
3d. Question 3: GDP and Gini

Scatter plot, time series line plot, bar chart

This subsection section explores the relationship between GDP and inequality with a scatter plot, a line chart of average inequality for each quintile of GDP over time, and a bar chart showing the average inequality of rich vs poor nations over all time.

In [ ]:
```
scatter_wellbeing(0,[1])
time_series_quintile(0, [1])
quintile_wellbeing(0, [1])
```


Relationship between GDP per person and
Gini coefficient for each Country (1966-2020)
Countries by Quintile
of GDP per person
0-20th quintile
20-40th quintile
40-60th quintile
60-80th quintile
80-100th quintile
Gini Coefficient (lower is better)
60
50
40
30
20
0
20000
40000
60000
80000
100000
120000
GDP per person (USD $)
Figure 15


Average Gini coefficient over time
by Quintile of GDP per person (1966-2020)
Countries by Quintile of GDP per person
0-20th quintile
20-40th quintile
40-60th quintile
60-80th quintile
80-100th quintile
Average Gini Coefficient (lower is better)
60
55
50
45
40
35
30
25
1966
1971
1976
1981
1986
1991
1996
2001
2006
2011
2016
Years
Figure 16

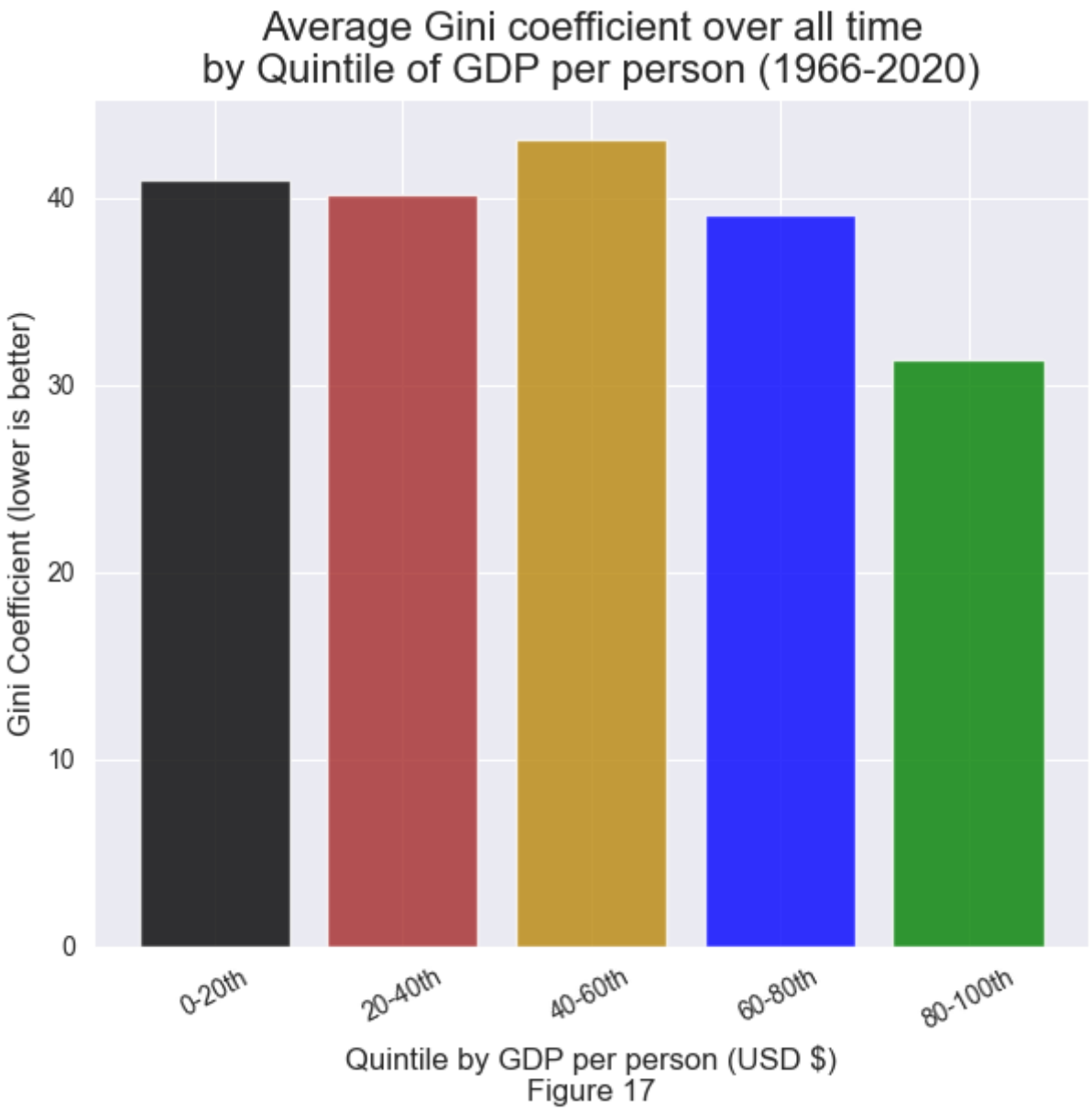


Figure 17

Comments

I noticed in the time series line plot that the Gini coefficient data was pretty noisy. So I ran a function to find out how many data points there were per year for all of the dataframes.

In [ ]:
```
def count_years(df):
    # this uses the years only function from earlier and counts the length of t
    df_years = len(years_only(df))

    # his then takes counts all the values in each column, then sums those numb
    df_total_values = df.iloc[:, :df_years].count().sum()

    # then finally returns the total number of values in that df divided by the
    return df_total_values / df_years

list_of_yearly_data = [count_years(i) for i in df_ls]

dict(zip(pretty_title, list_of_yearly_data))
```

Out[ ]:
```
{'GDP per person': 200.0,
 'Gini coefficient': 33.32727272727273,
 'Happiness': 121.58823529411765,
 'Murder rate': 47.25373134328358,
 'Freedom Index': 174.16326530612244,
 'Co2 emissions per person': 134.45762711864407,
 'Diversity': 216.0}
```

This indeed showed that the Gini data was more sparse than any of its counterparts at 33.3 data points per year, with only the murder rate dataframe coming close at 47.3 per year. This explained some of the Gini coefficients noisy appearance in the time series data, and in general makes me more hesitant to draw strong conclusions from this large amount of missing data. I also decided to find out what the exact numbers the bar charts were representing.

3e. Question 4: Gini and Welfare Metrics

Scatter plot, time series line plot, bar chart

To explore this question I looked at how the economic inequality of a country interacts with its happiness, murder rate, and freedom index through a scatter plot, a time series following the average outcomes of each Gini quintile through time, and a bar chart demonstrating the average welfare of the most unequal to the least unequal over all time.

Alert!

This final section will be using the **Gini Coefficient on the x-axis** and sorting all the nations into **Gini Quintiles** based on their average gini coefficient. It is important to note that the higher Gini Coefficient scores, the more unequal a society is. This means that the highest quintiles (80th-100th quintile range) are the bundle of the **MOST unequal nations**, while the lowest quintiles (0th-20th quintile range) are the **LEAST unequal nations**. Also the Gini data is the most sparse of all the dataframes I used, so sorting countries into Gini Quintiles from their averages over all time is likely to miss some nuance in the data.

In [ ]:
```
scatter_wellbeing(1, high_or_low=False)
time_series_quintile(1, high_or_low=False)
quintile_wellbeing(1, high_or_low=False)
```

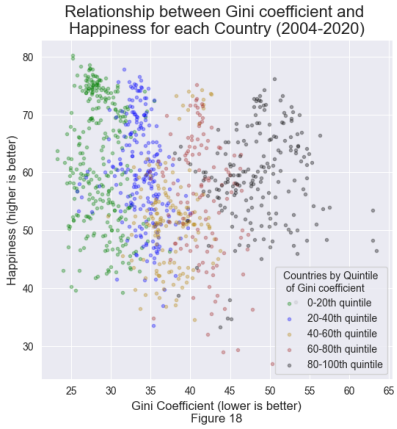


Figure 18

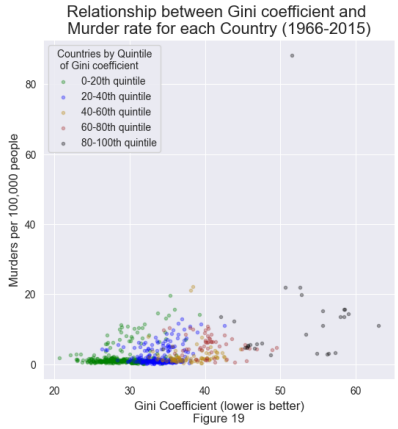


Figure 19

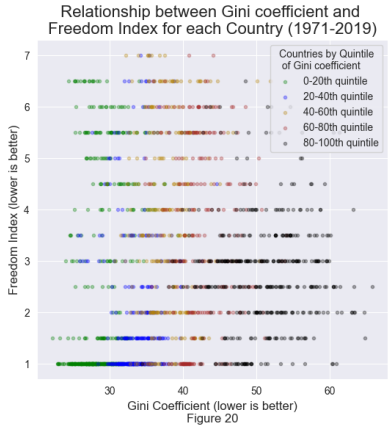


Figure 20

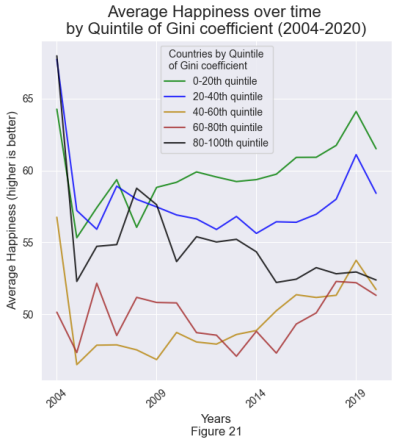


Figure 21

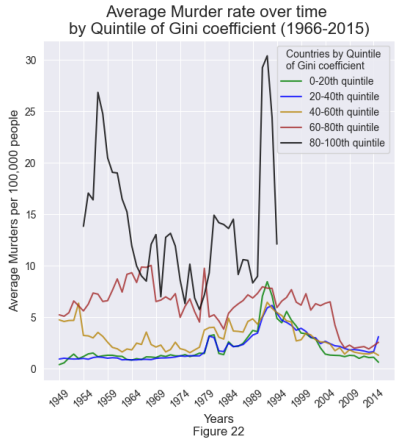


Figure 22

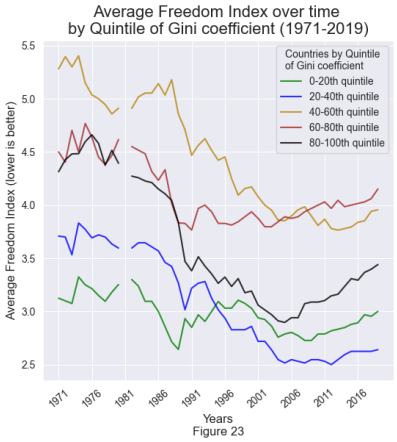


Figure 23

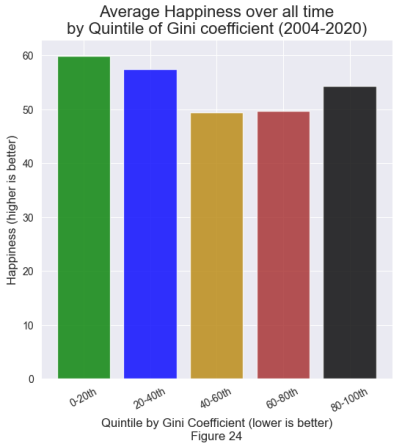


Figure 24

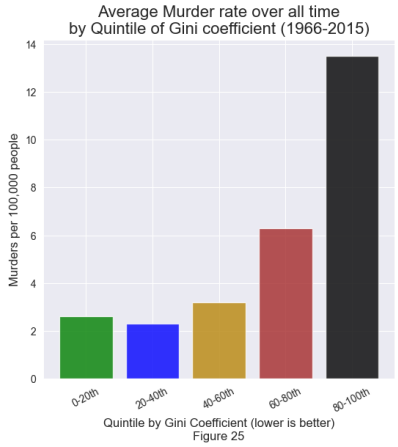


Figure 25

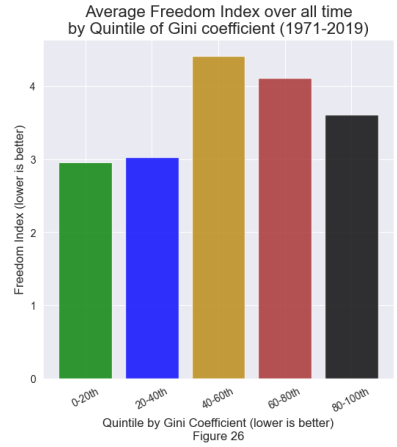


Figure 26

Comments

In the bar charts it looks like inequality as measured by the Gini Coefficient might very strongly correlate with deaths to murder -- however, when looking at the time series data it looks like the last measurement of the most unequal quintile of nations murder data comes from 1994! Also Murder Rate data has even more missing values amongst the most unequal countries of the world. I ran some code just to check how many data points per year there were in each quintile one average.

In [ ]:
```
# find amount of murder rate data per quintile
gini_quint_murders = murder_df.groupby('gini_quintile')\
    .count().iloc[:, :murder_years].sum(axis=1)

print("Total data per", gini_quint_murders)
```

```
# find average amount of murder rate data per year and per quintile
print("Average data per year in each", gini_quint_murders/murder_years)
```

```
Total data per gini_quintile
0-20th      1022
20-40th      839
40-60th      378
60-80th      208
80-100th     283
dtype: int64
Average data per year in each gini_quintile
0-20th      15.253731
20-40th     12.522388
40-60th      5.641791
60-80th      3.104478
80-100th     4.223881
dtype: float64
```

Wow, apparently the top three most unequal quintiles all have a lot of missing values in the murder dataframe also with less than 6 data points per year on average. This lack of data makes any conclusion very tentative.

3f. Correlation between all the variables

Heatmap of Correlations

After having explored all the relationships between the variables individually, I also wanted to see how all the variables correlated with each other. To do this I found the correlation of each column within the stacked_df dataframe with each other, then plotted that on a heatmap.

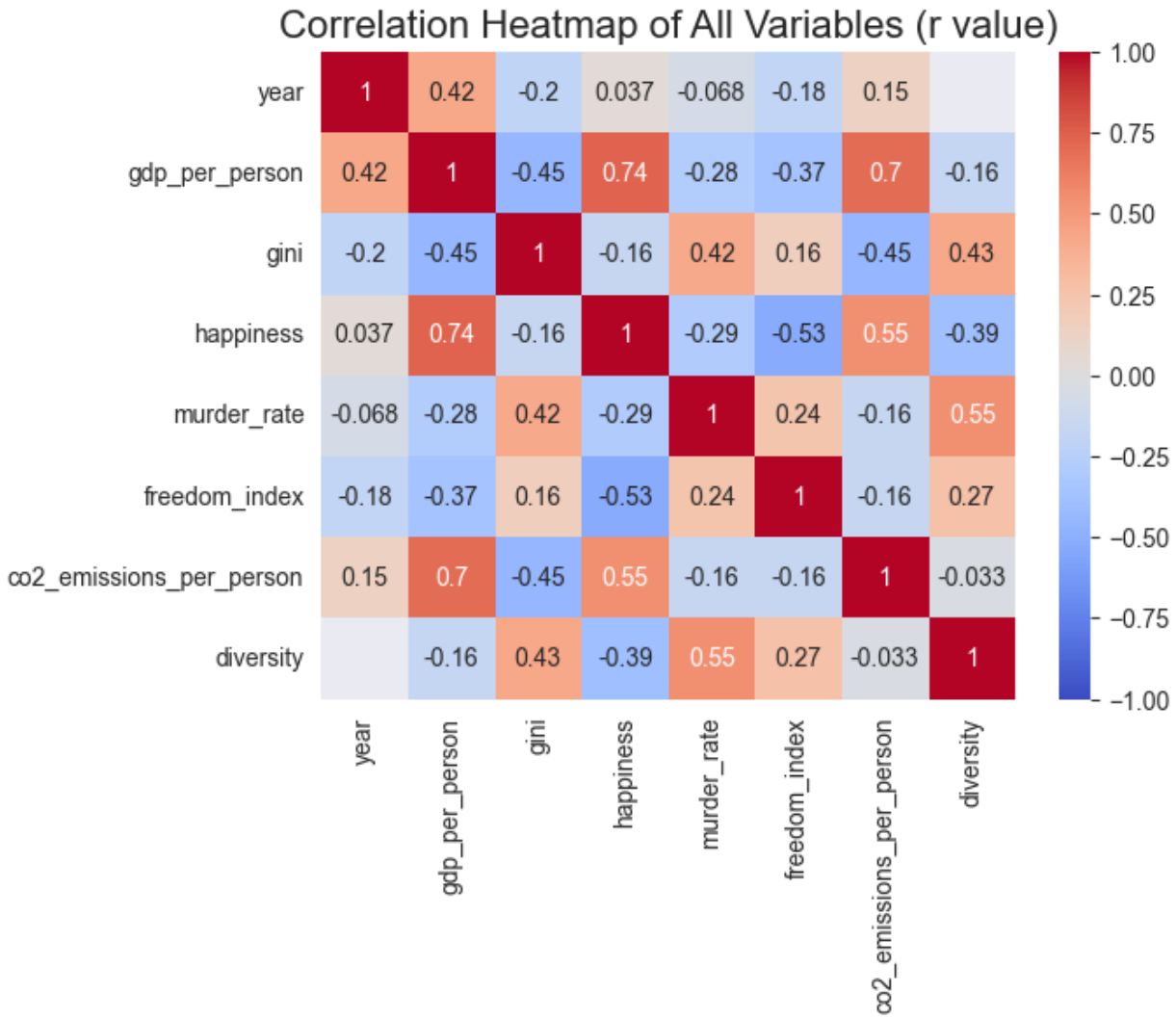
This first heatmap wasn't exactly what I wanted though. I wanted to organized the heatmap by whichever variable predicted the most variance in the other variables. So I added up the r squared values and displayed that result in a bar graph in Figure 28. Then I took a min-max normalization of these values, and made a second heatmap organized by which variables were the most impactful at the top and the least impactful at the bottom in Figure 29.

In [ ]:
```
stacked_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24424 entries, 0 to 24423
Data columns (total 16 columns):
 #   Column                              Non-Null Count  Dtype   
---  ------                              --------------  -----   
 0   country                             24424 non-null  object  
 1   year                                24424 non-null  int64   
 2   gdp_per_person                      24200 non-null  float64 
 3   gini                                1833 non-null   float64 
 4   happiness                           2067 non-null   float64 
 5   murder_rate                         3166 non-null   float64 
 6   freedom_index                       8534 non-null   float64 
 7   co2_emissions_per_person            15866 non-null  float64 
 8   diversity                           216 non-null    float64 
 9   gdp_per_person_quintile             24200 non-null  category
 10  gini_quintile                       20219 non-null  category
 11  happiness_quintile                  19723 non-null  category
 12  murder_rate_quintile                12735 non-null  category
 13  freedom_index_quintile              23389 non-null  category
 14  co2_emissions_per_person_quintile   23389 non-null  category
 15  diversity_quintile                  22361 non-null  category
dtypes: category(7), float64(7), int64(1), object(1)
memory usage: 1.8+ MB
```

In [ ]:
```
# calculating correlations
correlations = stacked_df.select_dtypes(include=['int64', 'float64']).corr()
sns.heatmap(correlations, annot=True, cmap='coolwarm', vmin=-1)
plt.title('Correlation Heatmap of All Variables (r value)', size=16)
```

Out[ ]:
```
Text(0.5, 1.0, 'Correlation Heatmap of All Variables (r value)')
```



In [ ]:
```
# create a series with all the absolute values of the correlations added togetl
determination_power = pd.Series((abs(correlations)).sum(axis=1), name='determir

# sort it by greatest correlation
sorted_determination_power = determination_power.sort_values(ascending=False)

# store order of greatest to least correlations
columns_in_order = sorted_determination_power.index
names_in_order = [i.replace('_',' ').title() for i in columns_in_order]

columns_in_order, names_in_order
```

Out[ ]:
```
(Index(['gdp_per_person', 'happiness', 'gini', 'co2_emissions_per_person',
        'murder_rate', 'freedom_index', 'diversity', 'year'],
       dtype='object'),
 ['Gdp Per Person',
  'Happiness',
  'Gini',
  'Co2 Emissions Per Person',
  'Murder Rate',
  'Freedom Index',
  'Diversity',
  'Year'])
```

In [ ]:
```
# display bar chart of summed absolute correlation values
plt.figure(figsize=(12,6))
```

```
sns.barplot(y = sorted_determination_power.values, x = sorted_determination_pow
plt.title('Summed Absolute Correlation Power of All Variables (Σ |r|)', size=16

plt.ylabel('Summed Determination Power')
plt.xticks(labels=names_in_order, ticks=np.array(range(0,len(pretty_title)+1)))
plt.xlabel(f"Figure {figure_number}")
figure_number+=1
```

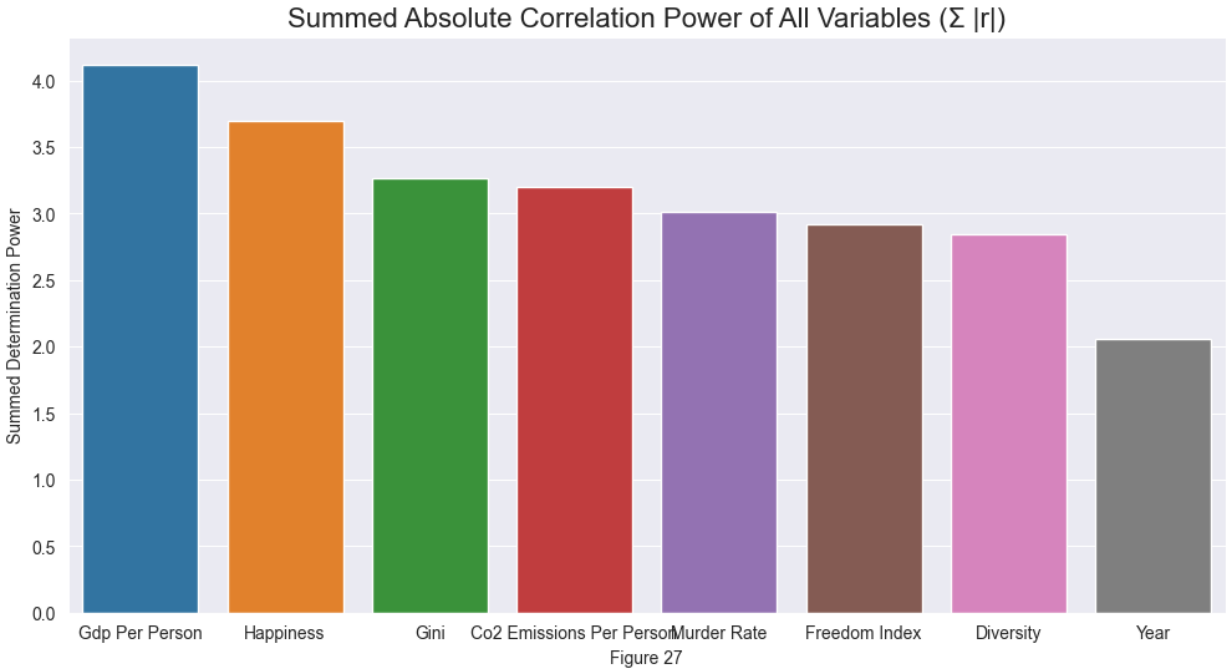


Figure 27

In [ ]:
```
# min-max the data so that it will fit onto the heatmap and give some informati
min_max_exp_power = (sorted_determination_power - sorted_determination_power.mi
    / (sorted_determination_power.max() - sorted_determination_power.min())

min_max_exp_power
```

Out[ ]:
```
gdp_per_person              1.000000
happiness                   0.795436
gini                        0.585633
co2_emissions_per_person    0.552450
murder_rate                 0.463052
freedom_index               0.415834
diversity                   0.380323
year                        0.000000
Name: determination_power, dtype: float64
```

In [ ]:
```
# concatenate the min_max values onto the bottom and side of the correlation da
# then organize it in both directions from strongest correlation to weakest

correlation_organized = pd.concat([correlations, min_max_exp_power], axis=1)\
    .sort_values(by='determination_power', ascending=False)

correlation_organized = pd.concat([correlation_organized, min_max_exp_power.to_
    .sort_values(by='determination_power', ascending=False, axis=1)

correlation_organized
```

Out[ ]:

	gdp_per_person	happiness	gini	co2_emissions_per_person
gdp_per_person	1.000000	0.738204	-0.450098	0.695806
happiness	0.738204	1.000000	-0.160008	0.551720
gini	-0.450098	-0.160008	1.000000	-0.446383
co2_emissions_per_person	0.695806	0.551720	-0.446383	1.000000
murder_rate	-0.284056	-0.290151	0.416301	-0.159775
freedom_index	-0.365148	-0.527162	0.164658	-0.163616
diversity	-0.164774	-0.394567	0.426033	-0.033416
year	0.423078	0.037196	-0.202558	0.146842
determination_power	1.000000	0.795436	0.585633	0.552450

In [ ]:

```
# record the order of all these columns and turn them into suitable axis titles
correlation_columns = correlation_organized.columns
abbreviated_names = [i.replace('_', ' ').split(" ")[0].title() \
    if 'power' not in i else "Relative\nCorrelation Strength" \
    for i in correlation_columns]

# plot final heatmap
plt.figure(figsize=(9,6))
sns.heatmap(correlation_organized.drop(index='determination_power'), annot=True
plt.title('Correlation Heatmap of All Variables (r value)\nOrdered by Min-Max c
plt.yticks(labels=abbreviated_names[:-1], ticks = np.array(range(0, int(len(pre
plt.xticks(labels=abbreviated_names, ticks=np.array(range(0, len(pretty_title)

plt.xlabel(f"Figure {figure_number}")
figure_number += 1
```

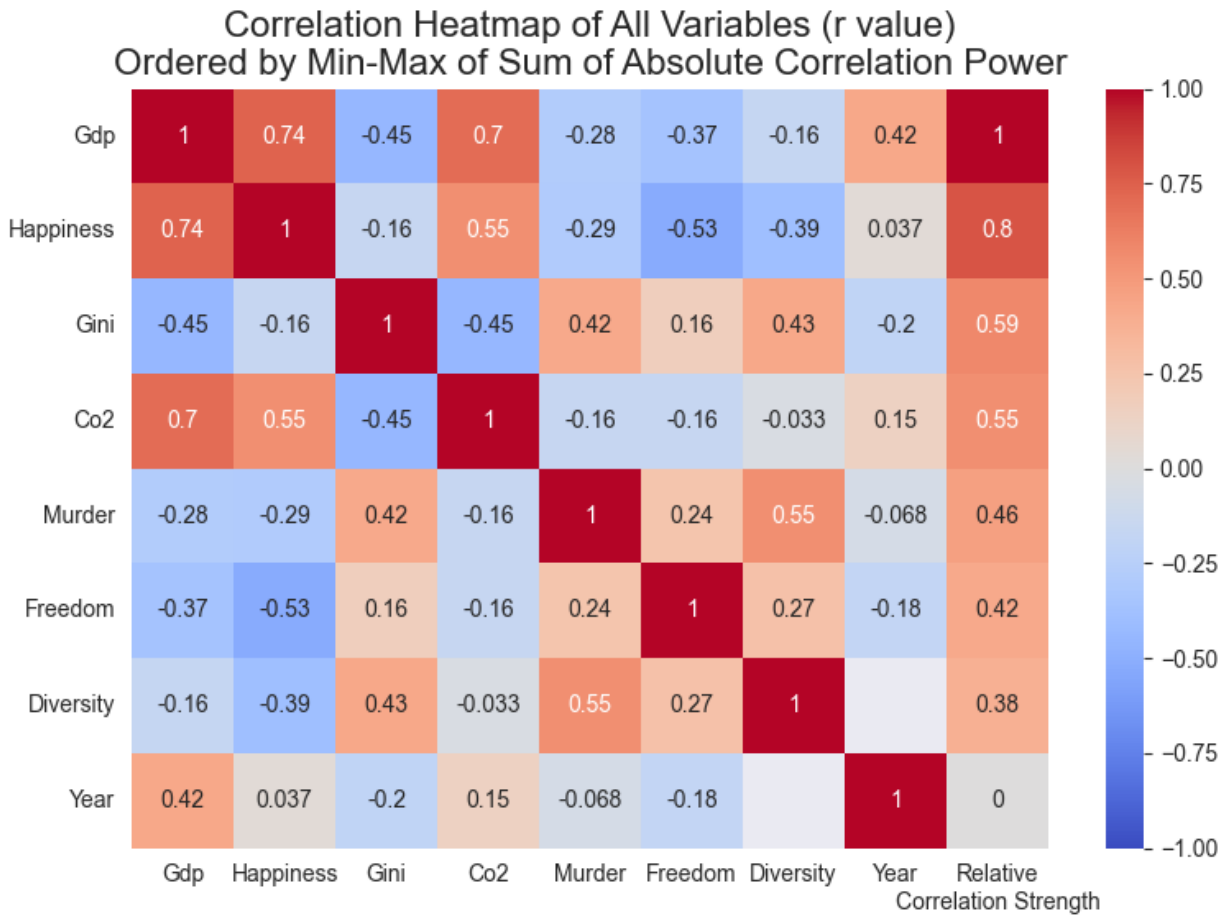


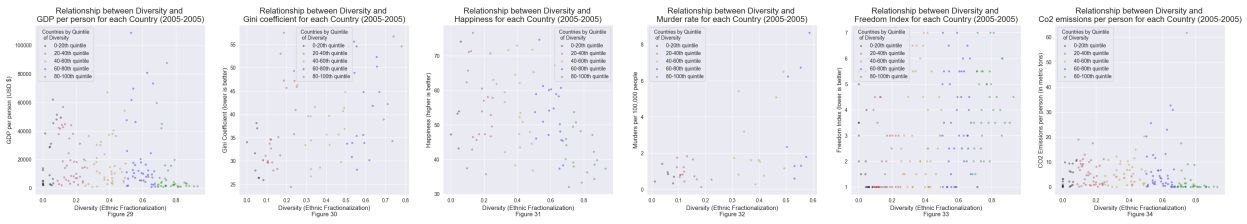
Figure 28

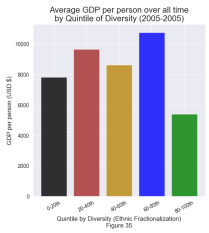
Comments

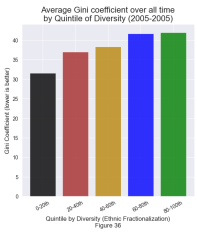
This one graph answers probably contains the most direct information about the correlation of all the variables with each other even though it doesn't give us much of an intuition for what is going on between all the variables, changes over time, and the values of the different quintiles of each variable.

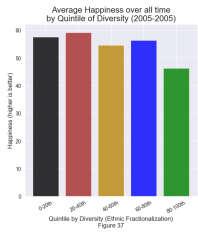
3z. New Dfs

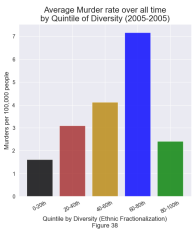
In [ ]:
```
scatter_wellbeing(6, [0, 1, 2, 3, 4, 5])
# time_series_quintile(0, [6])
quintile_wellbeing(6, [0, 1, 2, 3, 4, 5])
```

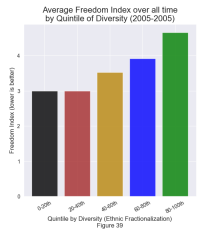


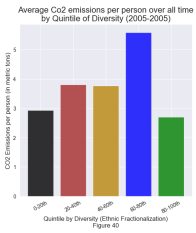












4. Conclusions

4a. Disclaimer

With this data I was only looking at correlations and using simple statistical methods like separating data into quintiles and calculating the means of aggregated data. No causal conclusions between the variables can be made from this analysis. Furthermore, I did not calculate any confidence intervals or do any t-tests in the data, so one must take these means and r values with an even greater grain of salt. The murder rate and the Gini coefficient data is particularly high variance and should be thought of has having large error bars around all of their estimations of central values.

4b. The Welfare of Nations

Question 1. Is an increase in GDP per capita is associated with increases in happiness, safety, and freedom?

It would seem that GDP per capita does have a strong positive correlation (r= 0.74) with improvements in happiness over the past 20 years. Also the average happiness improves for every increment in GDP quintile. The scatter plot does show though that this positive relationship probably doesn't hold for the super rich countries ($80,000$—120,000 GDP per capita PPP) which seem to have maxed out their happiness ratings around 60-75.

GDP per capita has a weak negative correlation (r = -0.28) with murder rate. While it would appear that the highest two quintiles have had consistently lower murder rates than the others over all time, the time series line plot shows that average murder rate has been steadily decreasing with time (r = -0.21) and is uniformly much lower in recent years. Of note, there is both a paucity of murder data for almost every country (average data entry per year is) and there is not a single reported murder rate for the lowest quintile of GDP countries, so these observations are very tenuous.

GDP per capita also appears to have a moderate negative correlation (r = -0.37) with the freedom index score, which is a good thing because the lower the freedom score the more civic freedom a country's citizens report experiencing. The time series chart also shows moderate increases in every countries freedom during the 1980's to 1990's, and that the 60-80th quintile has had a great increase in freedom

Overall, the average happiness and the freedom index of countries in the richer quintiles of GDP per capita seem to be consistently higher than their poorer counterparts. However, while the average murder rates appear to be very low in the richest quintile, its relationship with GDP per capita seems to be the weakest of these three.

4c. Rich Man, Poor World

Question 2. Is an increase in GDP per capita associated with an increase in CO2 emissions per capita?

There is a strong positive association between GDP per capita and CO2 emissions per capita (r = -0.70). This correlation is also evident in the trimmed down scatter plot graph of Figure 11. The bar charts in Figure 14 also point to an almost exponential increase in CO2 emissions per capita over all time as one increases in quintiles of GDP.

There do seem to some opposing trends though, specifically that global gas emissions per capita peaked in 1972 at a global average of 5.43 metric tonnes per capita, and yet out GDP per capita worldwide has continued to climb from $7,444 in 1972 to$16,207 in 2020 - it has basically doubled! A closer look at this period in the time series graph of Figure 12 and 13, reveals that during the past 30 years the richest quintile of countries has seen their CO2 emissions per capita consistently dropping while the other quintiles only slowly increase. This potentially means that GDP per capita growth could have recently disassociated with CO2 emissions for the richest countries in the world, but not for anyone else.

The data mostly points to the idea that GDP and CO2 emissions per capita are positively correlated. However since 1972, GDP and CO2 per capita growth has become increasingly less correlated with each other and for the past 30 years the top two richest quintiles of countries have seen their average CO2 emissions decrease even while their GDP increases. I did not however load in the gdp growth per capita dataset to fully explore this idea, so this is simply a potential trend.

4d. Progress and Poverty

Question 3. Is an increase in GDP per capita associated with an increase in economic inequality?

GDP per person and Gini coefficient did not positively correlate, in fact there was a moderate negative correlation between richness and inequality (r = -0.45). The bar chart in Figure 19 further shows that the richest quintile has less average inequality than any other quintile over the past 54 years.

On the other hand, it looks like the Gini inequality score has been lowering for all quintiles over time (r = -0.20), which is good but might mean that GDP per capita explains less of the general trend vs something else that has been changing over time. Also the Gini dataframe

has the least amount of data with an average of 33.3 data points per year, making this data even less reliable to make conclusions from.

In summary, as GDP per capita increases the inequality of a nation typically decreases. However, this relationship is speculative, because there is so little Gini data and the Gini has been decreasing as time goes on potentially independent of GDP per capita.

4e. The Gini's Curse

Question 4. Is an increase in economic inequality associated with a decrease in happiness, safety, and freedom?

More inequality seems to have very weak negative association on both happiness (r = -0.16) and the freedom index (r = 0.16) (lower is more freedom). The relationship that really jumps out at the viewer though especially in the bar chart of Figure 27 is the murder rate, which seems to moderately correlate with high inequality (r = 0.42).

Unfortunately, both the Gini and the murder data is very sparse so this correlation has some of the least amount of data to back it up with only 4.22 data points on average per year in the highest quintile of inequality.

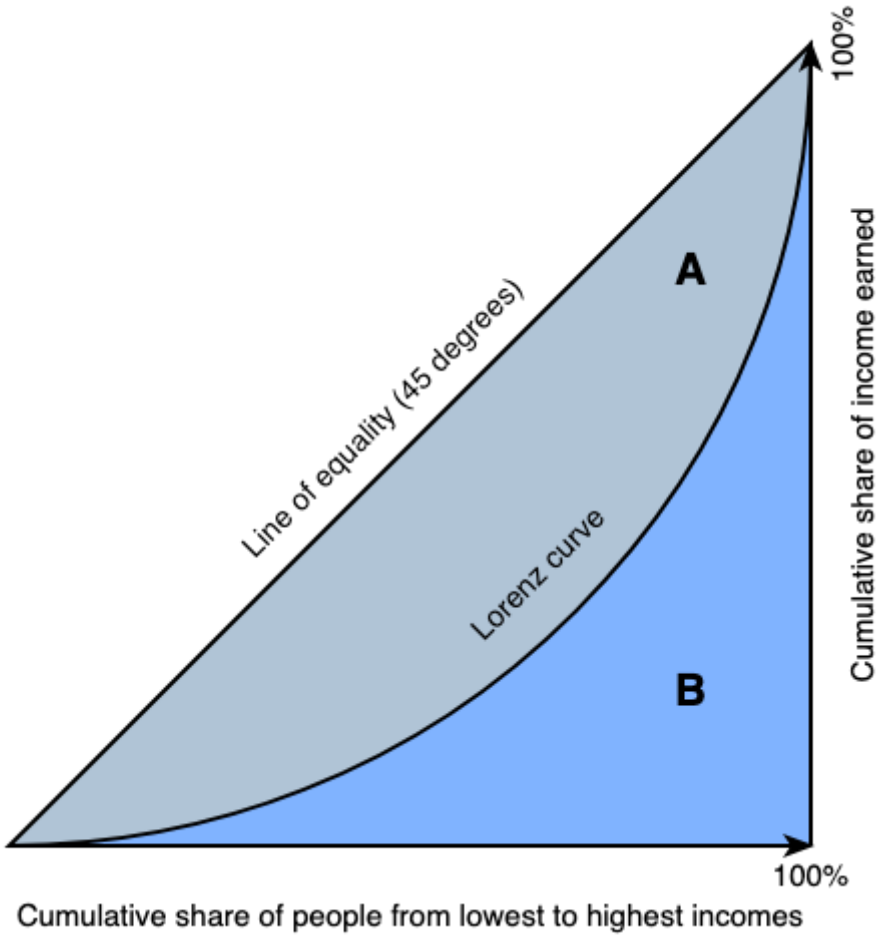
In conclusion, it looks like inequality very weakly correlates with worse happiness and freedom metrics in a country. The Gini Coefficient has a moderate correlation with increased murder rates, but missing data makes this relationship difficult to establish.

4f Footnotes:

1. **Math on how to compute a Gini Coefficient**

   The Gini coefficient is calculated by ordering each household's income from least to greatest, then cumulatively summing those incomes until it is equal to the total income of the nation. This data is then fitted with a nonlinear regression line called a Lorenz Curve and then plotted on a 2d graph with the percentiles of the population on the x axis and the percentage of total income on the y axis. The next step is to graph the same cumulative sum for a hypothetical country where each household's income is exactly equal. This will form a straight 'line of perfect equality' above the Lorenz Curve. The final step to find the Gini Coefficient is to find the difference in area between the line of perfect equality and the actual Lorenz Curve, and divide that difference by the total area under the line of perfect equality.

   GC = (Eq - LC) / Eq



4g. Research Notes

1. Wikipedia on the Gini Coefficient
   I had to do some background research on the Gini Coefficient.

1. Four Reasons why GDP is a Useful Number by Noah Smith
   This helped me better understand GDP and why we currently use it in economics

1. "How to slash carbon emissions while growing the economy, in one chart" by Vox
   This helped me better understand what I wanted to investigate in the GDP per capita vs CO2 per capita data.

1. Gapminder Website
   This had lots of great background information on all their databases.