CS-181: Final Project: Evaluation of Urban Population Percentage of Covid Deaths Across Continents In The World
Name: Viet Nguyen
April 20, 2024

Introduction

This investigation delves into how urbanization impacts the proliferation and fatality rates of COVID-19 across diverse societies and continents. Its objective is to ascertain whether the dense population in major cities like New York and Chicago has hastened the virus's spread and if differences in healthcare availability in urban areas have influenced pandemic management. Moreover, the study probes into whether a nation's proximity to China, the initial epicenter of COVID-19, has influenced the virus's transmission. Drawing on global data from the early 21st century, the research identifies patterns and connections in COVID-19's impact, considering urbanization levels, healthcare accessibility, and geographical proximity.

Ethical Consideration

This research highlights the ethical necessity of taking into account the environmental consequences of industrialization and economic advancement, particularly in relation to issues such as global warming and pollution. The differences in emissions observed among nations at various developmental phases emphasize the obligation of more developed countries to take the lead in minimizing their environmental impact and promoting sustainable growth in less developed nations. This entails ethical responsibilities to address the effects of climate change and to ensure that all countries have the opportunity to pursue development in a manner that is both environmentally sustainable and fair.

Methods

Our research methodology involved a detailed examination of three datasets, which provided valuable insights into the subject at hand. We utilized COVID-19 death statistics from "Our World in Data," urban population metrics from the World Bank's "World Development Indicators" (WDI), and overall population data from a comprehensive dataset on Kaggle. The main variables studied included "date," "country," "urban_percentage_2022," "population_2022," as well as COVID-19 indicators like "new_cases_2022," "new_deaths_2022," and "new_vaccinations_2022." To ensure data accuracy, a thorough cleaning process was conducted to address inconsistencies and prepare for analysis.

Data Cleaning and Visualization

The methodology employed for data cleaning and visualization in this project adopts a thorough strategy for extracting, organizing, and scrutinizing data pertaining to COVID-19 statistics, urban

population metrics, and total population data. The aim is to enable a detailed exploration of how urbanization influences COVID-19 occurrences and fatalities worldwide. This involves employing a blend of data-cleaning methods, integrating databases, and utilizing visualization software to provide comprehensive insights.

Data Cleaning

1. Reading and Converting Data:
   - The COVID-19 data is read from a JSON file using `json.load`, then transformed into a pandas DataFrame. This step requires iterating through the nested JSON structure to extract the relevant information, setting a multi-level index with country code and date.
   - Urban population data and overall population data are read from CSV files using `pandas.read_csv`. This ensures the data is loaded into DataFrames for further processing.

Code:

```
def read_json(infilename):
    """ Borrow from Dr. Buell and modified
        Read the owid-covid-data.json using the Python json package
        into Dataframe
        Parameters:
            infilename: the JSON file to read
        Returns:
            thedata: the dataframe
    """
    with open(infilename, encoding='utf-8') as thefile:
        covid = json.load(thefile)
    listofdicts = []
    for code, codesubtree in covid.items():
        for value in codesubtree:
            if value == 'data':
                list_dict = codesubtree['data']
                for items in list_dict:
                    items['code'] = code
                    listofdicts.append(items)
    dataframe = pd.DataFrame(listofdicts)
    dataframe.set_index(['code', 'date'], inplace=True)
    return dataframe
```

2. Cleaning and Reformatting:

- “Urban Population Data”: The data is cleaned by renaming columns to create a consistent and descriptive naming scheme. In cases where numeric data has inconsistencies (such as multiple decimal points), these are corrected by replacing them with `NaN`. "Country Code" is the index for this dataset.
- “Population Data”: The process involves renaming columns and eliminating unnecessary data. To standardize the structure, "CCA3" (country code) sets a new index. Redundant columns are removed to focus on the most relevant information.
- “COVID-19 Data”: The data undergoes a detailed transformation, where the "date" field is split into year, month, and day. This step allows for filtering by year to focus on 2022. The data is then aggregated by country code, summing the new cases, new deaths, and new vaccinations for the entire year.

Code:

```
def clean_urban_df(input_file):
    """
    Clean and reformat urban population data from a DataFrame.
    This function processes DataFrame columns, renaming them, handling
    incorrect data types, and setting appropriate indexes.
    Parameters:
        input (pandas.DataFrame): The DataFrame containing urban data.
    Returns:
        pandas.DataFrame: A cleaned DataFrame with urban population percentage.
    """
    clean_df = input_file[["Country Name", "Country Code", "2022"]]
    clean_df = clean_df.rename(columns={"Country Name": "Country",
                                        "Country Code": "Code",
                                        "2022": "Urban_Percentage_2022"})
    for element in clean_df["Urban_Percentage_2022"]:
        if str(element).count('.') > 1:
            clean_df["Urban_Percentage_2022"].replace(element, np.nan, inplace = True)
        else:
            clean_df["Urban_Percentage_2022"].replace(element, float(element), inplace
= True)
    clean_df = clean_df.dropna()
    clean_df.set_index(['Code'], inplace = True)
    return clean_df

#123456789 123456789 123456789 123456789 123456789 123456789 123456789
######################################################################
```

```
def clean_poppulation_df(input_file):
    """
    Clean and reformat population data from a DataFrame.
    This function processes DataFrame columns, renaming them and
    setting appropriate indexes.
    Parameters:
        input (pandas.DataFrame): The DataFrame containing population data.
    Returns:
        pandas.DataFrame: A cleaned DataFrame with relevant population data.
    """
    clean_df =  input_file[["CCA3", "Country/Territory", "Continent", "2022
Population"]]
    clean_df = clean_df.rename(columns={"CCA3": "Code",
                                        "Country/Territory": "Country",
                                        "2022 Population": "Population_2022"})
    clean_df.set_index(['Code'], inplace = True)
    return clean_df

#123456789 123456789 123456789 123456789 123456789 123456789 123456789
######################################################################

def clean_death_df(input_file):
    """
    Clean and reformat COVID-19 related data from a DataFrame.
    This function extracts year, month, and day from dates, and
    aggregates data for the year 2022.
    Parameters:
        input (pandas.DataFrame): The DataFrame containing raw COVID-19
        data.
    Returns:
        pandas.DataFrame: A DataFrame with aggregated COVID-19 data for 2022.
    """
    clean_df = input_file.reset_index()
    date = clean_df['date'].str.split(expand = True, pat = "-", n = 2)
    clean_df.insert(loc = 2, column = 'Year', value = date[0])
    clean_df.insert(loc = 3, column = 'Month', value = date[1])
    clean_df.insert(loc = 4, column = 'Day', value = date[2])
    clean_df = clean_df[['code', 'Year', 'Month', 'Day', 'new_cases',
                         'new_deaths', 'new_vaccinations']]
    clean_df = clean_df[clean_df['Year'] == '2022']
    clean_df = clean_df.groupby(['code']).mean(numeric_only=True)
    clean_df = clean_df.reset_index()
```

```
    clean_df = clean_df.rename(columns={"code": "Code",
                                        "new_cases": "New_cases_2022",
                                        "new_deaths": "Total_new_deaths_2022",
                                        "new_vaccinations": "New_vaccinations_2022"})
    clean_df.set_index(['Code'], inplace = True)
    return clean_df
```

3. Database Integration:
- After cleaning, the DataFrames are integrated into an SQLite database. Separate tables are created for urban population, overall population, and COVID-19 data, each with a defined structure and primary key.
- The data is merged into a unified table using SQL queries, joining the separate tables based on the common key "Code." This unified table serves as a basis for further analysis and visualization.

Code:

```
def urban_create(dataframe, dbfilename):
    """
    Create and populate an SQLite table for urban data.
    This function creates (if not exists) an SQLite table and populates
    it with urban data.
    Parameters:
        dataframe (pandas.DataFrame): The DataFrame to insert into the
        SQLite database.
        dbfilename (str): Path to the SQLite database file.
    Returns:
        none: create table in database
    """
    con = sqlite3.connect(dbfilename)
    cur = con.cursor()
    creation =  """ CREATE TABLE IF NOT EXISTS urban_table (
                Code VARCHAR(3) NOT NULL PRIMARY KEY,
                Country VARCHAR NULL,
                Urban_Percentage_2022 FLOAT NULL)"""
    cur.execute(creation)
    dataframe.to_sql("urban_table", con, if_exists = "replace", index = True)
    con.commit()

#123456789 123456789 123456789 123456789 123456789 123456789 123456789
##################################################################
```

```
def population_create(dataframe, dbfilename):
    """
    Create and populate an SQLite table for population data.
    This function creates (if not exists) an SQLite table and
    populates it with population data.
    Parameters:
        dataframe (pandas.DataFrame): The DataFrame to insert into the
        SQLite database.
        dbfilename (str): Path to the SQLite database file.
    Returns:
        none: create table in database
    """
    con = sqlite3.connect(dbfilename)
    cur = con.cursor()
    creation =  """ CREATE TABLE IF NOT EXISTS population_table (
                Code VARCHAR(3) NOT NULL PRIMARY KEY,
                Country VARCHAR NULL,
                Continent VARCHAR NULL,
                Population_2022 FLOAT NULL)"""
    cur.execute(creation)
    dataframe.to_sql("population_table", con, if_exists = "replace", index = True)
    con.commit()

#123456789 123456789 123456789 123456789 123456789 123456789 123456789
######################################################################

def death_create(dataframe, dbfilename):
    """
    Create and populate an SQLite table for COVID-19 death data.
    This function creates (if not exists) an SQLite table and populates
    it with COVID-19 data.
    Parameters:
        dataframe (pandas.DataFrame): The DataFrame to insert into the
        SQLite database.
        dbfilename (str): Path to the SQLite database file.
    Returns:
        none: create table in database
    """
    con = sqlite3.connect(dbfilename)
    cur = con.cursor()
    creation =  """ CREATE TABLE IF NOT EXISTS death_table (
```

```
                Code VARCHAR(3) NOT NULL PRIMARY KEY,
                New_cases_2022 FLOAT NULL,
                Total_new_death_2022 FLOAT NULL,
                New_vaccinations_20222 FLOAT NULL)"""
    cur.execute(creation)
    dataframe.to_sql("death_table", con, if_exists = "replace", index = True)
    con.commit()

#123456789 123456789 123456789 123456789 123456789 123456789 123456789
######################################################################

def merged_table(dbfilename, continent):
    """
    Create a merged table from existing tables and retrieve data for
    a specific continent.
    This function merges death, population, and urban data into a single
    table
    and filters the data based on the continent.
    Parameters:
        dbfilename (str): Path to the SQLite database file.
        continent (str): The continent name to filter the data.
    Returns:
        pandas.DataFrame: A DataFrame containing the merged data for the
        specified continent.
    """
    con = sqlite3.connect(dbfilename)
    cur = con.cursor()
    merge = """CREATE TABLE IF NOT EXISTS merged_table AS
            SELECT death_table.*, population_table.Population_2022,
population_table.Continent,
                    urban_table.Urban_Percentage_2022
            FROM death_table
            INNER JOIN population_table ON death_table.Code = population_table.Code
            INNER JOIN urban_table ON death_table.Code = urban_table.Code
        """
    cur.execute(merge)
    select_continent = """SELECT * FROM merged_table WHERE Continent = ?"""
    continent_df = pd.read_sql_query(select_continent, con, params=(continent,))
    con.commit()
    return continent_df
```

Data Visualization

Visualization plays a crucial role in deriving insights from the cleaned data. The key visualization technique used in this project is the scatter plot, which is employed to explore relationships between urbanization and COVID-19 deaths across various continents.

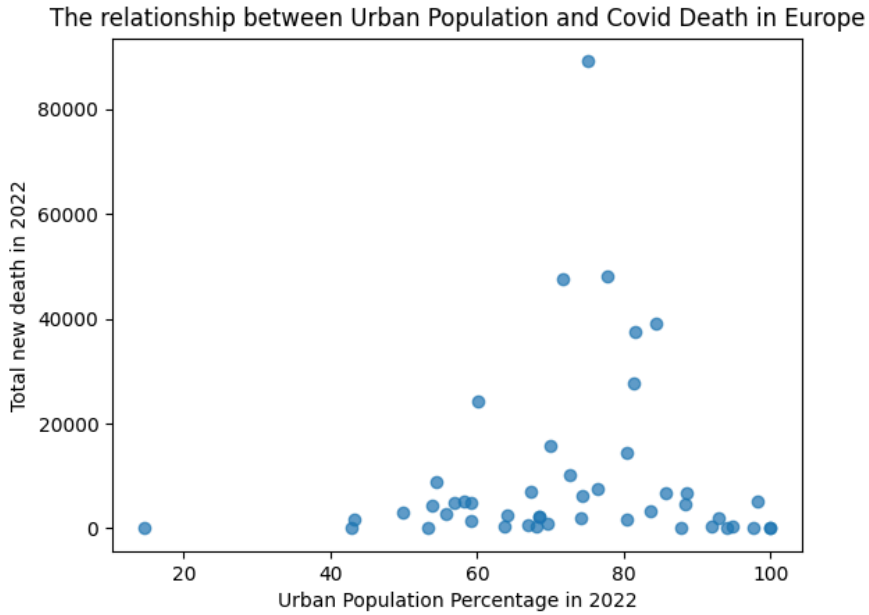
1. Scatter Plots for Different Continents:
   - Scatter plots are generated to examine the relationship between the percentage of urban population in 2022 and the total new COVID-19 deaths in 2022. Each scatter plot represents a different continent (e.g., Europe, Asia, Oceania, South America, and North America), allowing for a regional comparison of trends.
   - The x-axis represents "Urban_Percentage_2022," while the y-axis represents "Total_new_deaths_2022." The scatter plots include appropriate labels, titles, and legends to improve clarity and ease of interpretation.

2. Customization and Presentation
   - The scatter plots are customized with various stylistic elements to enhance visualization, including different colors and markers for each continent. Titles are dynamically generated based on the continent's name.
   - The scatter plots are then presented for analysis, providing a clear visual representation of the data to facilitate interpretation and discussion.

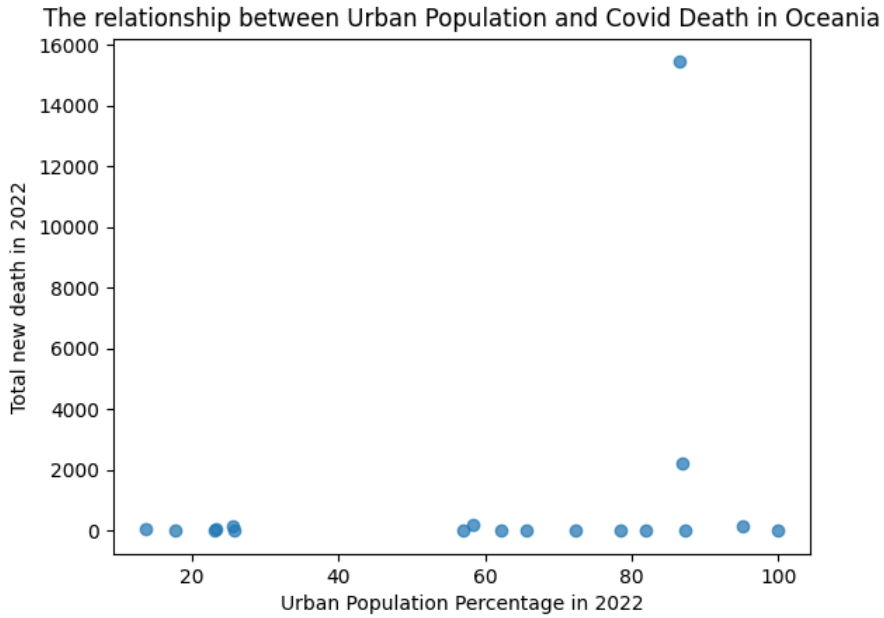
```
def make_scatterplot(dataframe, continent):
    """
    Generate a scatter plot showing the relationship between urban
    population percentage and COVID-19 deaths. This function plots
    data points representing the urban population percentage and the
    total new deaths from COVID-19 in 2022 for a specific continent.
    Parameters:
        dataframe (pandas.DataFrame): The DataFrame containing data
        to plot.
        continent (str): The continent name used for the plot title.
    Returns:
        none: show plot
    """
    x_axis = dataframe[
        'Urban_Percentage_2022']
    y_axis = dataframe['Total_new_deaths_2022']
    plt.scatter(x_axis,y_axis,alpha=0.7)
    plt.xlabel('Urban Population Percentage in 2022')
    plt.ylabel('Total new death in 2022')
    plt.title(f'The relationship between Urban Population and Covid Death in
{continent}')
```

```
plt.show()
```

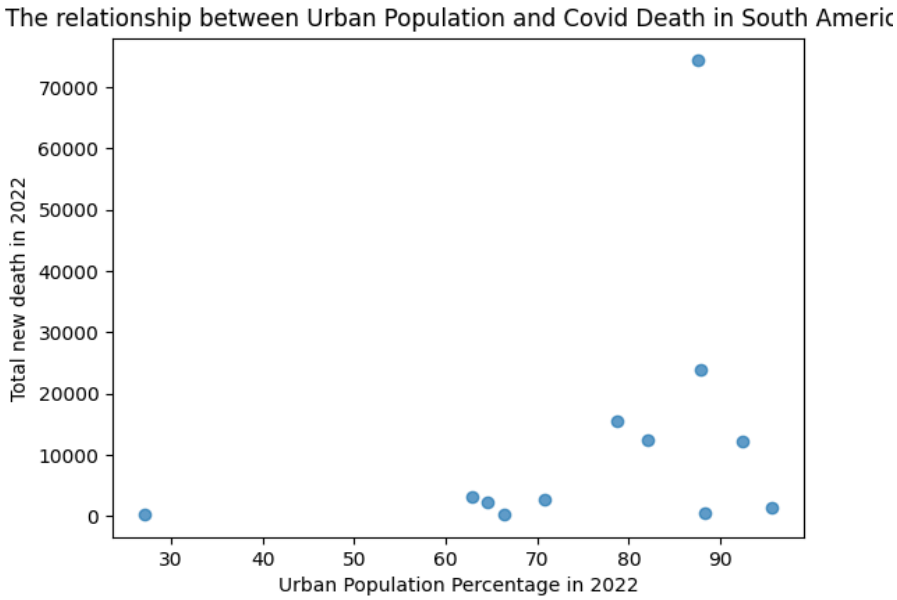
Results and Conclusions



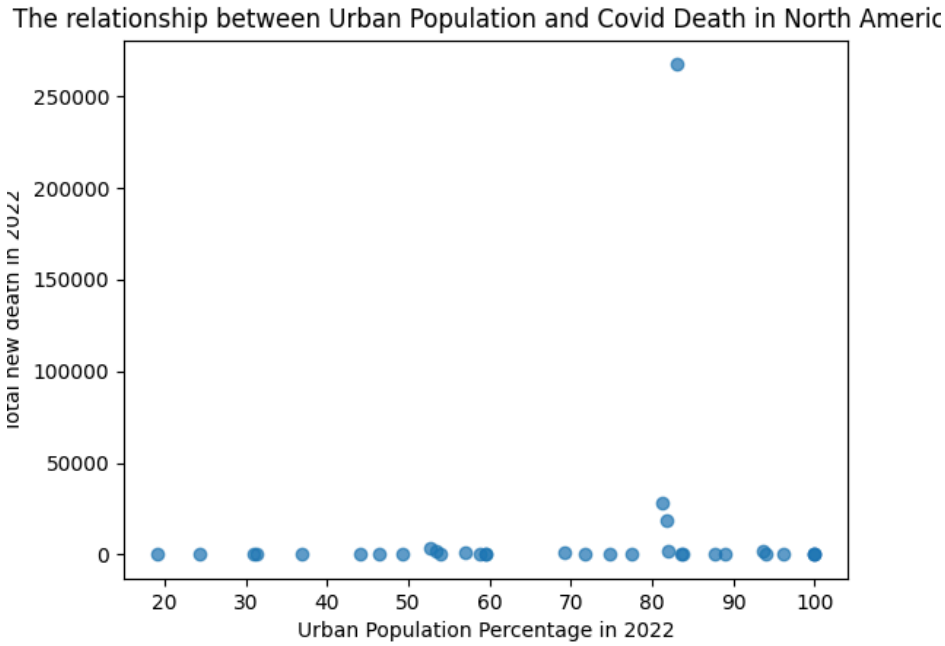
The graph illustrates that the majority of European nations have urban populations comprising 50% to 85%, with COVID-19-related deaths in 2022 generally remaining below 20,000. However, a handful of countries within this range stand out with notably higher fatality figures. One outlier records over 80,000 deaths attributed to COVID-19, while several others report more than 30,000 deaths in the same year.



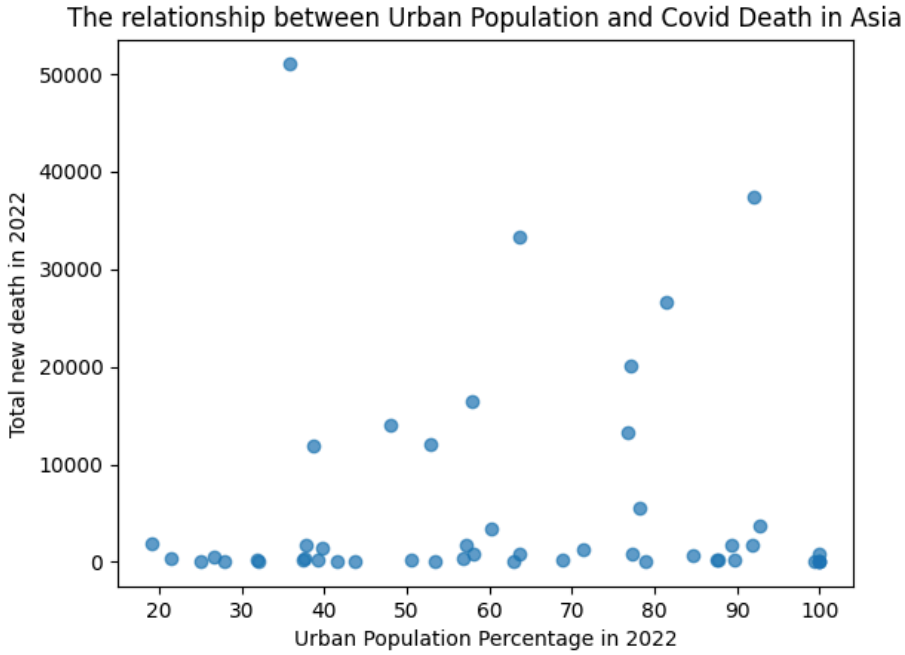
For nations in Oceania, the majority report very few deaths, often hovering around zero, despite having high urbanization rates, typically between 60% and 100%. However, one country stands out with an urbanization rate exceeding 80% and over 14,000 deaths, notably surpassing its neighbors in the same region.



The level of urbanization in South American countries seems to concentrate largely within the 60% to 90% brackets. Meanwhile, the distribution of death tolls across these nations appears relatively uniform, indicating a substantial impact of COVID-19 across the region. There is one exception, however, with a notable outlier around the 90% urbanization rate, where the death toll exceeds 70,000.



North America, as a continent, displays a notably consistent distribution of relatively low COVID-19 mortality rates, despite its urban population ranging widely from 40% to 100%. However, there is one notable exception within this range, which, although still falling within the 80% urbanization bracket like other continents, records the highest death toll exceeding 250,000. Overall, North America appears to be relatively safe, characterized by varying degrees of urban development across its countries and neighboring regions.



In conclusion, our main focus on the continents worldwide, particularly Asia, reveals intriguing and worrisome findings that warrant further examination. Asia's urban population distribution mirrors that of North America, spanning from 20% to 100%, while its mortality rates are notably varied. Several countries exhibit high death tolls compared to the continent's median, with new fatalities in 2022 ranging from 10,000 to 50,000. The significant number of countries, 10 out of 48, experiencing substantially higher fatality rates underscores Asia's heightened vulnerability compared to other continents, possibly due to its close proximity to the presumed origin of COVID-19, China.

Limitations and Future Work

The research is somewhat constrained due to the extensive scope of analyzing numerous countries. Ideally, we could identify the countries depicted in the graphs to gain deeper insights into geopolitical outliers and other factors influencing the death toll. Including and examining variables related to national healthcare policies could elucidate how certain continents, such as Europe, with many countries offering free healthcare, contribute to better citizen welfare compared to others.