DATA1002 Project - Stage 2

Group 12

**Members: 510615460, 480385312, 510465739, 500522378**

[**1.0 Overview**](#_kiwua7muqfa5) **1**

[1.1 Topic](#_yh69rt7vnyja) 1

[1.2 Dataset](#_xlhiw09ifsrr) 1

[**2.0 Individual Sections**](#_bm9geawn4bmc) **3**

[2.1 Section 1 - 510615460](#_p9kfjdc8snmi) 3

[2.1.1 Grouped Aggregate Summaries](#_fcgvjak5c353) 3

[2.1.1.1 Average Vaccinations (Per Country)](#_5zrjt03wvlin) 3

[2.1.1.2 Average Number of Tests (Per Population Bin)](#_c7jfr7r15hjj) 6

[2.1.2 Charts](#_7r55lqbbz53v) 7

[2.1.2.1 Average Number of Vaccinations (Per Country - Barplot)](#_bp7zbm189hj7) 7

[2.1.2.2 Average Number of Vaccinations and Average Number of New Tests in Each Country (Per Population Bin - Barplot)](#_z6d58n9dtzzp) 12

[2.2 Section 2 - 480385312](#_w9lwdjc4nky7) 15

[2.2.1 Grouped Aggregate Summaries](#_noqjezp1tvd6) 15

[2.2.1.1 Sum of Vaccinations over Population (Per Country)](#_wcqgfcimyn02) 15

[2.2.1.2 Sum of Vaccinations over Population (Per GDP Bin)](#_td4mgu1h0dqv) 19

[2.2.2 Charts](#_o7o148ikagsb) 21

[2.2.2.1 Sum of Vaccinations over Population (per Country - Overlay Bar Plot)](#_rpzer7wd10e5) 21

[2.2.2.2 Vaccinations per Person (per Country - Bar Plot)](#_smg03t270hvp) 25

[2.2.2.3 Vaccinations per Person (per GDP Bin - Line Plot)](#_pm17vqt1zjr0) 29

[2.3 Section 3 - 510465739](#_450xntvgnci6) 32

[2.3.1 Grouped Aggregate Summaries](#_fqnckm1yl34f) 32

[2.3.1.1 Average Number of Deaths Binned by Tests](#_ja27lyeowde0) 32

[2.3.1.2 Average Number of Tests Grouped by Country](#_q7jv2vlen5ba) 34

[2.3.2 Charts](#_62dx70bq3fdn) 37

[2.3.2.1 Average Number of Tests per million People (per Country - Barplot)](#_jiealgdlmt2o) 37

[2.3.2.2 Average Number of Cases and Deaths (per Vaccination Bin - Line Plot)](#_uddpl445m3jv) 42

[2.4 Section 4 - 500522378](#_v3z7bczcm0ug) 46

[2.4.1 Grouped Aggregate Summaries](#_u0adi17rj315) 46

[2.4.1.1 Average Number of Cases per Country](#_ofl3l8mjwkt6) 46

[2.4.1.2 Average Number of Deaths by Population Bin](#_m4nc6gwyb2eb) 48

[2.4.2 Charts](#_gl7zqh80w8et) 50

[2.4.2.1 Average Deaths by Population Bin](#_xr58rzd6tzzk) 50

[2.4.2.2 Average Cases and Vaccinations by Population Bin](#_m7qzs0hwr3zs) 52

[2.4.2.3 Average Cases per Country (Bar Plot)](#_eyq4z1ww98tx) 55

[**3.0 Combined Chart**](#_fb8envdfhh99) **61**

[**4.0 Conclusion**](#_nc0vzo93nnau) **65**

Part A

# 1.0 Overview

## 1.1 Topic

This report is investigating the factors that have contributed to the impact of the COVID-19 pandemic around the world. In particular, we are investigating aspects of each country, including their GDP, population and testing/vaccination response on the number of cases and deaths. This information is relevant to public health officials, as this can help them to compare their response to countries with similar factors to their own, as well as see how their response holds up at the global scale. This data may also help to determine estimates for how cases and deaths will develop in a given country, by comparing it to other countries. This, in turn, may help a country to determine the best way to alter their response to the pandemic, and potentially advise on how to ‘live with the virus’.

## 1.2 Dataset

The data used in this report is based on the one our group produced in Stage 1 of this project. In order to allow for more analysis, we acquired additional testing data from the [Worldometer source](https://www.worldometers.info/coronavirus/covid-19-testing/), which allowed us to extend our integrated dataset across the date range of July 2020 - June 2021.

**Data Schema**

| **Field Name** | **Type** | **Description** |
| --- | --- | --- |
| Date | Date | Date on which the data was collected. Represented in standard ISO format as YYYY-MM-DD. |
| Country | String | The country from which the data was recorded. |
| GDP | Integer | The gross domestic product for the current financial year, measured in USD. |
| Population | Integer | The population of the country. |
| New Cases | Integer | The new cases of COVID-19 recorded for this date. |
| New Deaths | Integer | The new deaths of COVID-19 recorded for this date. |
| Tests | Integer | The number of COVID-19 tests administered for this date. |
| Vaccinations | Integer | The number of COVID-19 vaccinations administered for this date. |
| Vaccines Available | String Array | The names of the brands of COVID-19 vaccines available in the country. |

# 2.0 Individual Sections

## 2.1 Section 1 - 510615460

### 2.1.1 Grouped Aggregate Summaries

#### 2.1.1.1 Average Vaccinations (Per Country)

This aggregate summary calculates the average number of vaccinations administered for each country.

**Code**

import csv

average\_vaccination\_per\_country = {}

country\_total\_vaccination = {}

is\_first\_line = True

csv\_reader = csv.reader(open("integrated\_data.csv"))

for row in csv\_reader:

# Ignore header row.

if is\_first\_line:

is\_first\_line = False

else:

country = str(row[1])

daily\_vaccination = float(row[7])

# Add country the first time it is seen.

if country not in country\_total\_vaccination:

country\_total\_vaccination[country] = daily\_vaccination

else:

country\_total\_vaccination[country] += daily\_vaccination

if country not in average\_vaccination\_per\_country:

average\_vaccination\_per\_country[country] = [daily\_vaccination]

else:

average\_vaccination\_per\_country[country].append(daily\_vaccination)

# Calculate average.

average\_sum = 0

print("\nAverage number of vaccination for each country:")

for country in average\_vaccination\_per\_country:

vaccination = average\_vaccination\_per\_country[country]

average = sum(vaccination) / len(vaccination)

average\_sum += average

print("\t{}: {}".format(country, average))

# Overall aggregate.

print('\n\n')

print('Overall:', average\_sum)

print('\n')

**Table of Results**

| **Country** | **Average Vaccinations** |  | **Country** | **Average Vaccinations** |
| --- | --- | --- | --- | --- |
| Argentina | 53553.93 |  | Malaysia | 19610.81 |
| Armenia | 202.86 |  | Maldives | 1393.67 |
| Australia | 19801.92 |  | Mexico | 118752.77 |
| Austria | 20580.88 |  | Morocco | 51423.58 |
| Bangladesh | 27746.09 |  | Mozambique | 1292.97 |
| Belarus | 3891.32 |  | Namibia | 377.85 |
| Belgium | 29046.06 |  | Nepal | 9282.47 |
| Bolivia | 6568343 |  | Netherlands | 41013.82 |
| Bosnia | 1242.33 |  | New Zealand | 2986.19 |
| Botswana | 681.80 |  | Nigeria | 8788.66 |
| Bulgaria | 4742.56 |  | North Macedonia | 1393.14 |
| Cabo Verde | 218.19 |  | Norway | 10916.40 |
| Canada | 96585.17 |  | Pakistan | 39194.80 |
| Chile | 61938.90 |  | Panama | 4087.68 |
| Colombia | 46733.21 |  | Paraguay | 2364.30 |
| Costa Rica | 6603.45 |  | Peru | 19959.47 |
| Cote d'Ivoire | 2042.97 |  | Philippines | 26289.93 |
| Croatia | 6880.45 |  | Poland | 77048.77 |
| Cyprus | 2157.83 |  | Portugal | 22264.65 |
| Denmark | 13784.39 |  | Qatar | 8238.68 |
| Dominican Republic | 20804.88 |  | Romania | 24381.51 |
| Ecuador | 10944.89 |  | Russia | 103023.76 |
| Estonia | 2551.62 |  | Rwanda | 1742.41 |
| Ethiopia | 4559.88 |  | Saudi Arabia | 46947.41 |
| Fiji | 837.63 |  | Senegal | 1833.59 |
| Finland | 11465.22 |  | Serbia | 14327.74 |
| France | 144097.31 |  | Singapore | 14407.98 |
| Germany | 199319.65 |  | Slovakia | 9606.07 |
| Ghana | 3656.81 |  | Slovenia | 4011.58 |
| Greece | 22020.59 |  | South Africa | 7194.30 |
| Guatemala | 2521.57 |  | South Korea | 51478.37 |
| Hungary | 28245.12 |  | Spain | 107951.41 |
| Iceland | 1045.03 |  | Sri Lanka | 9524.22 |
| India | 857582.72 |  | Switzerland | 19813.12 |
| Indonesia | 108076.55 |  | Thailand | 24334.89 |
| Iran | 14953.12 |  | Togo | 1093.12 |
| Iraq | 2530.87 |  | Trinidad and Tobago | 705.38 |
| Ireland | 10882.42 |  | Turkey | 130039.58 |
| Israel | 29609.34 |  | Uganda | 2343.00 |
| Italy | 135823.86 |  | United Kingdom | 21069.27 |
| Jamaica | 705.69 |  | United States | 885389.90 |
| Latvia | 3058.42 |  | Uruguay | 10388.03 |
| Lithuania | 6054.01 |  | Zambia | 415.52 |
| Luxembourg | 1407.69 |  | Zimbabwe | 3369.90 |
| Madagascar | 542.06 |  | OVERALL | 4179980.13 |

#### 2.1.1.2 Average Number of Tests (Per Population Bin)

This aggregate summary calculates the average number of tests per population bin. Each country is assigned into a bin based on its population range. In total, there are 17 separate bins and within those bins, the average number of tests are calculated.

**Code**

import csv

first\_line = True

# Dictionaries of populationBin -> value.

tests\_by\_pop\_bin = {}

average\_tests\_by\_pop\_bin = {}

csv\_reader = csv.reader(open("integrated\_data.csv"))

for row in csv\_reader:

# Ignore header row.

if first\_line:

first\_line = False

else:

pop = int(row[3])

tests = int(row[6])

pop\_bin = pop // 10000000

if pop\_bin not in tests\_by\_pop\_bin:

tests\_by\_pop\_bin[pop\_bin] = [tests]

else:

tests\_by\_pop\_bin[pop\_bin].append(tests)

# Calculate average.

for pop\_bin in tests\_by\_pop\_bin:

average\_tests = round((sum(tests\_by\_pop\_bin[pop\_bin]) / len(tests\_by\_pop\_bin[pop\_bin])))

average\_tests\_by\_pop\_bin[pop\_bin] = average\_tests

num = 0

for pop\_bin in average\_tests\_by\_pop\_bin:

print(num, ':', average\_tests\_by\_pop\_bin[pop\_bin])

num+=1

# Overall aggregate.

total\_test = 0

total\_pop\_bins = 0

for key in sorted(average\_tests\_by\_pop\_bin):

total\_test += average\_tests\_by\_pop\_bin[key]

total\_pop\_bins += 1

average\_tests\_by\_pop\_bin["OVERALL"] = round(total\_test / total\_pop\_bins)

print(average\_tests\_by\_pop\_bin)

**Table of Results**

| **Bin** | **Average Tests** |  | **Bin** | **Average Tests** |
| --- | --- | --- | --- | --- |
| 0 | 17663 |  | 10 | 1073862 |
| 1 | 15736 |  | 11 | 26973 |
| 2 | 9790 |  | 12 | 17931 |
| 3 | 15938 |  | 13 | 1694 |
| 4 | 12551 |  | 14 | 32870 |
| 5 | 33392 |  | 15 | 37208 |
| 6 | 35173 |  | 16 | 317768 |
| 7 | 7164 |  | 17 | 1191183 |
| 8 | 241627 |  | OVERALL | 174730 |
| 9 | 56610 |  |  |  |

### 2.1.2 Charts

#### 2.1.2.1 Average Number of Vaccinations (Per Country - Barplot)

**Code**

import csv

import matplotlib.pyplot as plt

average\_vaccination\_per\_country = {}

country\_total\_vaccination = {}

plot\_dict = {}

is\_first\_line = True

csv\_reader = csv.reader(open("integrated\_data.csv"))

for row in csv\_reader:

# Ignore header row,

if is\_first\_line:

is\_first\_line = False

else:

country = str(row[1])

daily\_vaccination = float(row[7])

if country not in country\_total\_vaccination:

country\_total\_vaccination[country] = daily\_vaccination

else:

country\_total\_vaccination[country] += daily\_vaccination

if country not in average\_vaccination\_per\_country:

average\_vaccination\_per\_country[country] = [daily\_vaccination]

else:

average\_vaccination\_per\_country[country].append(daily\_vaccination)

print("\nAverage number of vaccination for each country:")

for country in average\_vaccination\_per\_country:

vaccination = average\_vaccination\_per\_country[country]

average = sum(vaccination) / len(vaccination)

print("\t{}: {}".format(country, average))

plot\_dict[country] = average

keys = list(plot\_dict.keys())

values = list(plot\_dict.values())

# Additional formatting for y axis.

maxY = max(values)

yTickIncrement = 100000

yTicks = []

y = 0

while y <= maxY:

yTicks.append(y)

y += yTickIncrement

# Split the countries into separate graphs, so that the x axis isn't too crowded.

maxCountriesPerGraph = 15

offset = 0

while offset < len(keys):

plt.bar(keys[offset:offset+maxCountriesPerGraph], values[offset:offset+maxCountriesPerGraph])

plt.xticks(rotation=90)

plt.yticks(yTicks)

plt.xlabel("Country")

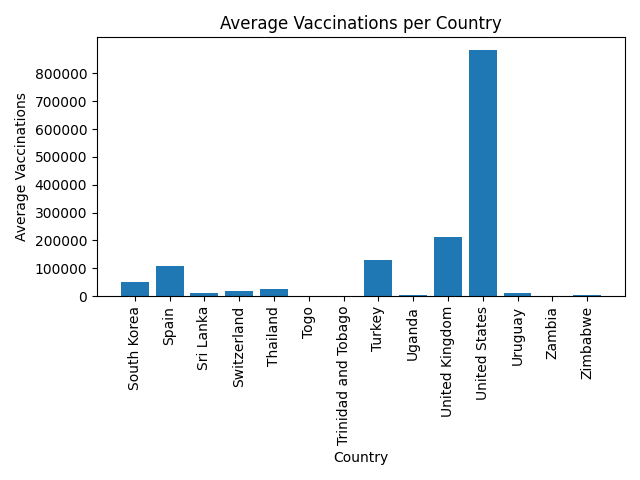
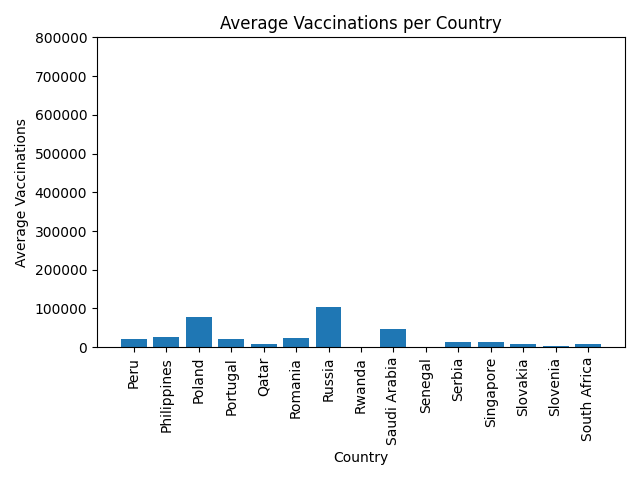
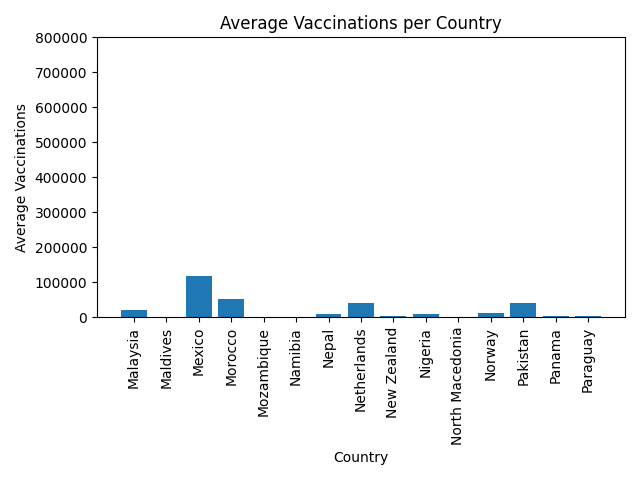
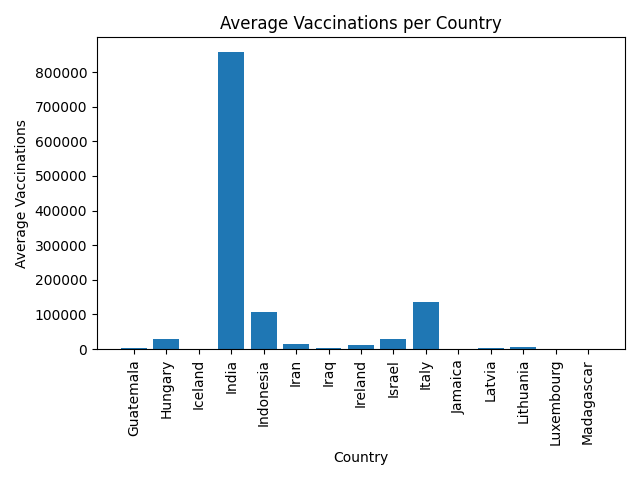
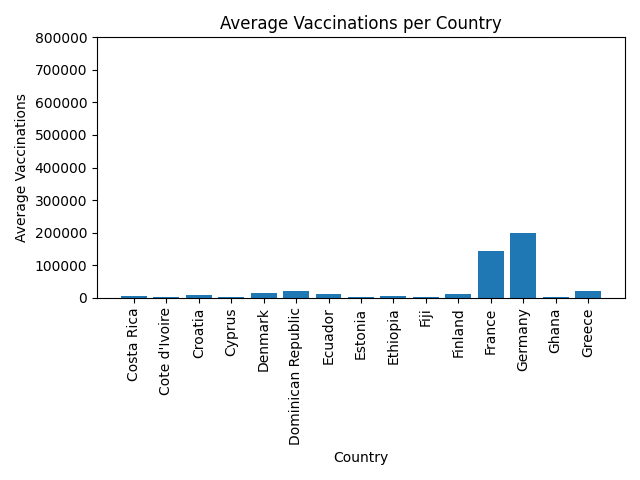
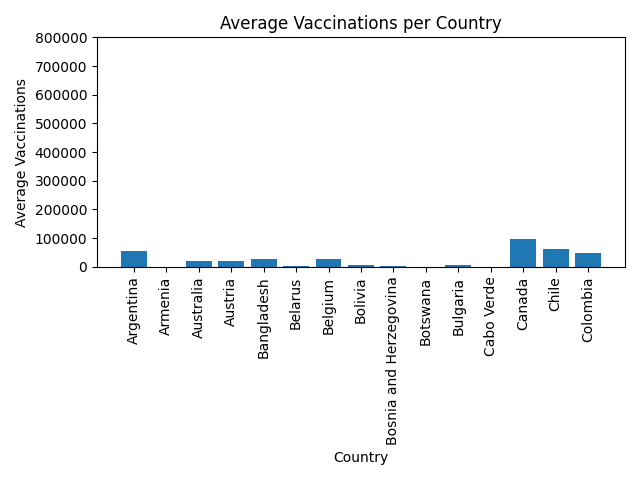
plt.ylabel("Average Vaccinations")

plt.title("Average Vaccinations per Country")

plt.show()

offset += maxCountriesPerGraph

**Chart**



**Evaluation**

The bar plots above show the average vaccinations administered for each country. The average number of vaccinations are encoded in the height of the bar plot as the y-axis and each country is horizontally encoded as the x-axis. A bar plot is most appropriate for this particular summary as it can effectively compare the average vaccinations administered between countries. While the chart can easily compare the averages through its height, some issues may arise if more data is added. If more data is added, there will be a change in the height of the bars which may make the chart less effective as the average for some countries (with smaller values) will be less visible, however, in the cases of countries that do have smaller averages (in the plot), it may make the chart more effective as their bars will be more visible. Thus, if there is more data for each country, it will be harder to read.

#### 2.1.2.2 Average Number of Vaccinations and Average Number of New Tests in Each Country (Per Population Bin - Barplot)

**Code**

import csv

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

first\_line = True

# Dictionaries of populationBin -> value.

tests\_by\_pop\_bin = {}

average\_tests\_by\_pop\_bin = {}

vac\_by\_pop\_bin = {}

average\_vac\_by\_pop\_bin = {}

csv\_reader = csv.reader(open("integrated\_data.csv"))

for row in csv\_reader:

# Ignore header row.

if first\_line:

first\_line = False

else:

pop = int(row[3])

tests = int(row[6])

vac = float(row[7])

# Calculate the population bin for this value.

pop\_bin = pop // 10000000

# Add the country the first time it is seen.

if pop\_bin not in tests\_by\_pop\_bin:

tests\_by\_pop\_bin[pop\_bin] = [tests]

else:

tests\_by\_pop\_bin[pop\_bin].append(tests)

if pop\_bin not in vac\_by\_pop\_bin:

vac\_by\_pop\_bin[pop\_bin] = [vac]

else:

vac\_by\_pop\_bin[pop\_bin].append(vac)

# Calculate averages.

avg\_test\_list = []

for pop\_bin in tests\_by\_pop\_bin:

average\_tests = round((sum(tests\_by\_pop\_bin[pop\_bin]) / len(tests\_by\_pop\_bin[pop\_bin])))

avg\_test\_list.append(average\_tests)

average\_tests\_by\_pop\_bin[pop\_bin] = average\_tests

avg\_vac\_list = []

for pop\_bin in vac\_by\_pop\_bin:

average\_vac = round((sum(vac\_by\_pop\_bin[pop\_bin]) / len(vac\_by\_pop\_bin[pop\_bin])))

avg\_vac\_list.append(average\_vac)

average\_vac\_by\_pop\_bin[pop\_bin] = average\_vac

data\_list\_1 = list(average\_tests\_by\_pop\_bin.items())

df1 = pd.DataFrame(data\_list\_1)

data\_list\_2 = list(average\_vac\_by\_pop\_bin.items())

df2 = pd.DataFrame(data\_list\_2)

combined\_data = [df1[1], df2[1]]

headers = ["Average Tests", "Average Vaccinations"]

df3 = pd.concat(combined\_data, axis = 1, keys = headers)

print(df3)

# Additional formatting for x axis.

x = df3.index.tolist()

xLabels = [str(value\*10) + " Million" for value in x]

y1 = df3['Average Tests'].tolist()

y2 = df3['Average Vaccinations'].tolist()

# Additional formatting for y axis.

maxY = max(max(y1), max(y2))

yTickIncrement = 100000

yTicks = []

yLabels = []

y = 0

while y <= maxY:

yTicks.append(y)

yLabels.append(str(y))

y += yTickIncrement

# Generate the plot.

xTicks = np.arange(len(x))

plt.bar(xTicks - 0.2, y1, 0.4, label = 'Tests')

plt.bar(xTicks + 0.2, y2, 0.4, label = 'Vaccinations')

plt.xticks(xTicks, xLabels, rotation=90)

plt.yticks(yTicks, yLabels)

plt.xlabel("Population Bin")

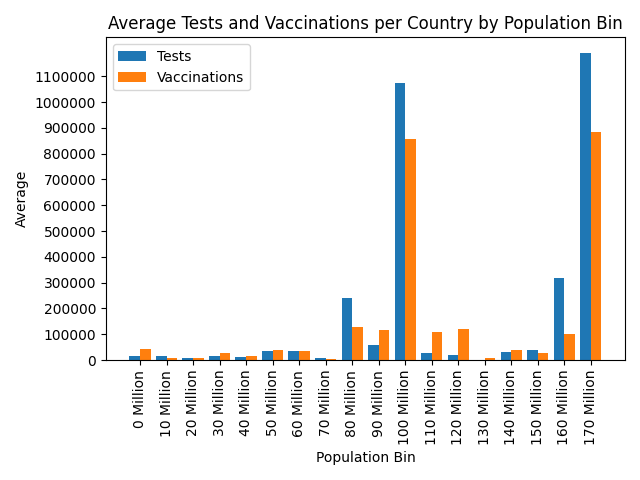
plt.ylabel("Average")

plt.title("Average Tests and Vaccinations per Country by Population Bin")

plt.legend()

plt.show()

**Chart**



**Evaluation**

The graph shown above is a group bar plot of the average tests and vaccinations administered per country by population bin. For this graph, the average values calculated are encoded in the y-axis (using scientific notation). Since this is a grouped bar chart, the x-axis shows all 17 bins of both tests and vaccinations. A group bar plot was chosen as it is the most effective in comparing both the average vaccinations and the average tests in accordance to each bin. While the chart is effective in communicating the results, there would definitely be problems if more data is added. As seen above, some averages are way lower than the others. If more data is added and countries in one bin have an increase in both vaccinations and tests, it would make it harder to read the bins with smaller values.

## 2.2 Section 2 - 480385312

### 2.2.1 Grouped Aggregate Summaries

#### 2.2.1.1 Sum of Vaccinations over Population (Per Country)

This aggregate summary calculates the total number of vaccinations for each country, and compares it to the population of this country. As part of this, a value for vaccinations per person is also calculated.

**Code**

import csv

def aggregateVaccinationsOverPopulationPerCountry(filename):

firstLine = True

# Dictionaries of country -> value.

populationPerCountry = {}

vaccinationsPerCountry = {}

csv\_reader = csv.reader(open(filename))

for row in csv\_reader:

# Ignore the header row.

if firstLine:

firstLine = False

else:

country = row[1]

population = int(row[3])

vaccinations = int(row[7])

# Set the country's population the first time it is seen.

if country not in populationPerCountry:

populationPerCountry[country] = population

# Calculate the total sum of vaccinations.

if country not in vaccinationsPerCountry:

vaccinationsPerCountry[country] = 0

vaccinationsPerCountry[country] += vaccinations

return populationPerCountry, vaccinationsPerCountry

if \_\_name\_\_ == "\_\_main\_\_":

populationPerCountry, vaccinationsPerCountry = aggregateVaccinationsOverPopulationPerCountry("integrated\_data.csv")

# Output the results.

for country in sorted(populationPerCountry):

perPerson = vaccinationsPerCountry[country] / populationPerCountry[country]

print("{} vaccinated {}/{} citizens ({} per person).".format(country, vaccinationsPerCountry[country], populationPerCountry[country], perPerson))

# Overall aggregate.

totalVaccinations = sum(vaccinationsPerCountry.values())

totalPopulation = sum(populationPerCountry.values())

totalPerPerson = totalVaccinations / totalPopulation

print("Total vaccinated {}/{} citizens ({} per person).".format(totalVaccinations, totalPopulation, totalPerPerson))

**Table of Results**

| **Country** | **Vaccinations / Population** | **Vaccines Per Person** |  | **Country** | **Vaccinations / Population** | **Vaccines Per Person** |
| --- | --- | --- | --- | --- | --- | --- |
| Argentina | 19493634 / 45196131 | 0.43 |  | Malaysia | 7118727 / 32365584 | 0.21 |
| Armenia | 73842 / 2963268 | 0.02 |  | Maldives | 505904 / 540511 | 0.93 |
| Australia | 7207899 / 25499772 | 0.28 |  | Mexico | 43107257 / 128933002 | 0.33 |
| Austria | 7491442 / 9006532 | 0.83 |  | Morocco | 18666763 / 36910339 | 0.50 |
| Bangladesh | 10099577 / 164690139 | 0.06 |  | Mozambique | 469350 / 31247497 | 0.01 |
| Belarus | 1416443 / 9449306 | 0.14 |  | Namibia | 137160 / 2540732 | 0.05 |
| Belgium | 10572768 / 11589789 | 0.91 |  | Nepal | 3369539 / 29134873 | 0.11 |
| Bolivia | 2390910 / 11672794 | 0.20 |  | Netherlands | 14888019 / 17135037 | 0.86 |
| Bosnia | 452211 / 3280646 | 0.13 |  | New Zealand | 1083989 / 5002100 | 0.21 |
| Botswana | 248177 / 2351396 | 0.10 |  | Nigeria | 3190285 / 206102237 | 0.01 |
| Bulgaria | 1726294 / 6947970 | 0.24 |  | North Macedonia | 505710 / 2083374 | 0.24 |
| Cabo Verde | 79424 / 555987 | 0.14 |  | Norway | 3962655 / 5421307 | 0.73 |
| Canada | 35157003 / 37742508 | 0.93 |  | Pakistan | 14227714 / 220873273 | 0.06 |
| Chile | 22545761 / 19116392 | 1.17 |  | Panama | 1483831 / 4314596 | 0.34 |
| Colombia | 17010892 / 50882948 | 0.33 |  | Paraguay | 858241 / 7132474 | 0.12 |
| Costa Rica | 2403658 / 5094159 | 0.47 |  | Peru | 7245288 / 32971130 | 0.21 |
| Cote d'Ivoire | 743642 / 26373506 | 0.02 |  | Philippines | 9543248 / 109579246 | 0.08 |
| Croatia | 2504578 / 4105053 | 0.61 |  | Poland | 27968706 / 37846363 | 0.73 |
| Cyprus | 785452 / 1207375 | 0.65 |  | Portugal | 8082070 / 10196505 | 0.79 |
| Denmark | 5017519 / 5792278 | 0.86 |  | Qatar | 2990644 / 2807805 | 1.06 |
| Dominican Republic | 7572978 / 10847957 | 0.69 |  | Romania | 8850491 / 19236577 | 0.46 |
| Ecuador | 3983941 / 17642457 | 0.22 |  | Russia | 37397628 / 145934790 | 0.25 |
| Estonia | 928793 / 1326540 | 0.70 |  | Rwanda | 632497 / 12949870 | 0.04 |
| Ethiopia | 1659798 / 114942793 | 0.01 |  | Saudi Arabia | 17041910 / 34812554 | 0.48 |
| Fiji | 304899 / 896457 | 0.34 |  | Senegal | 665595 / 16740327 | 0.03 |
| Finland | 4172249 / 5540760 | 0.75 |  | Serbia | 5200972 / 8737110 | 0.59 |
| France | 52451422 / 65274140 | 0.80 |  | Singapore | 5230097 / 5850412 | 0.89 |
| Germany | 72552356 / 83784978 | 0.86 |  | Slovakia | 3487005 / 5459656 | 0.63 |
| Ghana | 1331080 / 31069556 | 0.04 |  | Slovenia | 1456204 / 2078940 | 0.70 |
| Greece | 8015496 / 10422656 | 0.76 |  | South Africa | 2611532 / 59308021 | 0.04 |
| Guatemala | 917853 / 17914255 | 0.05 |  | South Korea | 18686650 / 51269407 | 0.36 |
| Hungary | 10281227 / 9660187 | 1.06 |  | Spain | 39186364 / 46754873 | 0.83 |
| Iceland | 380393 / 341248 | 1.11 |  | Sri Lanka | 3457293 / 21413552 | 0.16 |
| India | 312160148 / 1380011546 | 0.22 |  | Switzerland | 7192166 / 8654736 | 0.83 |
| Indonesia | 39339867 / 273524045 | 0.14 |  | Thailand | 8833567 / 69800715 | 0.12 |
| Iran | 5442938 / 83991882 | 0.06 |  | Togo | 396804 / 8277446 | 0.04 |
| Iraq | 921239 / 40217031 | 0.02 |  | Trinidad and Tobago | 256056 / 1399505 | 0.18 |
| Ireland | 3961202 / 4937777 | 0.80 |  | Turkey | 47204368 / 84339116 | 0.55 |
| Israel | 10777801 / 9197590 | 1.17 |  | Uganda | 850509 / 45725079 | 0.01 |
| Italy | 49304062 / 60461278 | 0.81 |  | United Kingdom | 76476577 / 67887024 | 1.12 |
| Jamaica | 256167 / 2961209 | 0.08 |  | United States | 321396535 / 331007570 | 0.97 |
| Latvia | 1110208 / 1885974 | 0.58 |  | Uruguay | 3770856 / 3473775 | 1.08 |
| Lithuania | 2197609 / 2721829 | 0.80 |  | Zambia | 150834 / 18379291 | 0.00 |
| Luxembourg | 510992 / 625950 | 0.81 |  | Zimbabwe | 1223274 / 14862513 | 0.08 |
| Madagascar | 196768 / 27685459 | 0.00 |  | OVERALL | 1519213496 / 4708894347 | 0.32 |

#### 2.2.1.2 Sum of Vaccinations over Population (Per GDP Bin)

This aggregate summary calculates the total number of vaccinations for each country and compares it to the country population, similar to the above summary. This time, however, each country is placed into one of five bins based on its GDP for the 2020 financial year. The proportion of vaccines to population (and the value of vaccines per person) is based on the total sum of vaccines and populations for all countries within the GDP bin.

**Code**

import csv

def aggregateVaccinationsOverPopulationPerGDPBin(filename, binRange=1000000000000):

firstLine = True

# Dictionaries of gdpBin -> value.

populationPerGDPBin = {}

vaccinationsPerGDPBin = {}

countriesCounted = set()

csv\_reader = csv.reader(open("integrated\_data.csv"))

for row in csv\_reader:

# Ignore the header row.

if firstLine:

firstLine = False

else:

country = row[1]

gdp = int(float(row[2]))

population = int(row[3])

vaccinations = int(row[7])

# Determine the bin this entry belongs to.

gdpBin = gdp // binRange

# Add the country's population the first time it is seen.

if country not in countriesCounted:

countriesCounted.add(country)

if gdpBin not in populationPerGDPBin:

populationPerGDPBin[gdpBin] = 0

populationPerGDPBin[gdpBin] += population

# Calculate the total sum of vaccinations.

if gdpBin not in vaccinationsPerGDPBin:

vaccinationsPerGDPBin[gdpBin] = 0

vaccinationsPerGDPBin[gdpBin] += vaccinations

return populationPerGDPBin, vaccinationsPerGDPBin

if \_\_name\_\_ == "\_\_main\_\_":

binRange = 1000000000000

populationPerGDPBin, vaccinationsPerGDPBin = aggregateVaccinationsOverPopulationPerGDPBin("integrated\_data.csv")

# Output the results.

for gdpBin in sorted(populationPerGDPBin):

perPerson = vaccinationsPerGDPBin[gdpBin] / populationPerGDPBin[gdpBin]

print("Countries with a GDP over ${} USD vaccinated {}/{} citizens ({} per person).".format(gdpBin \* binRange, vaccinationsPerGDPBin[gdpBin], populationPerGDPBin[gdpBin], perPerson))

# Overall aggregate.

totalVaccinations = sum(vaccinationsPerGDPBin.values())

totalPopulation = sum(populationPerGDPBin.values())

totalPerPerson = totalVaccinations / totalPopulation

print("Total vaccinated {}/{} citizens ({} per person).".format(totalVaccinations, totalPopulation, totalPerPerson))

**Table of Results**

| **GDP Range** | **Vaccinations / Population** | **Vaccines Per Person** |
| --- | --- | --- |
| < $1 Billion USD | 414789728 / 2010809414 | 0.20 |
| ≥ $1 Billion USD, < $2 Billion USD | 269386730 / 770119675 | 0.34 |
| ≥ $2 Billion USD, < $3 Billion USD | 441088147 / 1513172710 | 0.29 |
| ≥ $3 Billion USD, < $4 Billion USD | 72552356 / 83784978 | 0.86 |
| ≥ $4 Billion USD | 321396535 / 331007570 | 0.97 |
| OVERALL | 1519213496 / 4708894347 | 0.32 |

### 2.2.2 Charts

#### 2.2.2.1 Sum of Vaccinations over Population (per Country - Overlay Bar Plot)

**Code**

from vaccinationsOverPopulationPerCountry import aggregateVaccinationsOverPopulationPerCountry

import matplotlib.pyplot as plt

# Calculate the results.

populationPerCountry, vaccinationsPerCountry = aggregateVaccinationsOverPopulationPerCountry("integrated\_data.csv")

countries = sorted(populationPerCountry.keys())

# Format the results.

formattedPopulations = []

formattedVaccinations = []

for country in countries:

formattedPopulations.append(populationPerCountry[country])

formattedVaccinations.append(vaccinationsPerCountry[country])

# Additional formatting for y axis.

maxY = max(populationPerCountry.values())

yTickIncrement = 100000000

yTicks = []

yLabels = []

y = 0

while y <= maxY:

yTicks.append(y)

yLabels.append(str(y // 1000000) + " Million")

y += yTickIncrement

# Split the countries into separate graphs, so that the x axis isn't too crowded.

maxCountriesPerGraph = 15

offset = 0

while offset < len(countries):

# Create the plot.

plt.bar(countries[offset:offset+maxCountriesPerGraph], formattedPopulations[offset:offset+maxCountriesPerGraph], label="Population", color='r', alpha=0.5)

plt.bar(countries[offset:offset+maxCountriesPerGraph], formattedVaccinations[offset:offset+maxCountriesPerGraph], label="Vaccinations", color='b', alpha=0.5)

plt.xticks(rotation=90)

plt.yticks(yTicks, yLabels)

plt.xlabel("Country")

plt.ylabel("Number of people")

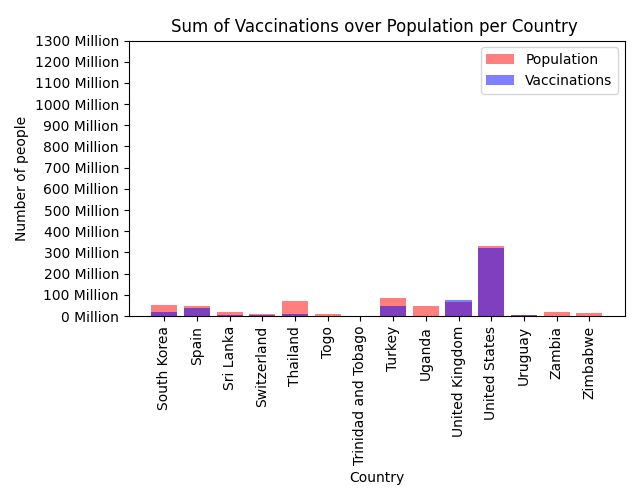
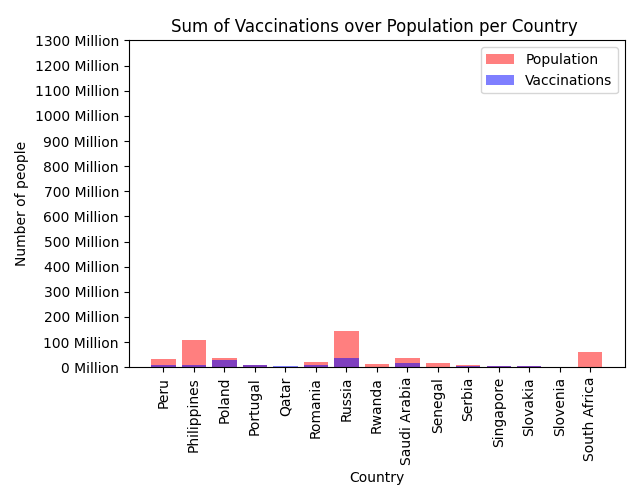
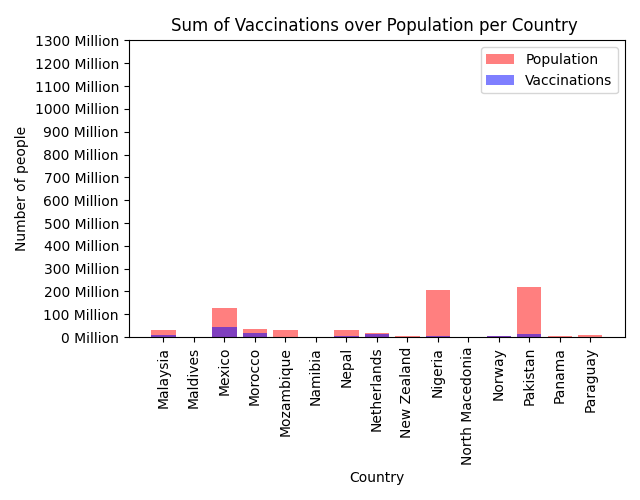
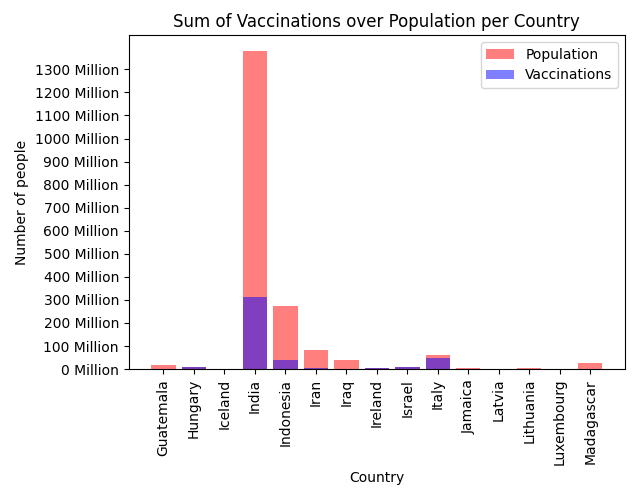
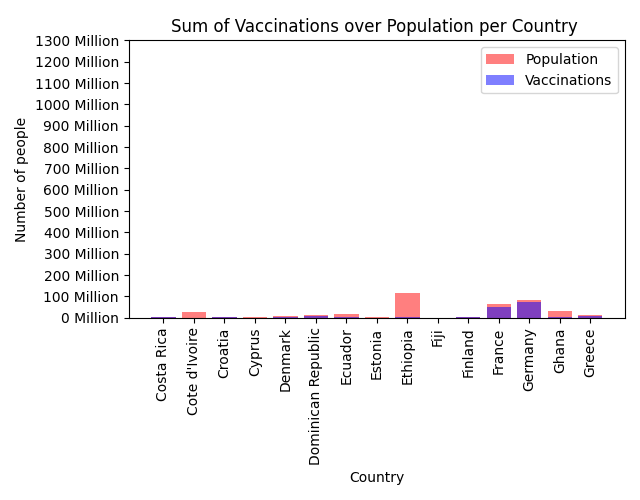
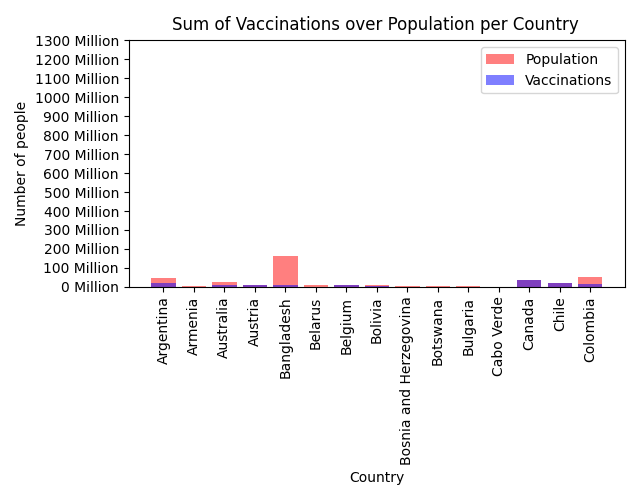
plt.legend()

plt.title("Sum of Vaccinations over Population per Country")

plt.show()

offset += maxCountriesPerGraph

**Chart**



**Evaluation**

This series of charts are actually a single chart with its x axis (countries) being partitioned to create separate charts, in order to avoid overcrowding. For each country, its population is shown as a semi-transparent red bar, and its total number of vaccinations administered is overlaid as a semi-transparent blue bar. In most cases, the number of vaccinations administered is less than the population, resulting in a purple portion of the bar, capped with a red portion. In cases where the number of vaccinations is greater than the population, the bar is instead capped with a blue portion (e.g. United Kingdom).

The overlay bar chart was chosen to allow for easy comparison between vaccinations and population for each country - while a side-by-side bar chart was considered, it was deemed easier to compare the two values by utilising the transparency and overlaying.

These charts, while useful for some countries, fall flat due to the extreme differences in population between some countries. Outliers such as India cause the y axis scaling to be such that it is difficult to properly gauge the values for the other, lower-population countries. This problem could become worse as more data is introduced, especially if the new countries have populations or vaccinations that cause a change to the range of values seen.

#### 2.2.2.2 Vaccinations per Person (per Country - Bar Plot)

**Code**

from vaccinationsOverPopulationPerCountry import aggregateVaccinationsOverPopulationPerCountry

import matplotlib.pyplot as plt

# Calculate the results.

populationPerCountry, vaccinationsPerCountry = aggregateVaccinationsOverPopulationPerCountry("integrated\_data.csv")

countries = sorted(populationPerCountry.keys())

# Format the results.

formattedPerPerson = []

for country in countries:

formattedPerPerson.append(vaccinationsPerCountry[country] / populationPerCountry[country])

# Additional formatting for y axis.

maxY = max(formattedPerPerson)

yTickIncrement = 0.1

yTicks = []

y = 0

while y <= maxY:

yTicks.append(y)

y += yTickIncrement

# Split the countries into separate graphs, so that the x axis isn't too crowded.

maxCountriesPerGraph = 15

offset = 0

while offset < len(countries):

# Create the plot.

plt.bar(countries[offset:offset+maxCountriesPerGraph], formattedPerPerson[offset:offset+maxCountriesPerGraph], color='b')

plt.xticks(rotation=90)

plt.yticks(yTicks)

plt.xlabel("Country")

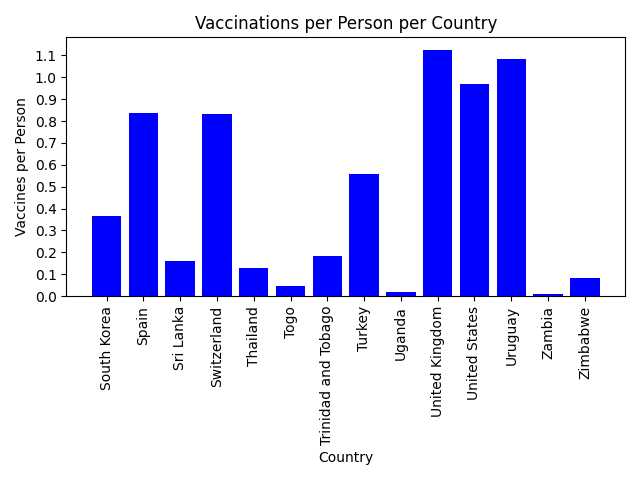
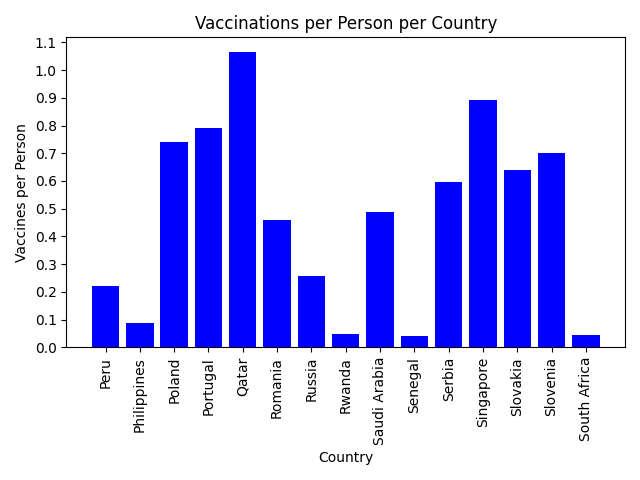
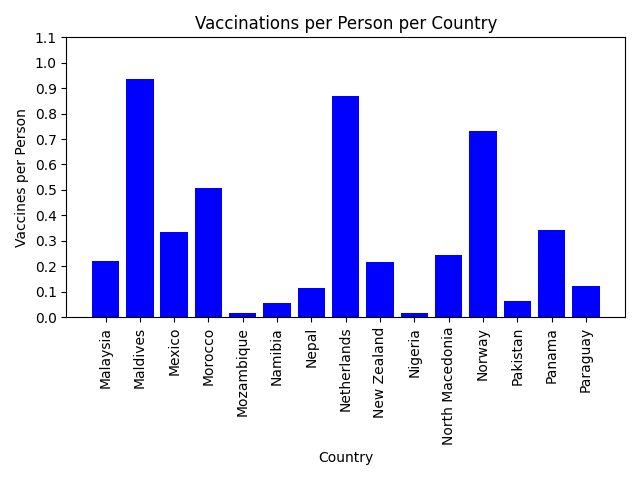
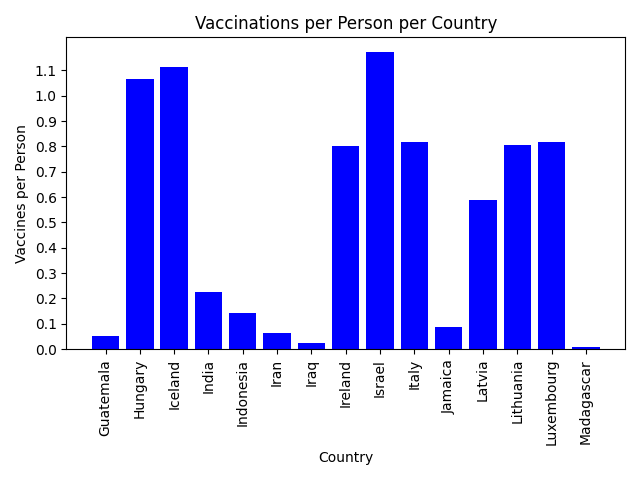
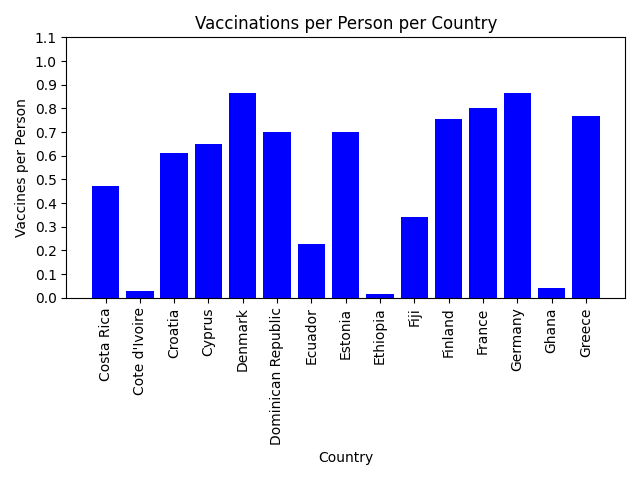
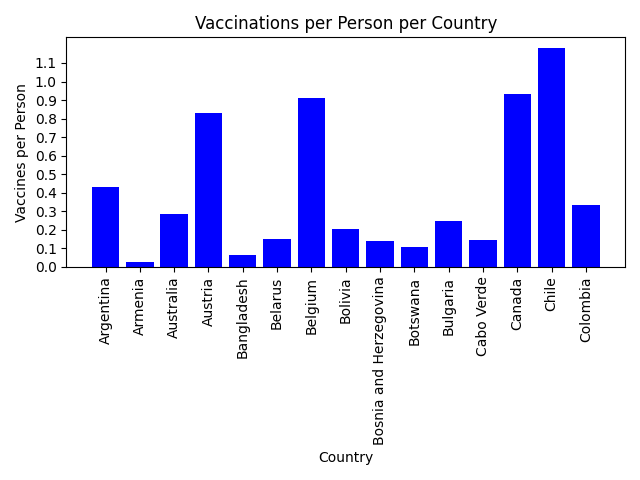
plt.ylabel("Vaccines per Person")

plt.title("Vaccinations per Person per Country")

plt.show()

offset += maxCountriesPerGraph

**Chart**



**Evaluation**

This series of charts are another way of representing the information from [Section 2.2.2.1](#_rpzer7wd10e5). Here, a figure is shown that normalises the vaccination and population counts, allowing for more direct comparison between countries. This normalised view makes it much easier to see the discrepancies between countries (e.g. between Uganda and the United Kingdom), and is overall more useful than the set of charts presented above.

Since these charts are an evolution from the previous set, they are also a bar plot, though they do not show any overlay. An ordered bar plot was considered for this set, and this would indeed serve as a useful tool for analysis. The countries were kept in alphabetical order purely for ease of finding a particular country.

These charts would adapt well to receiving more data, since the scaling issues on the y axis have been resolved. The addition of more countries would potentially create additional charts, but the generation code is already set up to automatically split and create a new chart when the current one becomes too crowded with countries.

Note that some countries report a vaccinations per person count of over 1.0. This is to be expected, since many vaccinations brands require multiple doses, and this metric is specifically counting the number of doses. For example, a vaccinations per person value of 1.5 would imply that each person in that country has received 1.5 doses of a vaccine.

#### 2.2.2.3 Vaccinations per Person (per GDP Bin - Line Plot)

**Code**

from vaccinationsOverPopulationPerGDPBin import aggregateVaccinationsOverPopulationPerGDPBin

import matplotlib.pyplot as plt

# Calculate the results.

populationPerGDPBin, vaccinationsPerGDPBin = aggregateVaccinationsOverPopulationPerGDPBin("integrated\_data.csv")

gdpBins = sorted(populationPerGDPBin.keys())

# Format the results.

formattedPerPerson = []

for gdpBin in gdpBins:

formattedPerPerson.append(vaccinationsPerGDPBin[gdpBin] / populationPerGDPBin[gdpBin])

# Additional formatting for x axis.

xTicks = []

xLabels = []

for x in range(1, len(gdpBins) + 1):

xTicks.append(x)

if x == 1:

xLabels.append("< $" + str(x) + " Billion USD")

elif x == len(gdpBins):

xLabels.append("$" + str(x) + " Billion USD +")

else:

xLabels.append("$" + str(x) + " Billion USD")

# Additional formatting for y axis.

maxY = max(formattedPerPerson)

yTickIncrement = 0.1

yTicks = []

y = 0

while y <= maxY:

yTicks.append(y)

y += yTickIncrement

# Create the plot.

plt.plot(xTicks, formattedPerPerson, color='b')

plt.xticks(xTicks, xLabels, rotation=90)

plt.yticks(yTicks)

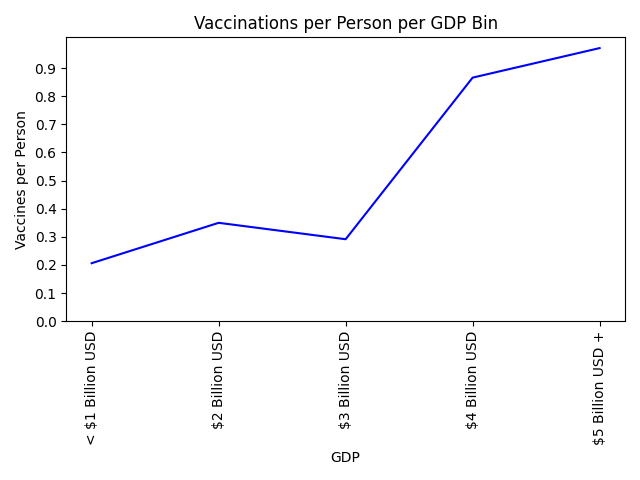
plt.xlabel("GDP")

plt.ylabel("Vaccines per Person")

plt.title("Vaccinations per Person per GDP Bin")

plt.show()

**Chart**



**Evaluation**

This chart attempts to demonstrate the correlation between a country’s GDP and its vaccinations per person metric (discussed above). The points of the x axis represent GDP bins, where each bin contains countries with a GDP greater than or equal to the value displayed for the previous bin but less than the value displayed for its data point. Note: these ranges are more clearly defined in [Section 2.2.1.2](#_td4mgu1h0dqv).

A line plot was chosen over a bar plot for this chart, since unlike individual countries, it is valid to directly compare the change in vaccinations per person between GDP bins (since GDP is a quantitative attribute).

Like the previous set of charts, this chart extends well to additional data, since new countries would likely fit into one of the existing GDP bins, and adding more data points to an individual bin is of no consequence. Additional countries may result in the creation of more GDP bins, which could potentially cause overcrowding of the x axis, but this is unlikely given the range of real-world GDP values.

## 2.3 Section 3 - 510465739

### 2.3.1 Grouped Aggregate Summaries

#### 2.3.1.1 Average Number of Deaths Binned by Tests

**Code**

import csv

first\_line = True

deaths\_by\_test\_bin = {}

# deaths\_by\_test\_bin is a dictionary

# key is the test bin

# value is a list of the number of deaths

average\_deaths\_by\_test\_bin = {}

# average\_deaths\_by\_test\_bin is a dictionary

# key is the test bin

# value is the average number of deaths

csv\_reader = csv.reader(open("integrated\_data.csv"))

for row in csv\_reader:

if first\_line:

first\_line = False

else:

deaths = int(row[5])

tests = int(row[6])

test\_bin = tests // 100000

if test\_bin not in deaths\_by\_test\_bin:

deaths\_by\_test\_bin[test\_bin] = [deaths]

else:

deaths\_by\_test\_bin[test\_bin].append(deaths)

for test\_bin in deaths\_by\_test\_bin:

average\_deaths = round((sum(deaths\_by\_test\_bin[test\_bin]) / len(deaths\_by\_test\_bin[test\_bin])))

average\_deaths\_by\_test\_bin[test\_bin] = average\_deaths

# Overall Aggregate

total\_deaths = 0

total\_test\_bins = 0

for key in sorted(average\_deaths\_by\_test\_bin):

total\_deaths += average\_deaths\_by\_test\_bin[key]

total\_test\_bins += 1

average\_deaths\_by\_test\_bin["OVERALL"] = round(total\_deaths / total\_test\_bins)

print(average\_deaths\_by\_test\_bin)

**Table of Results**

| **Test Bin (by 100000s)** | **Average Number of Deaths** |  | **Test Bin (by 100000s)** | **Average Number of Deaths** |
| --- | --- | --- | --- | --- |
| 0 | 48 |  | 16 | 1422 |
| 1 | 126 |  | 17 | 2293 |
| 2 | 215 |  | 18 | 2769 |
| 3 | 314 |  | 19 | 2652 |
| 4 | 332 |  | 20 | 3165 |
| 5 | 467 |  | 21 | 2523 |
| 6 | 597 |  | 22 | 2450 |
| 7 | 392 |  | 23 | 2129 |
| 8 | 597 |  | 24 | 1258 |
| 9 | 673 |  | 29 | 58 |
| 10 | 819 |  | 32 | 3207 |
| 11 | 737 |  | 35 | 2677 |
| 12 | 1071 |  | 36 | 2713 |
| 13 | 984 |  | 37 | 3134 |
| 14 | 1309 |  | OVERALL | 1419 |
| 15 | 1425 |  |  |  |

#### 2.3.1.2 Average Number of Tests Grouped by Country

**Code**

import csv

first\_line = True

daily\_tests\_per\_country = {}

# daily\_tests\_per\_country is a dictionary.

# Key is the country name

# Value is a list of the daily tests for that country

average\_tests\_per\_country = {}

# average\_tests\_per\_country is a dictionary.

# Key is the country name

# Value is the average number of daily tests for that country

csv\_reader = csv.reader(open("integrated\_data.csv"))

for row in csv\_reader:

if first\_line:

first\_line = False

else:

country = row[1]

daily\_tests = int(row[6])

if country not in daily\_tests\_per\_country:

daily\_tests\_per\_country[country] = [daily\_tests]

else:

daily\_tests\_per\_country[country].append(daily\_tests)

for key in daily\_tests\_per\_country:

average\_tests = round(sum(daily\_tests\_per\_country[key]) / len(daily\_tests\_per\_country[key]))

average\_tests\_per\_country[key] = average\_tests

# Overall Aggregate

total\_num\_of\_tests = 0

total\_num\_of\_countries = 0

for key in average\_tests\_per\_country:

total\_num\_of\_tests += average\_tests\_per\_country[key]

total\_num\_of\_countries += 1

average\_tests\_per\_country["OVERALL"] = round(total\_num\_of\_tests / total\_num\_of\_countries)

print(average\_tests\_per\_country)

**Table of Results**

| **Country** | **Average Tests** |  | **Country** | **Average Tests** |
| --- | --- | --- | --- | --- |
| Argentina | 45062 |  | Malaysia | 45980 |
| Armenia | 1732 |  | Maldives | 2646 |
| Australia | 27874 |  | Mexico | 17931 |
| Austria | 148874 |  | Morocco | 15020 |
| Bangladesh | 15938 |  | Mozambique | 1485 |
| Belarus | 412 |  | Namibia | 1334 |
| Belgium | 38498 |  | Nepal | 8265 |
| Bolivia | 4385 |  | Netherlands | 0 |
| Bosnia and Herzegovina | 1858 |  | New Zealand | 4018 |
| Botswana | 7 |  | Nigeria | 1694 |
| Bulgaria | 7124 |  | North Macedonia | 1888 |
| Cabo Verde | 434 |  | Norway | 16124 |
| Canada | 92887 |  | Pakistan | 32870 |
| Chile | 42865 |  | Panama | 7701 |
| Colombia | 52240 |  | Paraguay | 3974 |
| Costa Rica | 3297 |  | Peru | 11420 |
| Cote d'Ivoire | 1711 |  | Philippines | 37208 |
| Croatia | 4998 |  | Poland | 42922 |
| Cyprus | 22052 |  | Portugal | 33075 |
| Denmark | 93995 |  | Qatar | 4838 |
| Dominican Republic | 2935 |  | Romania | 10192 |
| Ecuador | 3728 |  | Russia | 317768 |
| Estonia | 3861 |  | Rwanda | 2816 |
| Ethiopia | 7164 |  | Saudi Arabia | 55322 |
| Fiji | 527 |  | Senegal | 1359 |
| Finland | 14297 |  | Serbia | 11008 |
| France | 251768 |  | Singapore | 0 |
| Germany | 0 |  | Slovakia | 109722 |
| Ghana | 2023 |  | Slovenia | 3422 |
| Greece | 25838 |  | South Africa | 31031 |
| Guatemala | 3435 |  | South Korea | 22200 |
| Hungary | 14822 |  | Spain | 0 |
| Iceland | 984 |  | Sri Lanka | 10786 |
| India | 1073862 |  | Switzerland | 21055 |
| Indonesia | 26973 |  | Thailand | 21407 |
| Iran | 13226 |  | Togo | 909 |
| Iraq | 22913 |  | Trinidad and Tobago | 611 |
| Ireland | 13976 |  | Turkey | 156880 |
| Israel | 44760 |  | Uganda | 2588 |
| Italy | 180897 |  | United Kingdom | 512407 |
| Jamaica | 437 |  | United States | 1191183 |
| Latvia | 7394 |  | Uruguay | 5070 |
| Lithuania | 9578 |  | Zambia | 4840 |
| Luxembourg | 7959 |  | Zimbabwe | 1660 |
| Madagascar | 286 |  | OVERALL | 57579 |

### 2.3.2 Charts

#### 2.3.2.1 Average Number of Tests per million People (per Country - Barplot)

**Code**

import pandas as pd

import matplotlib.pyplot as plt

import csv

# Find average tests per million per country

first\_line = True

daily\_tests\_per\_million\_country = {}

# daily\_tests\_per\_country is a dictionary.

# Key is the country name

# Value is a list of the daily tests for that country

average\_tests\_per\_country = {}

# average\_tests\_per\_country is a dictionary.

# Key is the country name

# Value is the average number of daily tests for that country

csv\_reader = csv.reader(open("integrated\_data.csv"))

for row in csv\_reader:

if first\_line:

first\_line = False

else:

country = row[1]

daily\_tests = int(row[6])

population = int(row[3])

tests\_per\_million = (daily\_tests / population) \* 1000000

if country not in daily\_tests\_per\_million\_country:

daily\_tests\_per\_million\_country[country] = [tests\_per\_million]

else:

daily\_tests\_per\_million\_country[country].append(tests\_per\_million)

for key in daily\_tests\_per\_million\_country:

average\_tests = round(sum(daily\_tests\_per\_million\_country[key]) / len(daily\_tests\_per\_million\_country[key]))

average\_tests\_per\_country[key] = average\_tests

# Convert dictionary into pandas dataframe

data\_list = list(average\_tests\_per\_country.items())

df = pd.DataFrame(data\_list)

# From website: https://www.kite.com/python/answers/how-to-convert-a-dictionary-into-a-pandas-dataframe-in-python

# Produce chart

max\_countries\_per\_graph = 15

offset = 0

while offset < len(country):

plt.bar(country[offset:offset+max\_countries\_per\_graph], tests[offset:offset+max\_countries\_per\_graph]) # To split the countries over multiple graphs to avoid overcrowding the x-axis

plt.title("Average COVID-19 Tests per Million People per Country")

plt.xlabel("Country")

plt.ylabel("Average tests per million people")

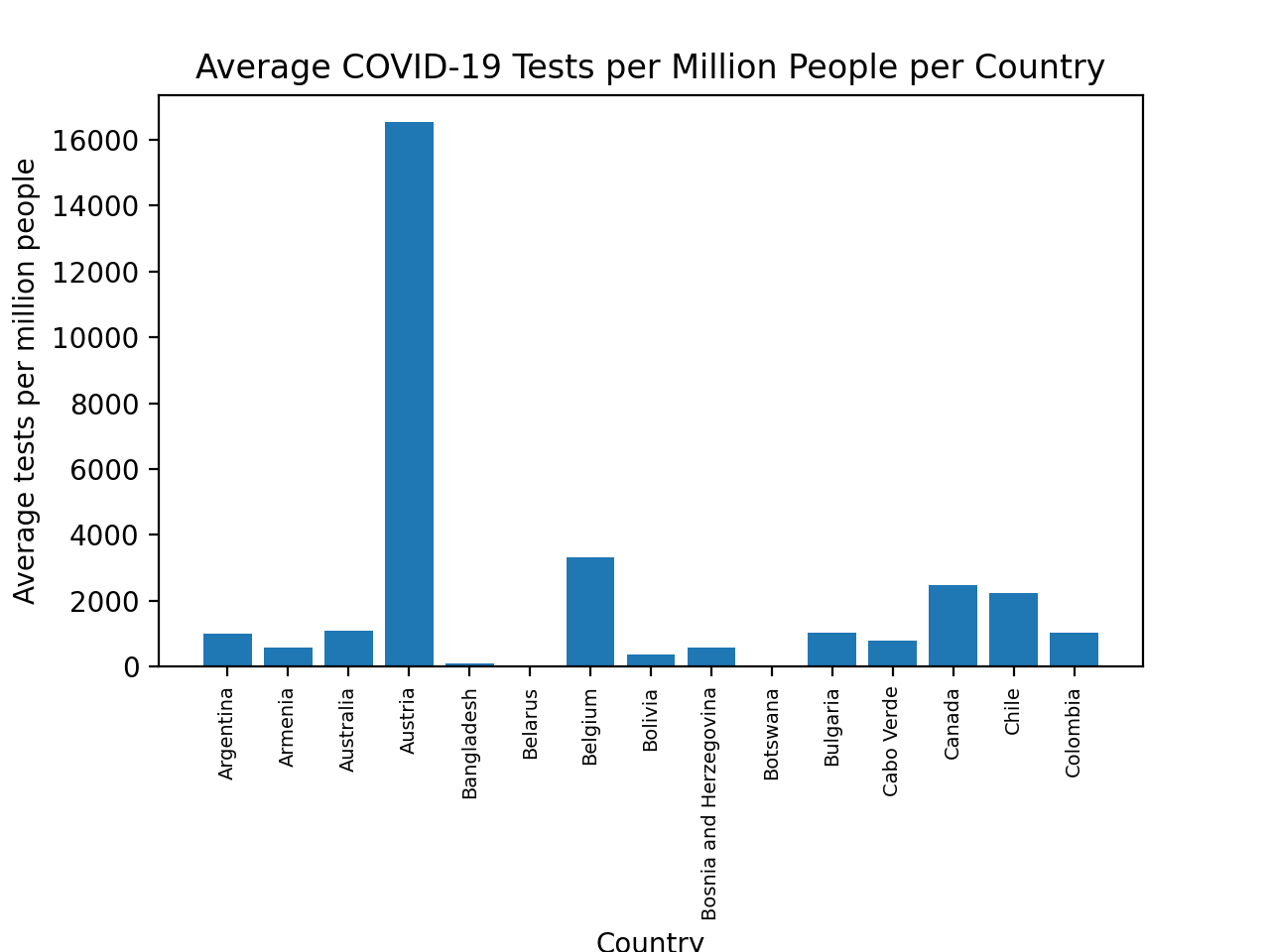
plt.xticks(rotation=90, fontsize=7)

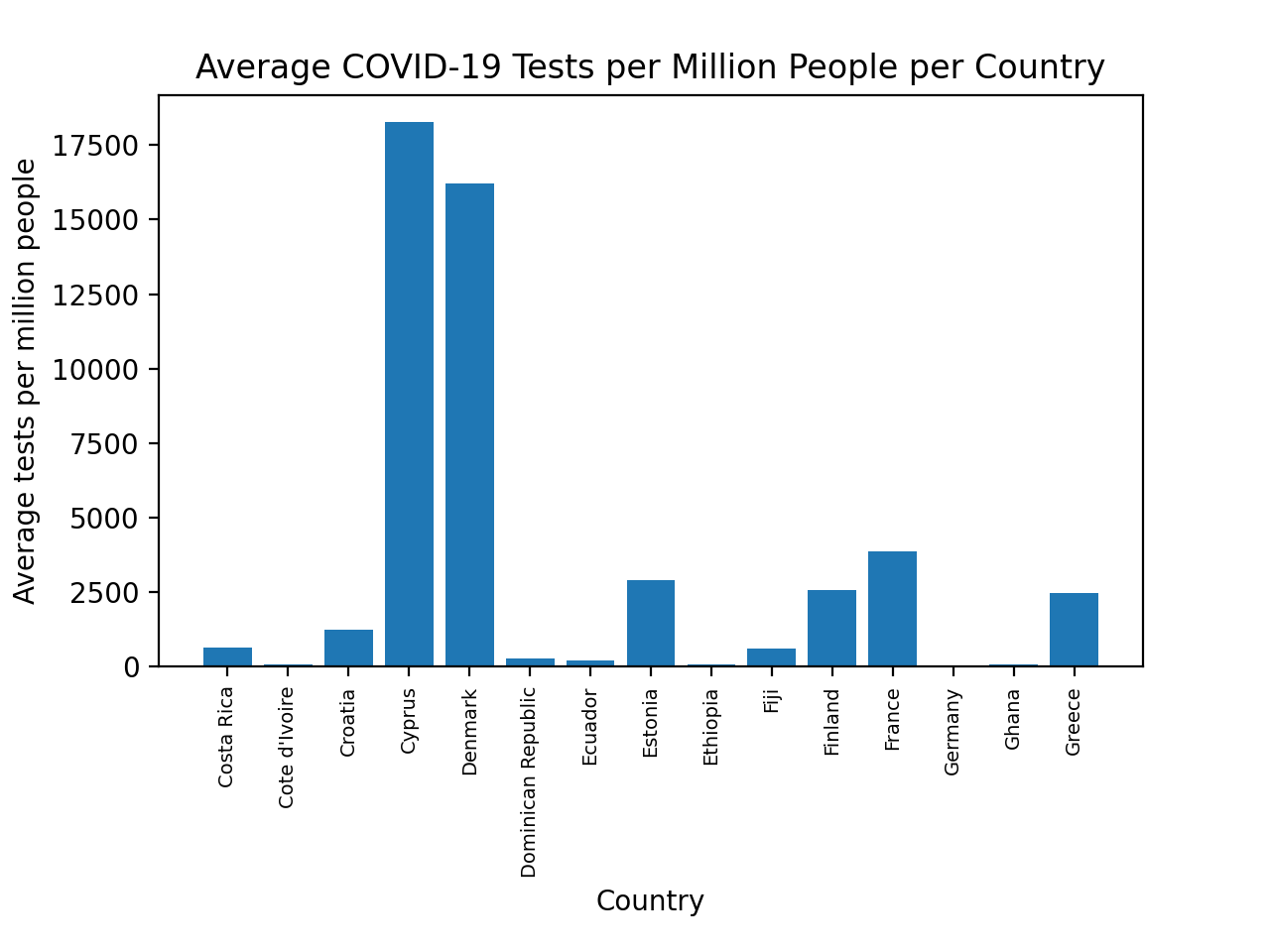
plt.subplots\_adjust(top=0.9, bottom=0.3)

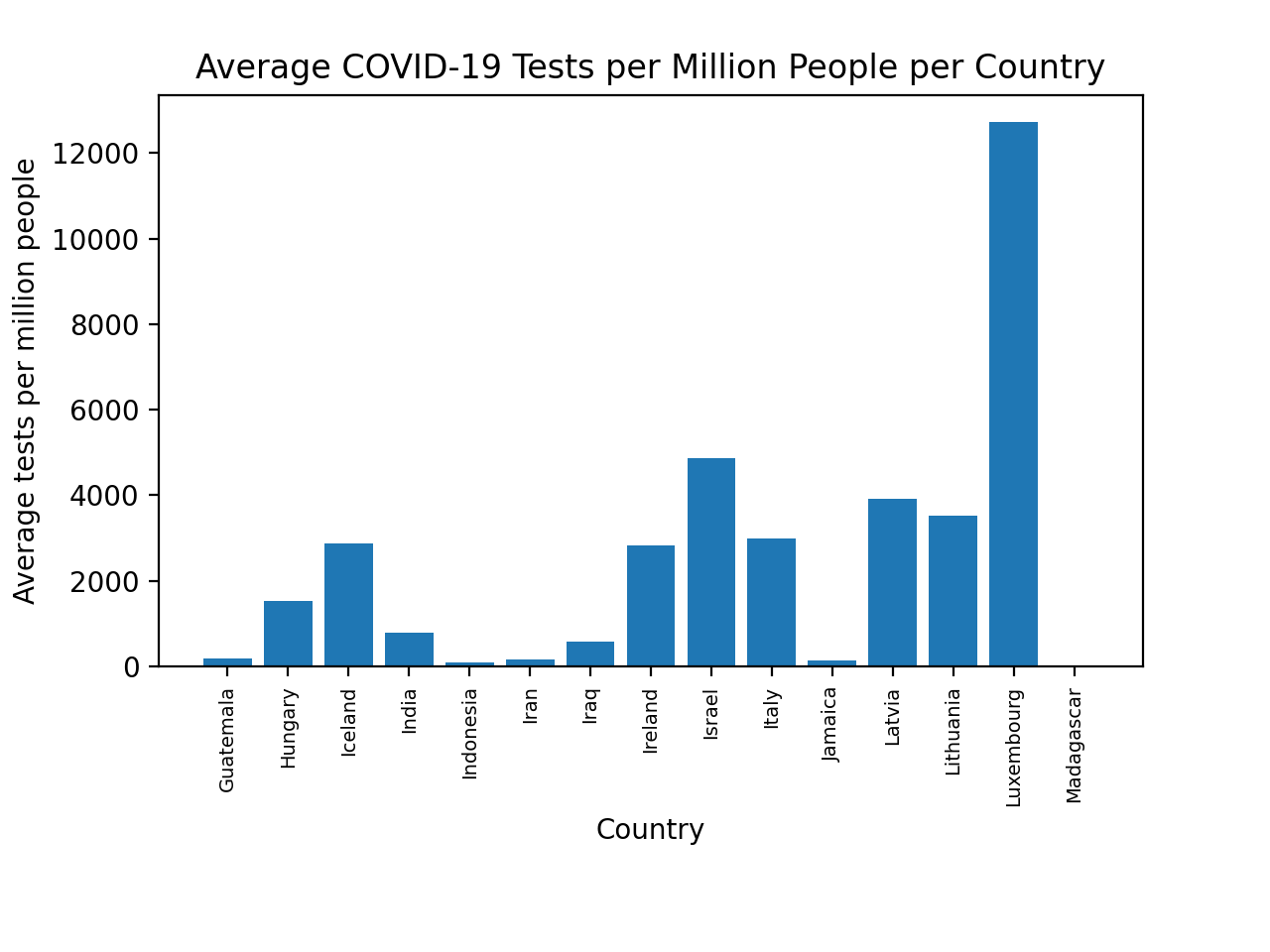
plt.show()

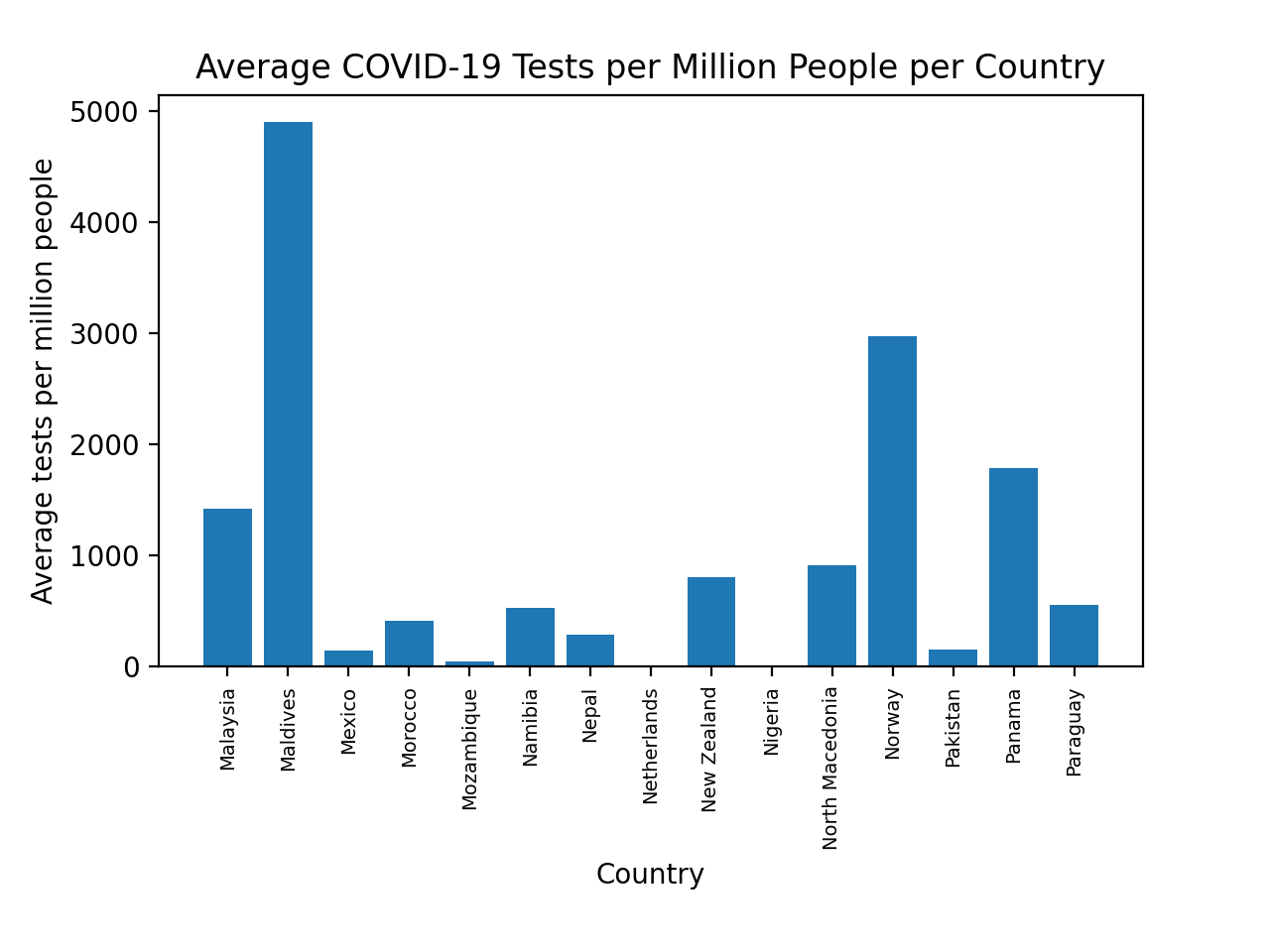
offset += max\_countries\_per\_graph

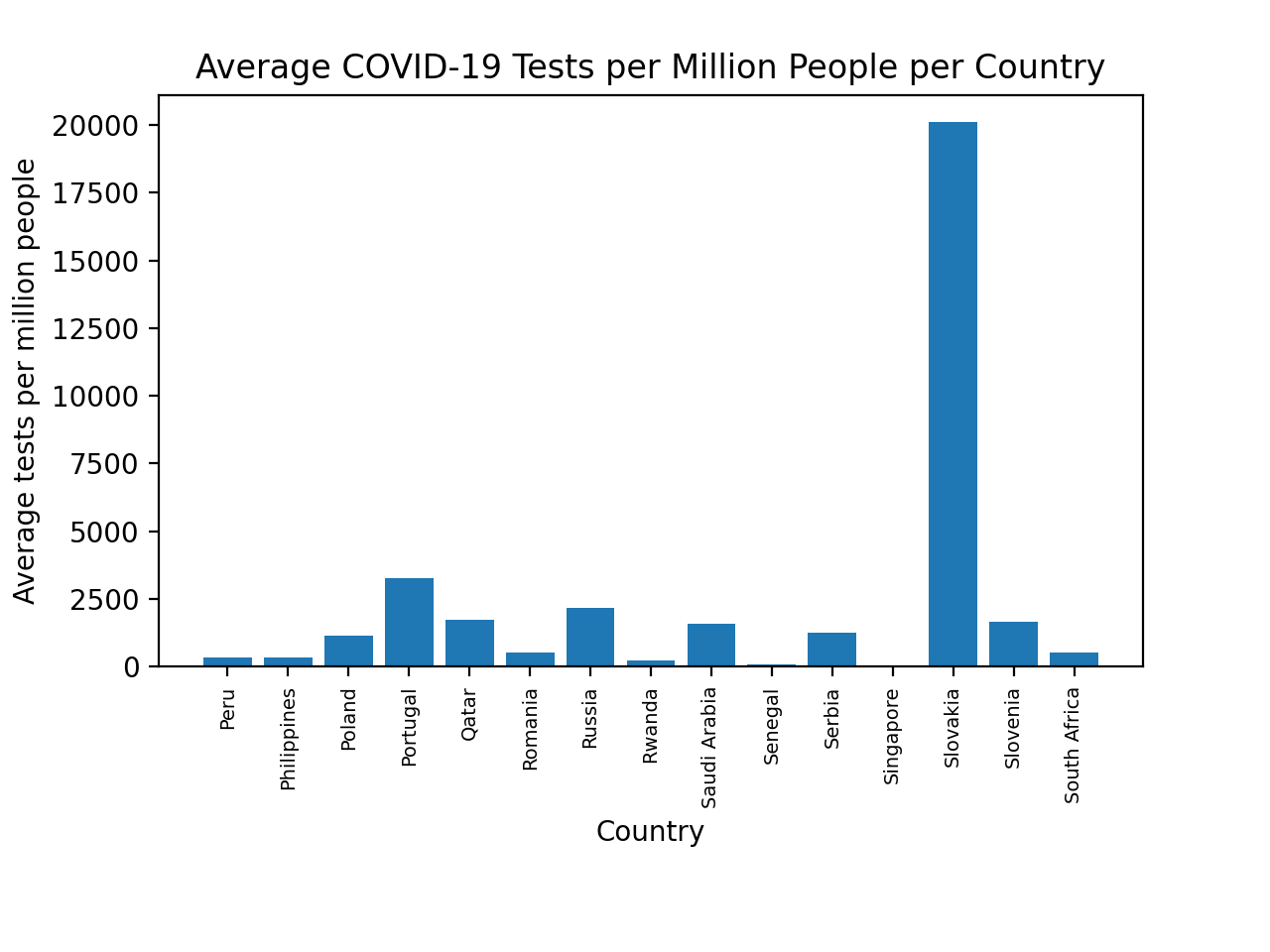
**Charts**

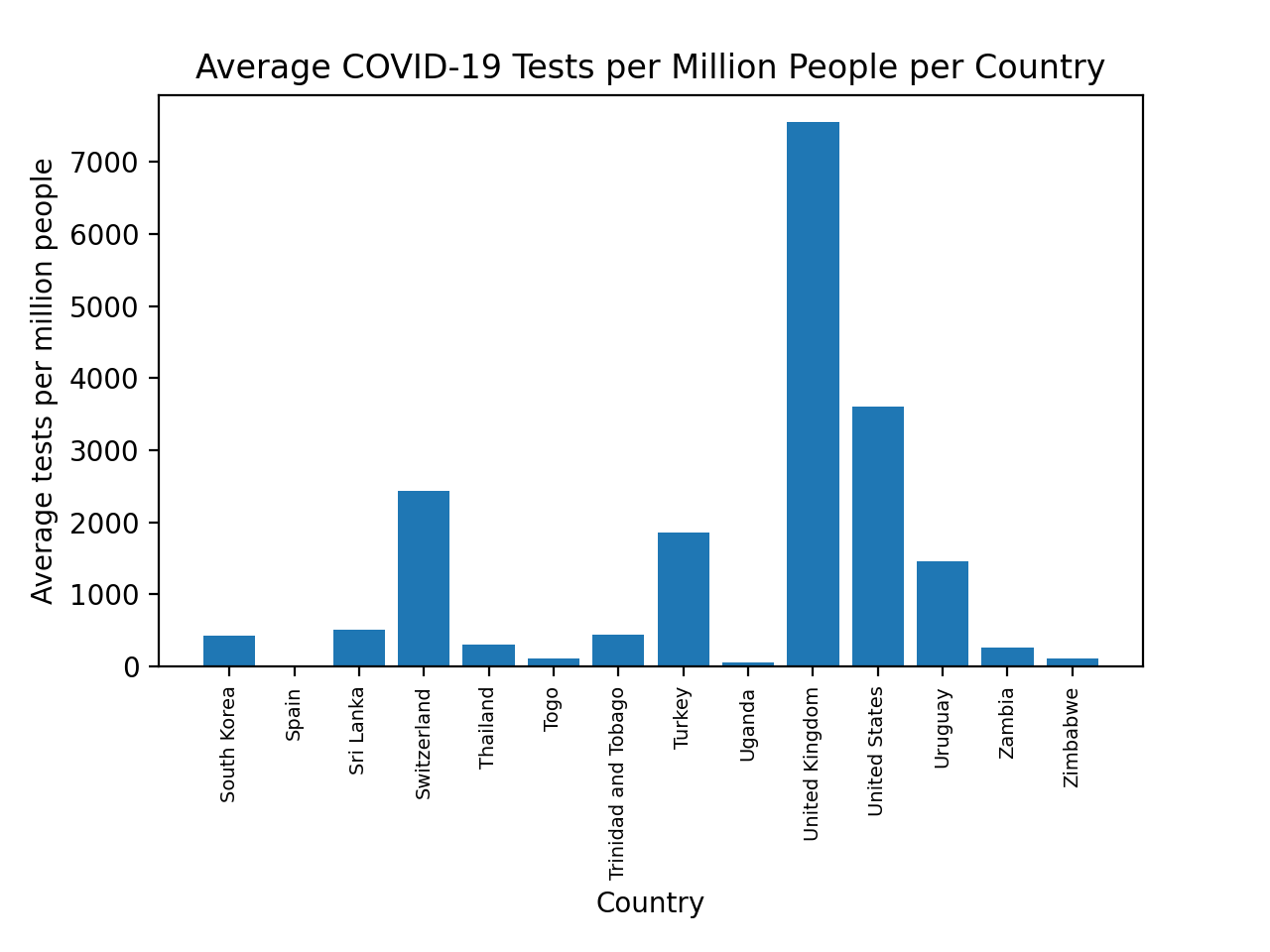












**Evaluation**

In these bar plots the average number of tests per million people in the population is encoded in the height of each bar according to the scale on the y-axis for each country on the x-axis. A bar plot was picked for this chart as it allows for effective comparison between countries as the difference in height of each bar can be easily seen. The chat has been split into 6 charts to avoid crowding the x-axis due to the large amount of countries. While this chart is effective at communicating the data, if more data was added some issues may cause it to be less effective. If more data for the existing countries was added the chart would still communicate effectively as the only change would be the change of height in the bars. However, if one particular country had a significant increase in their average number of tests per million it could make the countries with smaller values harder to read as the height of their bars would decrease. Furthermore if data for other countries was added it would cause the graph to become crowded and harder to read. This could be remedied however by splitting the charts into multiple charts with a smaller number of countries as has been done.

#### 

#### 

#### 

#### 

#### 2.3.2.2 Average Number of Cases and Deaths (per Vaccination Bin - Line Plot)

**Code**

import pandas as pd

import matplotlib.pyplot as plt

import csv

# Find average deaths and cases per vaccination bin

first\_line = True

deaths\_by\_vaccination\_bin = {}

# deaths\_by\_vaccination\_bin is a dictionary

# key is the vaccination bin

# value is a list of the number of deaths

average\_deaths\_by\_vaccination\_bin = {}

# average\_deaths\_by\_vaccination\_bin is a dictionary

# key is the vaccination bin

# value is the average number of deaths

cases\_by\_vaccination\_bin = {}

# cases\_by\_vaccination\_bin is a dictionary

# key is the vaccination bin

# value is a list of the number of cases

average\_cases\_by\_vaccination\_bin = {}

# average\_cases\_by\_vaccination\_bin is a dictionary

# key is the vaccination bin

# value is the average number of cases

csv\_reader = csv.reader(open("integrated\_data.csv"))

for row in csv\_reader:

if first\_line:

first\_line = False

else:

deaths = int(row[5])

cases = int(row[4])

vaccinations = int(row[-2])

vaccination\_bin = int(vaccinations // 500000)

if vaccination\_bin not in deaths\_by\_vaccination\_bin:

deaths\_by\_vaccination\_bin[vaccination\_bin] = [deaths]

cases\_by\_vaccination\_bin[vaccination\_bin] = [cases]

else:

deaths\_by\_vaccination\_bin[vaccination\_bin].append(deaths)

cases\_by\_vaccination\_bin[vaccination\_bin].append(cases)

for vaccination\_bin in sorted(deaths\_by\_vaccination\_bin):

average\_deaths = round((sum(deaths\_by\_vaccination\_bin[vaccination\_bin]) / len(deaths\_by\_vaccination\_bin[vaccination\_bin])))

average\_deaths\_by\_vaccination\_bin[vaccination\_bin] = average\_deaths

for vaccination\_bin in sorted(cases\_by\_vaccination\_bin):

average\_cases = round((sum(cases\_by\_vaccination\_bin[vaccination\_bin]) / len(cases\_by\_vaccination\_bin[vaccination\_bin])))

average\_cases\_by\_vaccination\_bin[vaccination\_bin] = average\_cases

# Convert dictionaries to pandas dataframe

data\_list1 = list(average\_deaths\_by\_vaccination\_bin.items())

df1 = pd.DataFrame(data\_list1)

data\_list2 = list(average\_cases\_by\_vaccination\_bin.items())

df2 = pd.DataFrame(data\_list2)

# From website: https://www.kite.com/python/answers/how-to-convert-a-dictionary-into-a-pandas-dataframe-in-python

combined\_data = [df1[0], df1[1], df2[1]]

headers = ["Vaccinations", "Deaths", "Cases"]

df3 = pd.concat(combined\_data, axis=1, keys=headers)

# From website: https://www.kite.com/python/answers/how-to-create-a-pandas-dataframe-from-columns-in-other-dataframes-in-python

df3 = df3.set\_index("Vaccinations")

# Produce chart

x = df3.index.tolist()

y1 = df3["Deaths"].tolist()

y2 = df3["Cases"].tolist()

labels = [str(value\*500000) for value in x]

plt.xticks(x, labels, fontsize=5)

plt.plot(y1, color="red", label="Deaths", linestyle="-.")

plt.plot(y2, color="blue", label="Cases")

plt.legend(loc="lower right")

plt.title("Deaths and Cases Over Vaccinations")

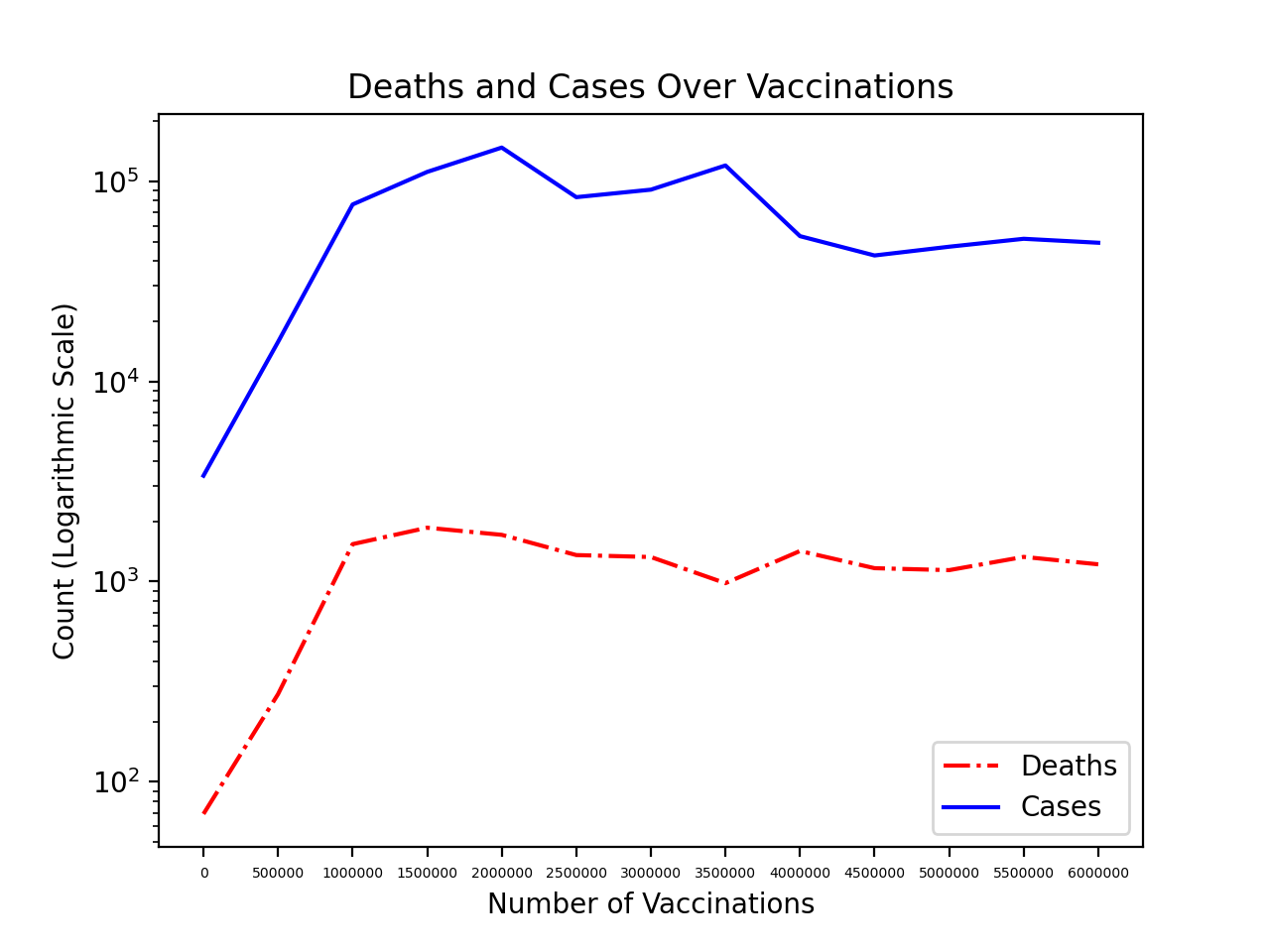
plt.xlabel("Number of Vaccinations")

plt.ylabel("Count (Logarithmic Scale)")

plt.yscale("log")

plt.show()

**Chart**



**Evaluation**

In this line chart the number of vaccinations is represented along the x-axis, the average number of deaths at each number of vaccinations is represented with the red dot-dash line according to the logarithmic scale on the y-axis, and the average number of cases at each number of vaccinations is represented by the blue solid line according to the logarithmic scale on the y-axis. A line graph was chosen to represent this data as it effectively shows the change in average deaths and cases as the vaccination numbers increase. A logarithmic scale for the y-axis was chosen because the number of deaths is always much smaller than the number of cases so a linear scale did not clearly show the trend for the average number of deaths. The difference in line style as well as colour was used to distinguish between the deaths and cases attributes so the graph could still be effectively viewed by people who are colourblind or if printed in black and white. This chart would still communicate the data effectively if more data was obtained as long as the appropriate changes were made. For example, if the number of vaccinations increased then the scale of the x-axis may need to be altered to increase at larger increments so that it could still be read clearly.

## 2.4 Section 4 - 500522378

### 2.4.1 Grouped Aggregate Summaries

#### 2.4.1.1 Average Number of Cases per Country

**Code**

import csv

first\_line = True

daily\_cases\_per\_country = {}

# daily\_cases\_per\_country is a dictionary.

# Key is the country name

# Value is a list of the cases for that country

average\_cases\_per\_country = {}

# average\_casess\_per\_country is a dictionary.

# Key is the country name

# Value is the average number of cases for that country

csv\_reader = csv.reader(open("integrated\_data.csv"))

for row in csv\_reader:

# Ignore header row.

if first\_line:

first\_line = False

else:

country = row[1]

daily\_cases = int(row[4])

if country not in daily\_cases\_per\_country:

daily\_cases\_per\_country[country] = [daily\_cases]

else:

daily\_cases\_per\_country[country].append(daily\_cases)

# Calculate average.

for key in daily\_cases\_per\_country:

average\_cases = round(

sum(daily\_cases\_per\_country[key]) / len(daily\_cases\_per\_country[key])

)

average\_cases\_per\_country[key] = average\_cases

total\_cases = 0

total\_num = 0

# Overall aggregate.

for key in average\_cases\_per\_country:

total\_cases += average\_cases\_per\_country[key]

total\_num += 1

average\_cases\_per\_country["Overall"] = round(total\_cases / total\_num)

print(average\_cases\_per\_country)

**Table of Results**

| **Country** | **Average Cases** |  | **Country** | **Average Cases** |
| --- | --- | --- | --- | --- |
| Argentina | 11938 |  | Malaysia | 1998 |
| Armenia | 550 |  | Maldives | 196 |
| Australia | 63 |  | Mexico | 6299 |
| Austria | 1726 |  | Morocco | 1425 |
| Bangladesh | 2085 |  | Mozambique | 205 |
| Belarus | 972 |  | Namibia | 233 |
| Belgium | 2818 |  | Nepal | 1712 |
| Bolivia | 1106 |  | Netherlands | 4499 |
| Bosnia and Herzegovina | 551 |  | New Zealand | 3 |
| Botswana | 192 |  | Nigeria | 392 |
| Bulgaria | 1145 |  | North Macedonia | 412 |
| Cabo Verde | 86 |  | Norway | 336 |
| Canada | 3599 |  | Pakistan | 2053 |
| Chile | 3503 |  | Panama | 1014 |
| Colombia | 11173 |  | Paraguay | 1142 |
| Costa Rica | 990 |  | Peru | 4867 |
| Cote d'Ivoire | 107 |  | Philippines | 3751 |
| Croatia | 981 |  | Poland | 7839 |
| Cyprus | 204 |  | Portugal | 2294 |
| Denmark | 771 |  | Qatar | 349 |
| Dominican Republic | 802 |  | Romania | 2904 |
| Ecuador | 1124 |  | Russia | 13292 |
| Estonia | 355 |  | Rwanda | 98 |
| Ethiopia | 742 |  | Saudi Arabia | 818 |
| Fiji | 10 |  | Senegal | 100 |
| Finland | 243 |  | Serbia | 1934 |
| France | 15115 |  | Singapore | 52 |
| Germany | 9706 |  | Slovakia | 1074 |
| Ghana | 215 |  | Slovenia | 704 |
| Greece | 1162 |  | South Africa | 4916 |
| Guatemala | 756 |  | South Korea | 393 |
| Hungary | 2209 |  | Spain | 9769 |
| Iceland | 13 |  | Sri Lanka | 698 |
| India | 81731 |  | Switzerland | 1840 |
| Indonesia | 5769 |  | Thailand | 680 |
| Iran | 8118 |  | Togo | 36 |
| Iraq | 3530 |  | Trinidad and Tobago | 88 |
| Ireland | 667 |  | Turkey | 14354 |
| Israel | 2243 |  | Uganda | 214 |
| Italy | 11068 |  | United Kingdom | 12257 |
| Jamaica | 136 |  | United States | 84683 |
| Latvia | 375 |  | Uruguay | 1003 |
| Lithuania | 763 |  | Zambia | 405 |
| Luxembourg | 185 |  | Zimbabwe | 126 |
| Madagascar | 110 |  | OVERALL | 38500 |

#### 2.4.1.2 Average Number of Deaths by Population Bin

**Code**

import csv

first\_line = True

# Dictionaries of populationBin -> value.

deaths\_by\_pop\_bin = {}

average\_deaths\_by\_pop\_bin = {}

csv\_reader = csv.reader(open("integrated\_data.csv"))

for row in csv\_reader:

# Ignore header row.

if first\_line:

first\_line = False

else:

deaths = int(row[5])

pop = int(row[3])

# Calculate the bin this value is in.

pop\_bin = pop // 10000000

if pop\_bin not in deaths\_by\_pop\_bin:

deaths\_by\_pop\_bin[pop\_bin] = [deaths]

else:

deaths\_by\_pop\_bin[pop\_bin].append(deaths)

# Calculate average.

for pop\_bin in deaths\_by\_pop\_bin:

average\_deaths = round(

(sum(deaths\_by\_pop\_bin[pop\_bin]) / len(deaths\_by\_pop\_bin[pop\_bin]))

)

average\_deaths\_by\_pop\_bin[pop\_bin] = average\_deaths

num = 0

for pop\_bin in average\_deaths\_by\_pop\_bin:

print(num, ":", average\_deaths\_by\_pop\_bin[pop\_bin])

num += 1

# Overall aggregate.

total\_deaths = 0

total\_pop\_bins = 0

for key in sorted(average\_deaths\_by\_pop\_bin):

total\_deaths += average\_deaths\_by\_pop\_bin[key]

total\_pop\_bins += 1

average\_deaths\_by\_pop\_bin["OVERALL"] = round(total\_deaths / total\_pop\_bins)

print(average\_deaths\_by\_pop\_bin)

**Table of Results**

| **Bin** | **Average deaths** |  | **Bin** | **Average deaths** |
| --- | --- | --- | --- | --- |
| 0 | 109 |  | 9 | 183 |
| 1 | 13 |  | 10 | 1046 |
| 2 | 8 |  | 11 | 152 |
| 3 | 34 |  | 12 | 567 |
| 4 | 32 |  | 13 | 4 |
| 5 | 92 |  | 14 | 49 |
| 6 | 147 |  | 15 | 64 |
| 7 | 12 |  | 16 | 343 |
| 8 | 181 |  | 17 | 1297 |
|  |  |  | Overall | 241 |

### 2.4.2 Charts

#### 2.4.2.1 Average Deaths by Population Bin

**Code**

import csv

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

first\_line = True

# Dictionaries of populatinBin -> value.

deaths\_by\_pop\_bin = {}

average\_deaths\_by\_pop\_bin = {}

csv\_reader = csv.reader(open("integrated\_data.csv"))

for row in csv\_reader:

# Ignore header row.

if first\_line:

first\_line = False

else:

pop = int(row[3])

deaths = int(row[5])

# Calculate the bin this value is in.

pop\_bin = pop // 10000000

if pop\_bin not in deaths\_by\_pop\_bin:

deaths\_by\_pop\_bin[pop\_bin] = [deaths]

else:

deaths\_by\_pop\_bin[pop\_bin].append(deaths)

# Calculate average.

avg\_deaths\_list = []

for pop\_bin in deaths\_by\_pop\_bin:

average\_deaths = round((sum(deaths\_by\_pop\_bin[pop\_bin]) / len(deaths\_by\_pop\_bin[pop\_bin])))

avg\_deaths\_list.append(average\_deaths)

average\_deaths\_by\_pop\_bin[pop\_bin] = average\_deaths

# Additional formatting.

data\_list\_1 = list(average\_deaths\_by\_pop\_bin.items())

df1 = pd.DataFrame(data\_list\_1)

combined\_data = [df1[1]]

headers = ["Average Deaths"]

df3 = pd.concat(combined\_data, axis = 1, keys = headers)

print(df3)

x = df3.index.tolist()

xLabels = [str(value\*10) + " Million" for value in x]

y = df3['Average Deaths'].tolist()

X\_axis = np.arange(len(x))

# Generate plot.

plt.bar(X\_axis, y, 0.4, label = 'Average Deaths')

plt.xticks(X\_axis, xLabels, rotation=90)

plt.xlabel("Population Bin")

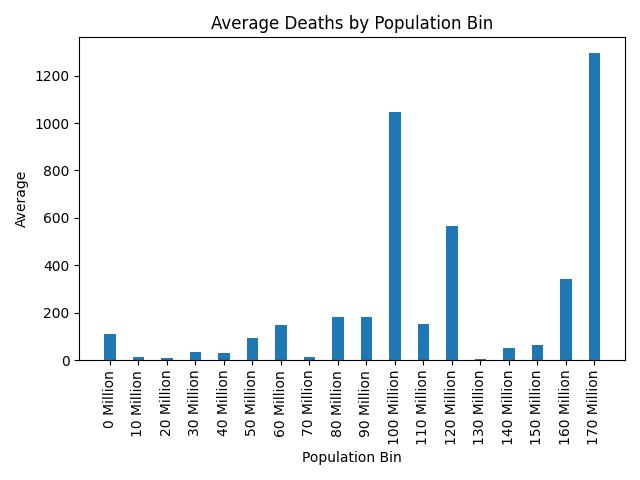
plt.ylabel("Average")

plt.title("Average Deaths by Population Bin")

plt.tight\_layout()

plt.show()

**Chart**

****

**Evaluation**

In this bar plot the population bin is represented along the x-axis, the average number of deaths is represented along the y-axis. A bar plot graph was chosen to showcase the relationship between the average number of deaths and a different range of population. According to the graph, it’s clearly indicating the average number of deaths is much higher in the high population bin than in the low population bin. However, due to the limitation of dataset that we gained, the lack of information of large population country for average number of deaths affects the consistency of the result, not all range of population bin could be shown in the graph. If more information is provided, the height of the bar would be longer and the width of each bar should be increase.

#### 2.4.2.2 Average Cases and Vaccinations by Population Bin

**Code**

import csv

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

first\_line = True

# Dictionaries of populatinBin -> value.

cases\_by\_pop\_bin = {}

average\_cases\_by\_pop\_bin = {}

vac\_by\_pop\_bin = {}

average\_vac\_by\_pop\_bin = {}

csv\_reader = csv.reader(open("integrated\_data.csv"))

for row in csv\_reader:

# Ignore header row.

if first\_line:

first\_line = False

else:

pop = int(row[3])

cases = int(row[4])

vac = float(row[7])

# Calculate the bin this value is in.

pop\_bin = pop // 10000000

if pop\_bin not in cases\_by\_pop\_bin:

cases\_by\_pop\_bin[pop\_bin] = [cases]

else:

cases\_by\_pop\_bin[pop\_bin].append(cases)

if pop\_bin not in vac\_by\_pop\_bin:

vac\_by\_pop\_bin[pop\_bin] = [vac]

else:

vac\_by\_pop\_bin[pop\_bin].append(vac)

# Calculate averages.

avg\_cases\_list = []

for pop\_bin in cases\_by\_pop\_bin:

average\_cases = round((sum(cases\_by\_pop\_bin[pop\_bin]) / len(cases\_by\_pop\_bin[pop\_bin])))

avg\_cases\_list.append(average\_cases)

average\_cases\_by\_pop\_bin[pop\_bin] = average\_cases

avg\_vac\_list = []

for pop\_bin in vac\_by\_pop\_bin:

average\_vac = round((sum(vac\_by\_pop\_bin[pop\_bin]) / len(vac\_by\_pop\_bin[pop\_bin])))

avg\_vac\_list.append(average\_vac)

average\_vac\_by\_pop\_bin[pop\_bin] = average\_vac

# Additional formatting.

data\_list\_1 = list(average\_cases\_by\_pop\_bin.items())

df1 = pd.DataFrame(data\_list\_1)

data\_list\_2 = list(average\_vac\_by\_pop\_bin.items())

df2 = pd.DataFrame(data\_list\_2)

combined\_data = [df1[1], df2[1]]

headers = ["Average Cases", "Average Vaccinations"]

df3 = pd.concat(combined\_data, axis = 1, keys = headers)

print(df3)

x = df3.index.tolist()

xLabels = [str(value\*10) + " Million" for value in x]

y1 = df3['Average Cases'].tolist()

y2 = df3['Average Vaccinations'].tolist()

X\_axis = np.arange(len(x))

# Generate plot.

plt.bar(X\_axis - 0.2, y1, 0.4, label = 'Average Cases')

plt.bar(X\_axis + 0.2, y2, 0.4, label = 'Average Vaccinations')

plt.xticks(X\_axis, xLabels, rotation=90)

plt.xlabel("Population Bin")

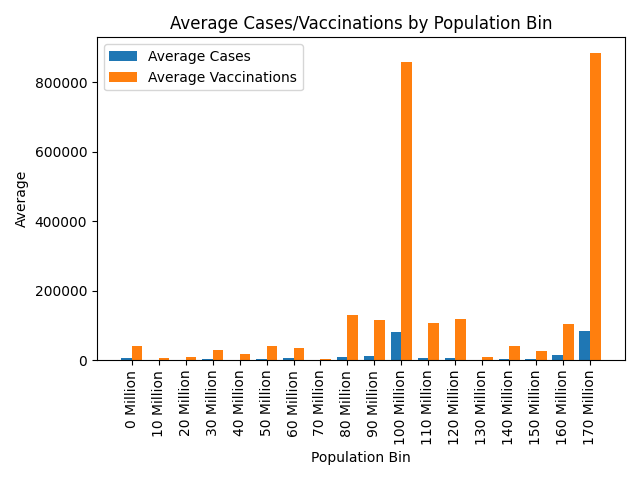
plt.ylabel("Average")

plt.title("Average Cases/Vaccinations by Population Bin")

plt.legend()

plt.show()

**Chart**

****

**Evaluation**

In this bar plot the population bin is represented along the x-axis, the average number of cases and vaccinations are represented along the y-axis. While the attribute of “Cases” is highlighted in blue bar and the “Vaccinations” one is highlighted in orange bar. A bar plot graph was chosen to showcase the relationship between the average number of cases and the number of average vaccinations with different range of population bin. The difference of the height of each bar in each population bin is easily to differentiate tell the difference for readers. However, due to the limitation of dataset that we gained, the lack of information of large population country for average number of deaths affects the consistency of the result, not all range of population bin could be shown specifically in the graph.

#### 2.4.2.3 Average Cases per Country (Bar Plot)

**Code**

import pandas as pd

import matplotlib.pyplot as plt

import csv

first\_line = True

# Dictionaries of country -> value.

daily\_cases\_per\_million\_country = {}

average\_cases\_per\_country = {}

csv\_reader = csv.reader(open("integrated\_data.csv"))

for row in csv\_reader:

# Ignore header row.

if first\_line:

first\_line = False

else:

country = row[1]

daily\_cases = int(row[4])

population = int(row[3])

cases\_per\_million = (daily\_cases / population) \* 1000000

if country not in daily\_cases\_per\_million\_country:

daily\_cases\_per\_million\_country[country] = [cases\_per\_million]

else:

daily\_cases\_per\_million\_country[country].append(cases\_per\_million)

# Calculate average.

for key in daily\_cases\_per\_million\_country:

average\_cases = round(

sum(daily\_cases\_per\_million\_country[key])

/ len(daily\_cases\_per\_million\_country[key])

)

average\_cases\_per\_country[key] = average\_cases

datals = list(average\_cases\_per\_country.items())

df = pd.DataFrame(datals)

country = df[0]

cases = df[1]

# Split the countries into separate graphs, so that the x axis isn't too crowded.

maxCountriesPerGraph = 15

offset = 0

while offset < len(country):

# Generate the plot.

plt.bar(country[offset:offset+maxCountriesPerGraph], cases[offset:offset+maxCountriesPerGraph])

plt.title("Average Cases per Country")

plt.xlabel("Country")

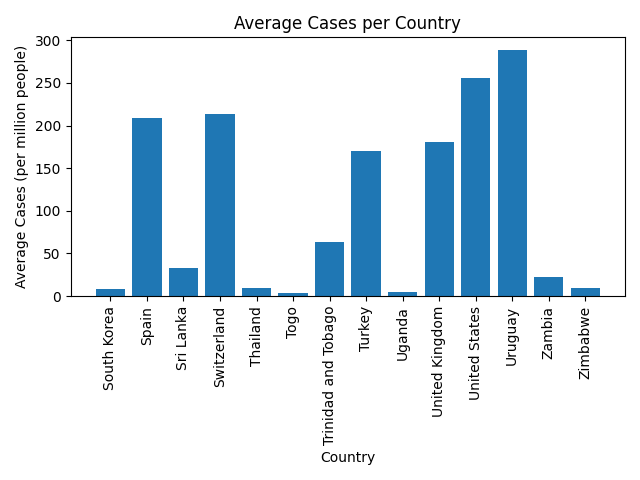
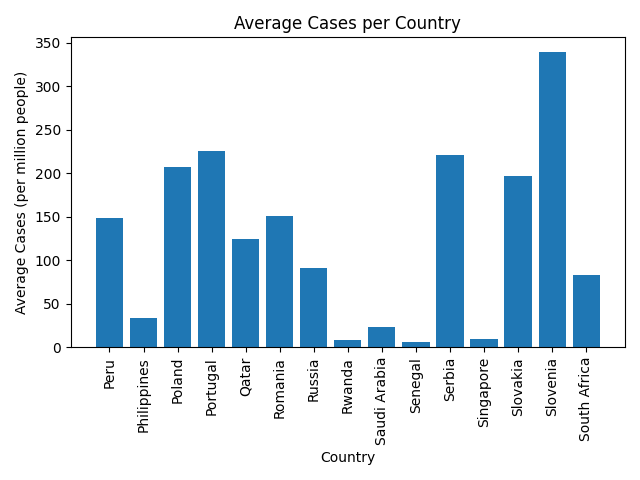
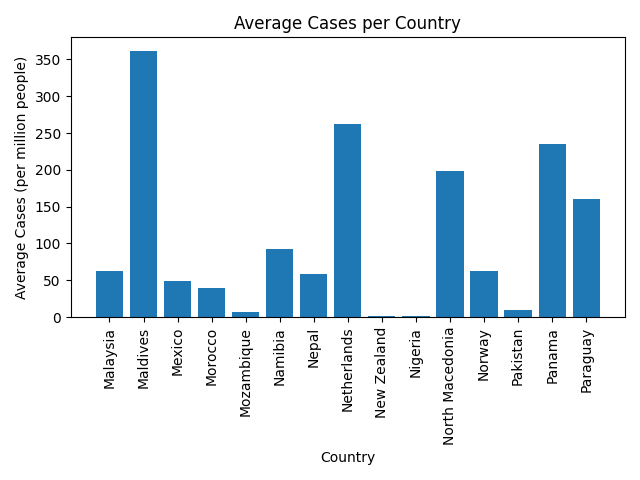
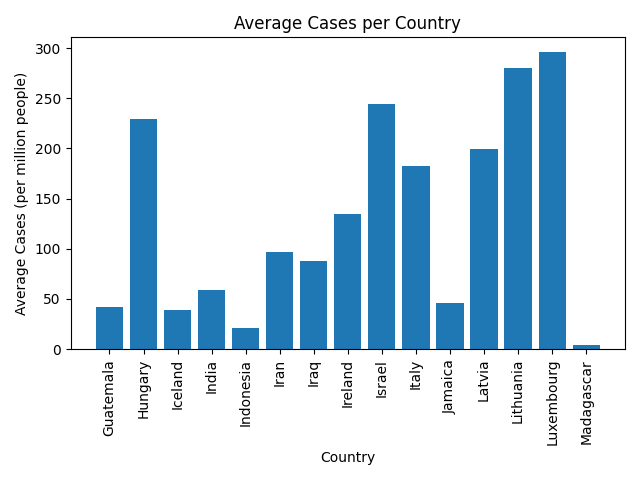
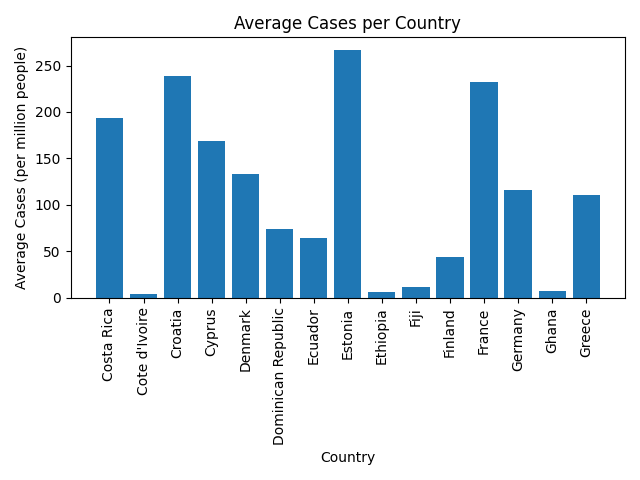
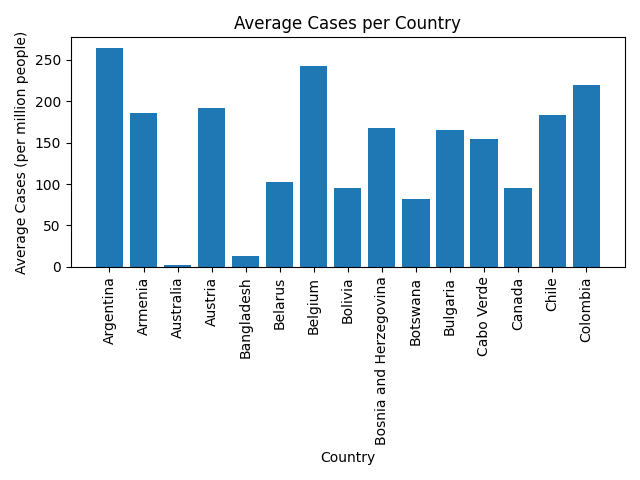
plt.ylabel("Average Cases (per million people)")

plt.xticks(rotation=90)

plt.show()

offset += maxCountriesPerGraph

**Chart**



**Evaluation**

In these bar plots the average number of cases per country is represented in the height of each bar according to the scale on the y-axis and each country is listed on the x-axis. A bar plot was chosen for this analysis because it allows for comparison between different countries, since the height of each bar could be easily distinguished. However, since too many attributes from x-axis are listed, it’s not very easy to look up each “country”’s value in this graph, if more attributes are added to the “country”, the graph have to change the width of each bar to make it more effective to read and analysis. Moreover, if more resources are obtained, the more reliable the graph would be.

# 3.0 Combined Chart

**Code**

import matplotlib.pyplot as plt

from matplotlib.lines import Line2D

import csv

def aggregatesOverPopulationPerGDPBin(filename, month, binRange=1000000000000):

firstLine = True

populationPerGDPBin = {}

vaccinationsPerGDPBin = {}

deathsPerGDPBin = {}

casesPerGDPBin = {}

countriesCounted = set()

csv\_reader = csv.reader(open("integrated\_data.csv"))

for row in csv\_reader:

if firstLine:

firstLine = False

else:

readMonth = row[0].split("-")[1]

if readMonth != month:

continue

country = row[1]

gdp = int(float(row[2]))

population = int(row[3])

cases = int(row[4])

deaths = int(row[5])

vaccinations = int(row[7])

gdpBin = gdp // binRange

if country not in countriesCounted:

countriesCounted.add(country)

if gdpBin not in populationPerGDPBin:

populationPerGDPBin[gdpBin] = 0

populationPerGDPBin[gdpBin] += population

if gdpBin not in vaccinationsPerGDPBin:

vaccinationsPerGDPBin[gdpBin] = 0

vaccinationsPerGDPBin[gdpBin] += vaccinations

if gdpBin not in casesPerGDPBin:

casesPerGDPBin[gdpBin] = 0

casesPerGDPBin[gdpBin] += cases

if gdpBin not in deathsPerGDPBin:

deathsPerGDPBin[gdpBin] = 0

deathsPerGDPBin[gdpBin] += deaths

return populationPerGDPBin, vaccinationsPerGDPBin, deathsPerGDPBin, casesPerGDPBin

# Define time range for chart.

monthsDict = {

"07": "July 2020",

"08": "August 2020",

"09": "September 2020",

"10": "October 2020",

"11": "November 2020",

"12": "December 2020",

"01": "January 2021",

"02": "February 2021",

"03": "March 2021",

"04": "April 2021",

"05": "May 2021",

"06": "June 2021"

}

months = ["07", "08", "09", "10", "11", "12", "01", "02", "03", "04", "05", "06"]

# Define GDP range/styles for chart.

gdpBins = [(0, "tab:red", "< $1 Billion USD"), (1, "tab:orange", "$2 Billion USD"), (2, "tab:olive", "$3 Billion USD"), (3, "tab:cyan", "$4 Billion USD"), (20, "tab:blue", "$5 Billion USD +")]

# Additional formatting for x axis.

xTicks = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

xLabels = []

for month in months:

xLabels.append(monthsDict[month])

# Generate/format data.

for gdpBin in gdpBins:

formattedVaccinations = []

formattedDeaths = []

for month in months:

populationPerGDPBin, vaccinationsPerGDPBin, deathsPerGDPBin, casesPerGDPBin = aggregatesOverPopulationPerGDPBin("integrated\_data.csv", month)

formattedVaccinations.append(vaccinationsPerGDPBin[gdpBin[0]] / populationPerGDPBin[gdpBin[0]])

formattedDeaths.append(deathsPerGDPBin[gdpBin[0]] / populationPerGDPBin[gdpBin[0]])

plt.plot(xTicks, formattedVaccinations, label=gdpBin[2], color=gdpBin[1])

plt.plot(xTicks, formattedDeaths, color=gdpBin[1], linestyle='dashed')

# Additional formatting for legend.

legendLines = []

legendLabels = []

for gdpBin in gdpBins:

legendLines.append(Line2D([0], [0], color=gdpBin[1]))

legendLabels.append(gdpBin[2])

legendLabels.append("Vaccinations")

legendLines.append(Line2D([0], [0], color='k'))

legendLabels.append("Deaths")

legendLines.append(Line2D([0], [0], color='k', linestyle="dashed"))

plt.legend(legendLines, legendLabels)

plt.xticks(xTicks, xLabels, rotation=90)

plt.xlabel("Time")

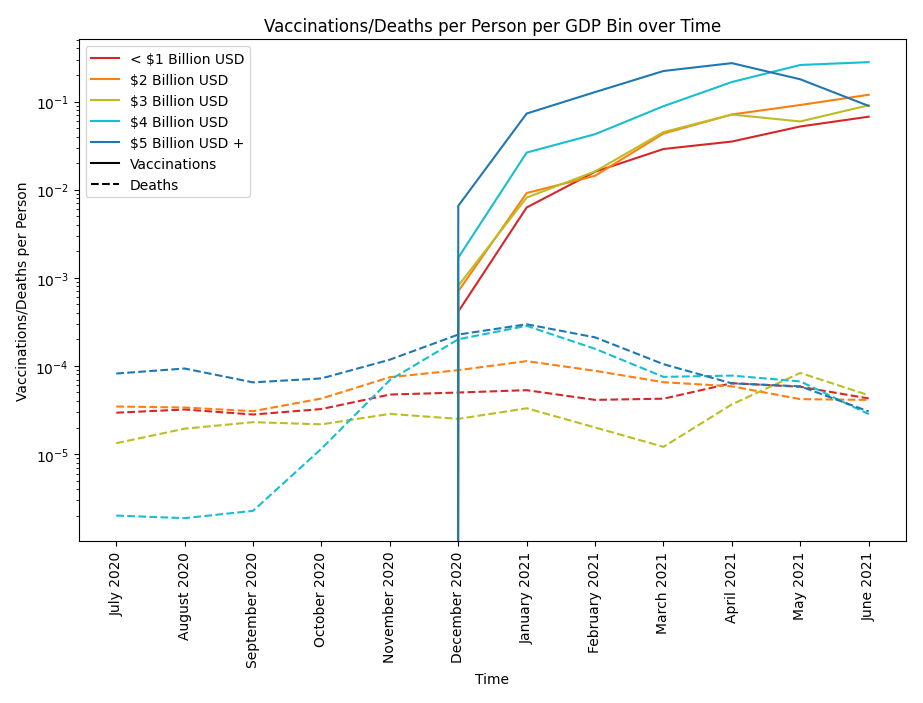
plt.ylabel("Vaccinations/Deaths per Person")

plt.yscale('log')

plt.title("Vaccinations/Deaths per Person per GDP Bin over Time")

plt.show()

**Chart**



**Evaluation**

The chart above is a line chart that shows both vaccination and deaths per person per GDP bin from July 2020 to June 2021. In this line chart, the vaccination and death rate is encoded in the y-axis and the time frame is encoded in the x-axis. The GDP bin is separated into five ranges while the two attributes (vaccination and deaths) have 2 different line styles - each bin/range is assigned one color. The legend is placed in the upper left corner of the plot with information regarding both the GDP ranges and the attributes. For this combined chart, a line chart is most appropriate to measure the change in vaccinations and death overtime as it effectively tracks both short term or long term changes better than other available plots. Additionally, this chart also communicates effectively because of the log scale used in the y-axis. The log scale in the y-axis was used to clearly show the trend of both variables as the vaccination rate is much higher than the death rate. Without making use of the log scale, the line of the death rate is too close to zero, thus, the log scale was used to make the chart more readable and understandable. Unlike most plots in the previous section of the paper, if more data was added, there might only be minor changes and problems regarding the effectiveness of the graph will not arise. For instance, if more data was added from June 2021 onwards, there will only be a change in the x-axis that indicates the time frame, making the horizontal axis longer than it is now. If the values of each attribute were higher or lower, there may only be a slight change in the trend lines.

Part B

# 4.0 Conclusion

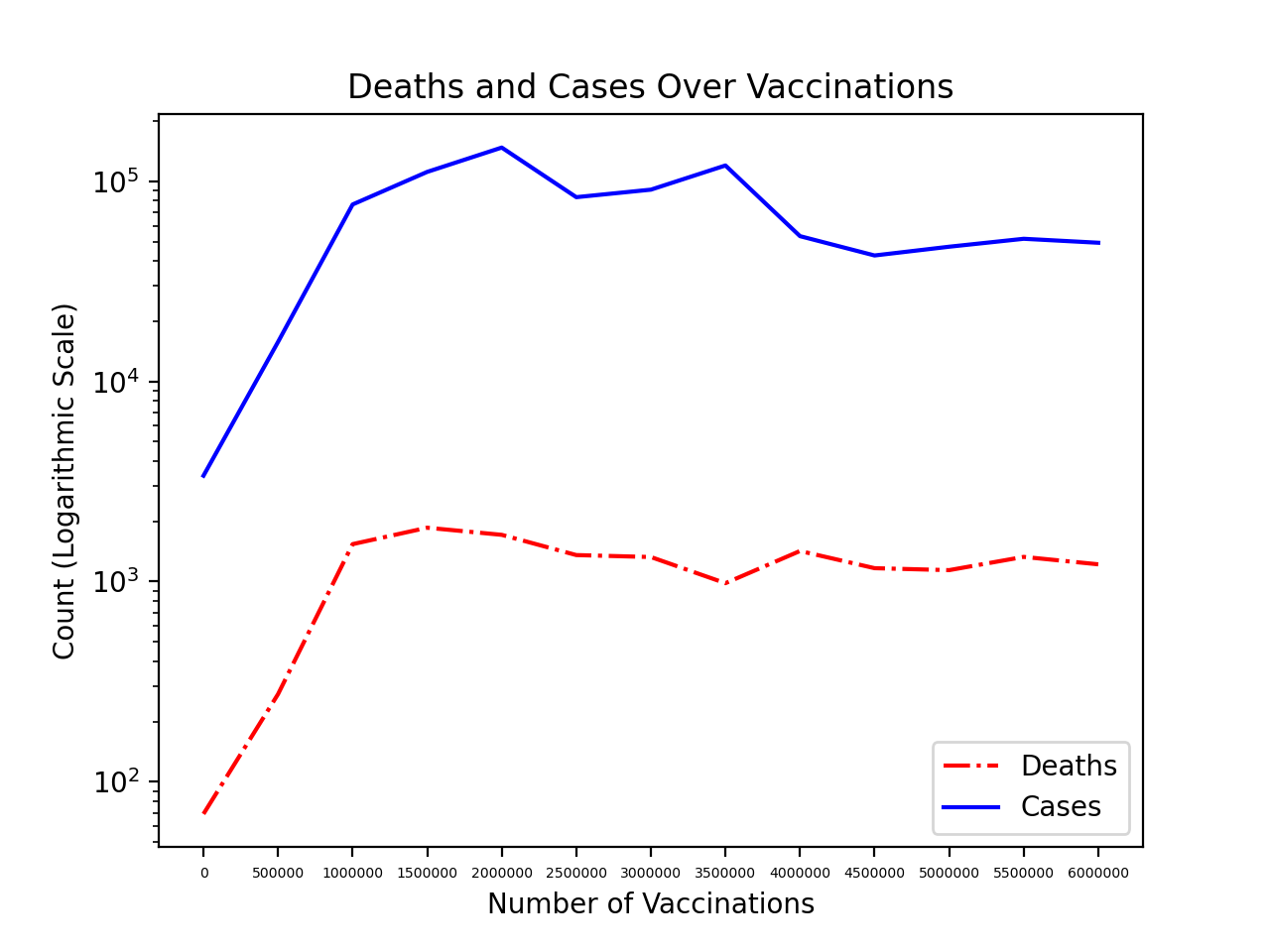
The COVID-19 pandemic has been an unprecedented period of new challenges and hardships. Governments around the world have had to find the best ways to keep the spread and impact of the virus as small as possible. This report has taken steps to determine how effective the responses to the pandemic have been in accomplishing this goal.

One response to the rise of COVID-19 has been the increased rates of testing for the virus. This has allowed for the quick treatment and isolation of people with COVID-19. From our analysis, we saw that this correlates with a decreased number of deaths:

| **Test Bin (by 100000s)** | **Average Number of Deaths** |  | **Test Bin (by 100000s)** | **Average Number of Deaths** |
| --- | --- | --- | --- | --- |
| 0 | 48 |  | 16 | 1422 |
| 1 | 126 |  | 17 | 2293 |
| 2 | 215 |  | 18 | 2769 |
| 3 | 314 |  | 19 | 2652 |
| 4 | 332 |  | 20 | 3165 |
| 5 | 467 |  | 21 | 2523 |
| 6 | 597 |  | 22 | 2450 |
| 7 | 392 |  | 23 | 2129 |
| 8 | 597 |  | 24 | 1258 |
| 9 | 673 |  | 29 | 58 |
| 10 | 819 |  | 32 | 3207 |
| 11 | 737 |  | 35 | 2677 |
| 12 | 1071 |  | 36 | 2713 |
| 13 | 984 |  | 37 | 3134 |
| 14 | 1309 |  | OVERALL | 1419 |
| 15 | 1425 |  |  |  |

In the above table, the number of deaths do rise over the first few test bins. This can be attributed to the rise in cases over the start of the pandemic and the delay in the increase in testing rates. However, at bins 20-21 (around 210000 tests per day), the number of deaths starts to decrease, showing that testing is an effective measure at reducing the number of deaths caused by COVID-19.

While the importance of COVID-19 testing was emphasised in 2020, this year we saw the focus shift to vaccinations as we look towards a way to safely live with the virus still in the community. This report has also found that vaccinations correlate with reduced cases and deaths, suggesting that they are an effective measure of mitigating COVID-19’s effects.



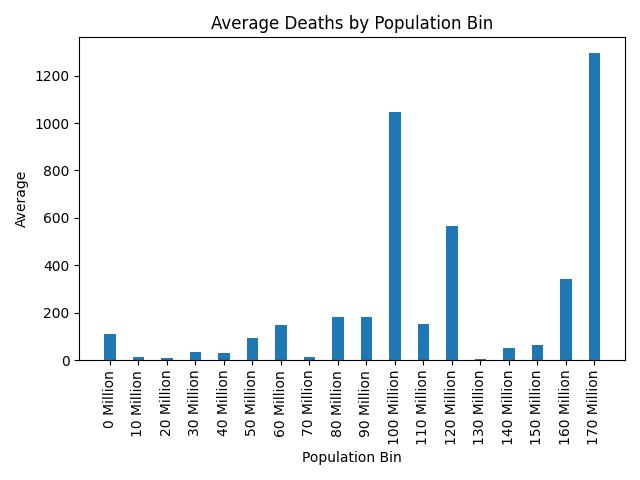
As with the tests, there is an initial increase in both cases and deaths as the vaccination rates increase due to the delay in producing and rolling out vaccines. However, at around 1 million vaccinations a day, the rate of cases and deaths begins to stabilise. While there is not much of a decrease in either, the stabilisation shows that vaccines are helping keep the impact of the COVID-19 virus at manageable levels, even as many countries begin to reopen and return to normal life.

However, some countries have had the odds stacked against them with factors that have made responding to the threat of the virus harder than others. This report has also investigated the factors that have limited a country's response to the COVID-19 pandemic.

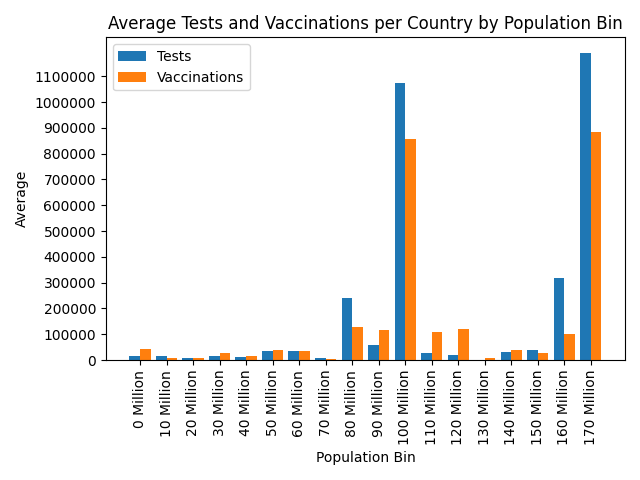
For example, countries with higher populations have demonstrated higher case numbers than countries of lower populations, as shown by our investigation:

| **Country** | **Average Cases** |  | **Country** | **Average Cases** |
| --- | --- | --- | --- | --- |
| Argentina | 11938 |  | Malaysia | 1998 |
| Armenia | 550 |  | Maldives | 196 |
| Australia | 63 |  | Mexico | 6299 |
| Austria | 1726 |  | Morocco | 1425 |
| Bangladesh | 2085 |  | Mozambique | 205 |
| Belarus | 972 |  | Namibia | 233 |
| Belgium | 2818 |  | Nepal | 1712 |
| Bolivia | 1106 |  | Netherlands | 4499 |
| Bosnia and Herzegovina | 551 |  | New Zealand | 3 |
| Botswana | 192 |  | Nigeria | 392 |
| Bulgaria | 1145 |  | North Macedonia | 412 |
| Cabo Verde | 86 |  | Norway | 336 |
| Canada | 3599 |  | Pakistan | 2053 |
| Chile | 3503 |  | Panama | 1014 |
| Colombia | 11173 |  | Paraguay | 1142 |
| Costa Rica | 990 |  | Peru | 4867 |
| Cote d'Ivoire | 107 |  | Philippines | 3751 |
| Croatia | 981 |  | Poland | 7839 |
| Cyprus | 204 |  | Portugal | 2294 |
| Denmark | 771 |  | Qatar | 349 |
| Dominican Republic | 802 |  | Romania | 2904 |
| Ecuador | 1124 |  | Russia | 13292 |
| Estonia | 355 |  | Rwanda | 98 |
| Ethiopia | 742 |  | Saudi Arabia | 818 |
| Fiji | 10 |  | Senegal | 100 |
| Finland | 243 |  | Serbia | 1934 |
| France | 15115 |  | Singapore | 52 |
| Germany | 9706 |  | Slovakia | 1074 |
| Ghana | 215 |  | Slovenia | 704 |
| Greece | 1162 |  | South Africa | 4916 |
| Guatemala | 756 |  | South Korea | 393 |
| Hungary | 2209 |  | Spain | 9769 |
| Iceland | 13 |  | Sri Lanka | 698 |
| India | 81731 |  | Switzerland | 1840 |
| Indonesia | 5769 |  | Thailand | 680 |
| Iran | 8118 |  | Togo | 36 |
| Iraq | 3530 |  | Trinidad and Tobago | 88 |
| Ireland | 667 |  | Turkey | 14354 |
| Israel | 2243 |  | Uganda | 214 |
| Italy | 11068 |  | United Kingdom | 12257 |
| Jamaica | 136 |  | United States | 84683 |
| Latvia | 375 |  | Uruguay | 1003 |
| Lithuania | 763 |  | Zambia | 405 |
| Luxembourg | 185 |  | Zimbabwe | 126 |
| Madagascar | 110 |  | OVERALL | 38500 |

In the table above, the countries with higher populations have a higher average number of COVID-19 cases recorded. This makes sense as in a larger population it is easier for the virus to spread. However, due to variations in how countries report their new cases the final findings cannot be as thorough as expected. These countries also experience more deaths from COVID-19 as shown by the graph below.

****

Even so, the graph and table below suggest that countries with higher populations have also had higher rates of vaccinations and testing, which, as previously shown, are effective in reducing the spread and deaths of the virus. This causes some uncertainty in our results so more investigation into this would be beneficial in the future.

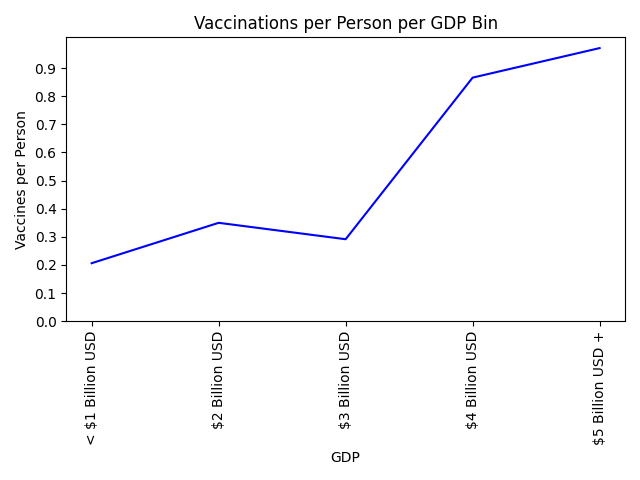


| **Bin** | **Average Tests** |  | **Bin** | **Average Tests** |
| --- | --- | --- | --- | --- |
| 0 | 17663 |  | 10 | 1073862 |
| 1 | 15736 |  | 11 | 26973 |
| 2 | 9790 |  | 12 | 17931 |
| 3 | 15938 |  | 13 | 1694 |
| 4 | 12551 |  | 14 | 32870 |
| 5 | 33392 |  | 15 | 37208 |
| 6 | 35173 |  | 16 | 317768 |
| 7 | 7164 |  | 17 | 1191183 |
| 8 | 241627 |  | OVERALL | 174730 |
| 9 | 56610 |  |  |  |

Countries with lower populations also often have lower GDP and economic resources which can lead to less access to healthcare and sanitation. This in turn makes it more likely for people to die from COVID-19 in these countries. However, similar to the case data, variations in the reporting of deaths between countries have certainly affected our final findings.

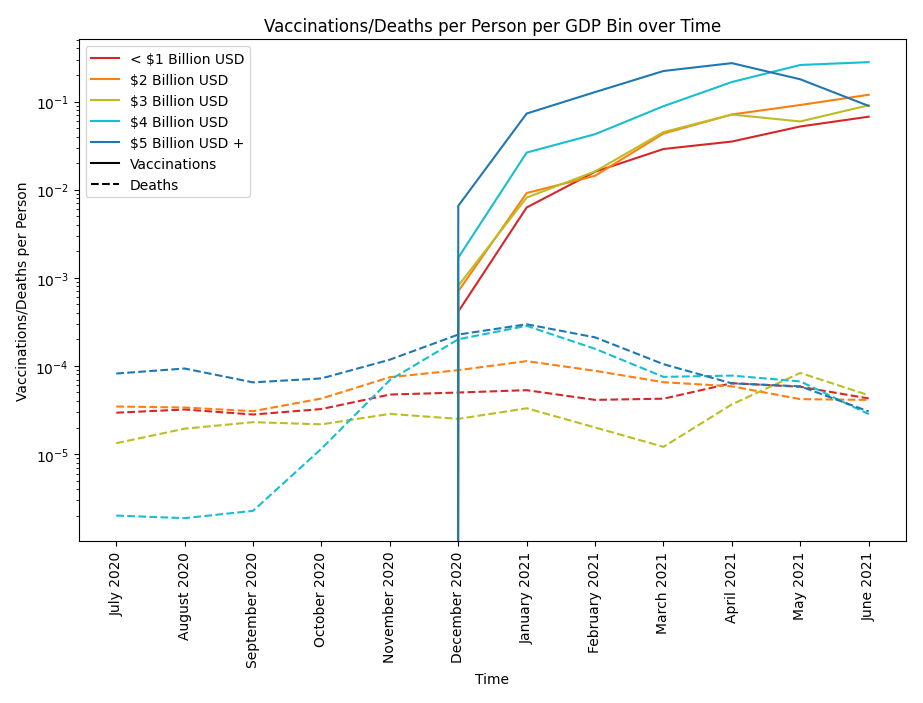
Gross Domestic Product (GDP) also plays an important role in a country’s ability to respond to the COVID-19 pandemic, as shown by the below table and graph:

| **GDP Range** | **Vaccinations / Population** | **Vaccines Per Person** |
| --- | --- | --- |
| < $1 Billion USD | 414789728 / 2010809414 | 0.20 |
| ≥ $1 Billion USD, < $2 Billion USD | 269386730 / 770119675 | 0.34 |
| ≥ $2 Billion USD, < $3 Billion USD | 441088147 / 1513172710 | 0.29 |
| ≥ $3 Billion USD, < $4 Billion USD | 72552356 / 83784978 | 0.86 |
| ≥ $4 Billion USD | 321396535 / 331007570 | 0.97 |
| OVERALL | 1519213496 / 4708894347 | 0.32 |



These clearly show that countries with higher GDPs have a much higher ability to vaccinate their population. This is intuitive, since countries with higher GDPs are able to purchase more vaccine doses and their population often has better access to the health facilities needed to get vaccinated.

However, higher GDP does not necessarily correlate with reduced deaths, as seen by the following graph:



Before the vaccination rollout began in December 2020, there does not appear to be a trend in how GDP impacts the number of deaths per person. In fact, the highest GDP group has the highest number of deaths per person. While we cannot give a definitive reason for this, it may be due to the fact that countries with higher GDPs often have more densely populated areas where the virus can spread more quickly, resulting in more cases of COVID-19 and therefore more deaths. While our dataset did contain data on the population of each country, it did not include data on the population density of each country, which limits our analysis. However, this would be an interesting direction for future study into this topic.

A clear relationship exists between GDP and the rate of the vaccination rollout. The higher GDP groups have a more rapid increase in vaccinations per person than the lower GDP groups. This results in a larger decrease in deaths after December 2020 in the higher GDP groups than in the lower GDP groups.

In conclusion, this report finds that higher testing and vaccination rates result in lower numbers of cases and deaths; countries with higher populations experience more cases, however, due to higher testing and vaccination rates, the actual impact on these countries is comparable to those with lower populations; and that countries with higher GDP have higher vaccination rates than those with lower GDPs, meaning that after the the vaccination rollout began in December 2020 they also had less deaths. Even with these findings, there are still uncertainties into the factors that cause these results. Further research into how factors such as population density and access to healthcare and sanitation relate to the factors discussed in the report would be highly valuable in advancing our knowledge of the COVID-19 pandemic.