The Critical Influence of Air Pollution and Socioeconomic Status on Cardiovascular Disease Mortality Rates in the U.S. with Public Health and Social Justice Implications

Bayowa Onabajo

Department of Applied Data Science and Analytics, Howard University

April 22, 2025

Research questions:

What is the association between air pollution(PM2.5), socioeconomic factors (poverty, education, and health insurance) and cardiovascular mortality rates in the U.S.

How does hypertension rate influence cardiovascular mortality rates in the U.S.

Problem statement:

Cardiovascular disease (CVD) is a leading cause of death in the United States, with growing evidence suggesting that air pollution exposure measured as particulate matter 2.5(PM 2.5) influences cardiovascular morbidity, mortality and this disproportionately affects low-income populations. Individuals from lower socioeconomic backgrounds are more likely to live in areas with higher pollution levels, overcrowding, limited healthcare access, and economic stressors that contribute to CVD risk factors such as hypertension. These inequalities raise concerns about how socioeconomic and environmental conditions intersect in shaping public health outcomes. To what degree does air pollution and socioeconomic status influence cardiovascular mortality rates in disadvantaged populations?

Data Definition

American Community Survey (2009,2010): 1-Year Estimates.

Last Updated: January 25, 2024. https://www.census.gov/data/developers/data-sets/acs-1year/2009.html https://www.census.gov/data/developers/data-sets/acs-1year/2010.html These datasets consists of above 48,000 variables as part of the

American community survey which provides data annually. The dataset covers broad social, housing, economic and demographic variables in all U.S. nations and states.The data are presented as counts. The variables from the ACS1 dataset were used in this paper as they are appropriate for the statistical approach needed to match the other datasets.

PM2.5 and cardiovascular mortality rate.

Last Updated: November 12, 2020 https://catalog.data.gov/dataset/annual-pm2-5-and-cardiovascular-mortality-rate-data-trends-modified-by-county-socioeconomi The dataset comprises socioeconomic status information for 2,132 counties in form of indexes and quintiles across the United States, provided by the U.S. Environmental Protection Agency. It also includes average annual cardiovascular mortality rates and total particulate matter 2.5 concentrations for each county over a 21-year span (1990–2010). The cardiovascular mortality data was collected from the U.S. National Center for Health Statistics, while PM2.5 levels were estimated using the EPA's Community Multiscale Air Quality (CMAQ) modeling system. Additionally, socioeconomic data was extracted from the U.S. Census Bureau.

Heart Disease Mortality by State.

Last Updated: February 25, 2022
https://www.cdc.gov/nchs/pressroom/sosmap/heart_disease_mortality/heart_disease.htm
The dataset shows the number of deaths per 100,000 population attributed to heart disease in U.S. states with variables like death rate and number of deaths. It also adjusts for differences in age distribution and population size.

Hypertension Mortality by State

Last Updated: March 3, 2022
https://www.cdc.gov/nchs/pressroom/sosmap/hypertension_mortality/hypertension.htm
The dataset shows the number of deaths per 100,000 population attributed to hypertension in U.S. states with variables like death rate and number of deaths. It also adjusts for differences in age distribution and population size.

In [5]:
```
# Import libraries
import numpy as np                    # Scientific Computing
import pandas as pd                   # Data Analysis
import matplotlib.pyplot as plt       # Plotting
import seaborn as sns                 # Statistical Data Visualization

# pandas returns all the rows and columns for the dataframe
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)

# Force pandas to display full numbers instead of scientific notation
```

```
# pd.options.display.float_format = '{:.0f}'.format

# Library to suppress warnings
import warnings
warnings.filterwarnings('ignore')
```

In [6]:
```
# Read the dataset
path = pd.read_csv('/Users/bayowaonabajo/Downloads/SES_PM25_CMR_data-2/Count

# Create the Dataframe
df_annualcounty_pm25_cmr = pd.DataFrame(path)
```

In [7]:
```
# Read the dataset
path = pd.read_csv('/Users/bayowaonabajo/Downloads/SES_PM25_CMR_data-2/Count

# Create the Dataframe
df_county_sespm25_index_quintile = pd.DataFrame(path)
```

In [8]:
```
# Read the dataset
path = pd.read_csv('/Users/bayowaonabajo/Downloads/data-table-heart-dx-mort.

# Create the Dataframe
df_heart_dx_mort = pd.DataFrame(path)
```

In [9]:
```
# Read the dataset
path = pd.read_csv('/Users/bayowaonabajo/Downloads/data-table-htn-dx-mort.cs

# Create the Dataframe
df_htn_dx_mort = pd.DataFrame(path)
```

In [10]:
```
# Read the dataset
path = pd.read_csv('/Users/bayowaonabajo/Downloads/acs_vars_2009_2010_states
```

METHODOLOGY :

Data for this study were drawn from publicly available national sources and harmonized across a cross-sectional frame (2009–2010). Data cleaning, feature engineering were done for data analysis and visualizations. Statistical and visual analysis were done with explanations for key findings.

Data Cleaning and Preparation

In [12]:
```
df_annualcounty_pm25_cmr.head()
```

Out[12]:

	Unnamed: 0	FIPS	Year	PM2.5	CMR	fip_state	state
0	1	1001	1990	9.749792	471.758888	1	AL
1	2	1001	1991	9.069443	456.869651	1	AL
2	3	1001	1992	9.105352	520.014377	1	AL
3	4	1001	1993	8.752873	454.436425	1	AL
4	5	1001	1994	9.024049	415.035332	1	AL

In [13]:

```
# Load the dataset
df = pd.read_csv('/Users/bayowaonabajo/Downloads/acs_vars_2009_2010_states.c

# State abbreviations mapping
state_abbreviations = {
    'Alabama': 'AL',
    'Alaska': 'AK',
    'Arizona': 'AZ',
    'Arkansas': 'AR',
    'California': 'CA',
    'Colorado': 'CO',
    'Connecticut': 'CT',
    'Delaware': 'DE',
    'District of Columbia': 'DC',
    'Florida': 'FL',
    'Georgia': 'GA',
    'Hawaii': 'HI',
    'Idaho': 'ID',
    'Illinois': 'IL',
    'Indiana': 'IN',
    'Iowa': 'IA',
    'Kansas': 'KS',
    'Kentucky': 'KY',
    'Louisiana': 'LA',
    'Maine': 'ME',
    'Maryland': 'MD',
    'Massachusetts': 'MA',
    'Michigan': 'MI',
    'Minnesota': 'MN',
    'Mississippi': 'MS',
    'Missouri': 'MO',
    'Montana': 'MT',
    'Nebraska': 'NE',
    'Nevada': 'NV',
    'New Hampshire': 'NH',
    'New Jersey': 'NJ',
    'New Mexico': 'NM',
    'New York': 'NY',
    'North Carolina': 'NC',
    'North Dakota': 'ND',
    'Ohio': 'OH',
    'Oklahoma': 'OK',
    'Oregon': 'OR',
    'Pennsylvania': 'PA',
```

```
    'Rhode Island': 'RI',
    'South Carolina': 'SC',
    'South Dakota': 'SD',
    'Tennessee': 'TN',
    'Texas': 'TX',
    'Utah': 'UT',
    'Vermont': 'VT',
    'Virginia': 'VA',
    'Washington': 'WA',
    'West Virginia': 'WV',
    'Wisconsin': 'WI',
    'Wyoming': 'WY',
    'Puerto Rico': 'PR'
}

# Replace state names with abbreviations
df['state'] = df['state'].map(state_abbreviations)



# Save the updated dataset to a new variable
df_acs_2009_2010_states = df

# Rename columns
df_acs_2009_2010_states = df_acs_2009_2010_states.rename(columns={'state.1':


df_acs_2009_2010_states.head()
```

Out[13]:

	state	median_income	total_population_poverty	poverty_count	total_population_u
0	AL	40489	4588899	804683	
1	AK	66953	682412	61653	
2	AZ	48745	6475485	1069897	
3	AR	37823	2806056	527378	
4	CA	58931	36202780	5128708	3

Block for extracting the merging the acs variables needed

import censusdata

import requests

import pandas as pd

censusdata.census_api_key = "YOURAPIKEY" #apikey

Define API endpoint and parameters

base_url = "https://api.census.gov/data/%7Byear%7D/acs/acs1" variables = "NAME,B19013_001E,B17001_001E,B17001_002E,B27010_001E,B27010_017E,B15002_001E state_code = "*" # Fetch data for all states

Store dataframes in a list

all_dfs = []

Loop through the years 2009, and 2010

for year in [2009, 2010]: # Construct the API request URL, inserting the current year url = f"{base_url.format(year=year)}?get={variables}&for=state:{state_code}&key={censusdata.census_api_key}"

```
    # Make the API request
    response = requests.get(url)

    # Check if successful
    if response.status_code == 200:
        print(f"Data fetched for {year}!")
        data = response.json()  # Parse through JSON response

        header = data[0]  # First row contains column names
        rows = data[1:]  # Remaining rows containing data
        df_acs = pd.DataFrame(rows, columns=header)


        # Rename columns for clarity
        df_acs = df_acs.rename(columns={
            "NAME": "state",
            "B19013_001E": "median_income",
            "B17001_001E": "total_population_poverty",
            "B17001_002E": "poverty_count",
            "B27010_001E": "total_population_uninsured",
            "B27010_017E": "uninsured_count",
            "B15002_001E": "total_population_education_18",
            "B15002_010E": "high_school_diploma",
            "B15002_011E": "ged_alternative",
            "B15002_014E": "associates_degree",
```

```
        "B15002_015E": "bachelors_degree",
        "B15002_016E": "masters_degree",
        "B15002_017E": "professional_degree",
        "B15002_018E": "doctorate_degree"
    })

    # Convert numeric columns to appropriate data types
    numeric_columns = ["median_income",
"total_population_poverty", "poverty_count",
                       "total_population_uninsured",
"uninsured_count",
                       "total_population_education_18",
"high_school_diploma",
                       "ged_alternative",
"associates_degree", "bachelors_degree",
                       "masters_degree",
"professional_degree", "doctorate_degree"]
    df_acs[numeric_columns] =
df_acs[numeric_columns].apply(pd.to_numeric, errors="coerce")

    # Calculate percentages
    df_acs["poverty_rate"] = (df_acs["poverty_count"] /
df_acs["total_population_poverty"]) * 100
    df_acs["uninsured_rate"] = (df_acs["uninsured_count"] /
df_acs["total_population_uninsured"]) * 100


   #Calculate Educated Adults
    df_acs["educated_adults"] = df_acs["high_school_diploma"]
+ df_acs["ged_alternative"] + \
                                  df_acs["associates_degree"]
+ df_acs["bachelors_degree"] + \
                                  df_acs["masters_degree"] +
df_acs["professional_degree"] + \
                                  df_acs["doctorate_degree"]

    df_acs["education_percent_educated_18"] =
(df_acs["educated_adults"] /
df_acs["total_population_education_18"]) * 100

    df_acs['year'] = year #add the year
    all_dfs.append(df_acs) #append to the list
else:
    print(f"Error for {year}: {response.status_code}")
    print(response.text)
    continue #Skips the current year to the next.
```

if not all_dfs:

print("Warning: No data was able to be collected.")

else:

df_acs_vars_09_10_states = pd.concat(all_dfs, ignore_index=True)
df_acs_vars_09_10_states

In [15]:
```
import pandas as pd

# Load the dataset
Ses_pm25_cmr_data = '/Users/bayowaonabajo/Downloads/SES_PM25_CMR_data-2/Cour

df2 = pd.read_csv(Ses_pm25_cmr_data, dtype={'FIPS': str})

# State FIPS to state abbreviation extracted from FIPS in original ses_pm25_
state_fips_mapping = {
    '01': 'AL', '02': 'AK', '04': 'AZ', '05': 'AR', '06': 'CA', '08': 'CO',
    '10': 'DE', '11': 'DC', '12': 'FL', '13': 'GA', '15': 'HI', '16': 'ID',
    '18': 'IN', '19': 'IA', '20': 'KS', '21': 'KY', '22': 'LA', '23': 'ME',
    '25': 'MA', '26': 'MI', '27': 'MN', '28': 'MS', '29': 'MO', '30': 'MT',
    '32': 'NV', '33': 'NH', '34': 'NJ', '35': 'NM', '36': 'NY', '37': 'NC',
    '39': 'OH', '40': 'OK', '41': 'OR', '42': 'PA', '44': 'RI', '45': 'SC',
    '47': 'TN', '48': 'TX', '49': 'UT', '50': 'VT', '51': 'VA', '53': 'WA',
    '55': 'WI', '56': 'WY'
}

# Extract state FIPS and map to abbreviations
def extract_state_info(df):
    df['fip_state'] = df['FIPS'].str[:2]  # Extract first two digits
    df['state'] = df['fip_state'].map(state_fips_mapping)
    return df

df2 = extract_state_info(df2)
df2.head()

# update dataset with fip state codes and states
updated_file = '/Users/bayowaonabajo/Downloads/SES_PM25_CMR_data-2/County_ar
df2.to_csv(updated_file, index=False)

# Display few rows
df2.head()
```

Out[15]:

	Unnamed: 0	FIPS	Year	PM2.5	CMR	fip_state	state
0	1	01001	1990	9.749792	471.758888	01	AL
1	2	01001	1991	9.069443	456.869651	01	AL
2	3	01001	1992	9.105352	520.014377	01	AL
3	4	01001	1993	8.752873	454.436425	01	AL
4	5	01001	1994	9.024049	415.035332	01	AL

In [16]:

```
import pandas as pd

# Load the dataset
Ses_index_quintile_file = '/Users/bayowaonabajo/Downloads/SES_PM25_CMR_data-

df1 = pd.read_csv(Ses_index_quintile_file, dtype={'FIPS': str})

# State FIPS to state abbreviation extracted from FIPS in original ses_index
state_fips_mapping = {
    '01': 'AL', '02': 'AK', '04': 'AZ', '05': 'AR', '06': 'CA', '08': 'CO',
    '10': 'DE', '11': 'DC', '12': 'FL', '13': 'GA', '15': 'HI', '16': 'ID',
    '18': 'IN', '19': 'IA', '20': 'KS', '21': 'KY', '22': 'LA', '23': 'ME',
    '25': 'MA', '26': 'MI', '27': 'MN', '28': 'MS', '29': 'MO', '30': 'MT',
    '32': 'NV', '33': 'NH', '34': 'NJ', '35': 'NM', '36': 'NY', '37': 'NC',
    '39': 'OH', '40': 'OK', '41': 'OR', '42': 'PA', '44': 'RI', '45': 'SC',
    '47': 'TN', '48': 'TX', '49': 'UT', '50': 'VT', '51': 'VA', '53': 'WA',
    '55': 'WI', '56': 'WY'
}

# Extract state FIPS and map to abbreviations
def extract_state_info(df):
    df['fip_state'] = df['FIPS'].str[:2]  # Extract first two digits
    df['state'] = df['fip_state'].map(state_fips_mapping)
    return df

df1 = extract_state_info(df1)
df1.head()

# update dataset with fip state codes and states
updated_file = '/Users/bayowaonabajo/Downloads/SES_PM25_CMR_data-2/County_SE
df1.to_csv(updated_file, index=False)

df1

# Display few rows
df1.head()
```

Out[16]:

	Unnamed: 0	FIPS	SES_index_1990	SES_index_2000	SES_index_2010	SES_quinti
0	1	01001	-0.079387	-0.322846	-0.405150	
1	2	01003	-0.187240	-0.467794	-0.403987	
2	3	01005	1.279538	2.013751	1.740142	
3	4	01009	0.124421	-0.375181	-0.405849	
4	5	01011	2.877256	3.519681	2.617074	

In [17]:
```
# Display first ten rows of the dataframe
df_annualcounty_pm25_cmr.head()
```

Out[17]:

	Unnamed: 0	FIPS	Year	PM2.5	CMR	fip_state	state
0	1	1001	1990	9.749792	471.758888	1	AL
1	2	1001	1991	9.069443	456.869651	1	AL
2	3	1001	1992	9.105352	520.014377	1	AL
3	4	1001	1993	8.752873	454.436425	1	AL
4	5	1001	1994	9.024049	415.035332	1	AL

In [18]:
```
# Display last ten rows of the dataframe
df_annualcounty_pm25_cmr.tail(5)
```

Out[18]:

	Unnamed: 0	FIPS	Year	PM2.5	CMR	fip_state	state
44767	44768	56037	2006	3.776910	247.510138	56	WY
44768	44769	56037	2007	3.609803	292.450269	56	WY
44769	44770	56037	2008	3.297100	182.189745	56	WY
44770	44771	56037	2009	3.119896	242.828987	56	WY
44771	44772	56037	2010	3.230996	254.860863	56	WY

In [19]:
```
path = pd.read_csv('/Users/bayowaonabajo/Downloads/SES_PM25_CMR_data-2/Count

df_county_sespm25_index_quintile = pd.DataFrame(path)
```

In [20]:
```
df_county_sespm25_index_quintile.head()
```

Out[20]:

	Unnamed: 0	FIPS	SES_index_1990	SES_index_2000	SES_index_2010	SES_quintile
0	1	1001	-0.079387	-0.322846	-0.405150	
1	2	1003	-0.187240	-0.467794	-0.403987	
2	3	1005	1.279538	2.013751	1.740142	
3	4	1009	0.124421	-0.375181	-0.405849	
4	5	1011	2.877256	3.519681	2.617074	

In [21]:
```
#df_county_sespm25_index_quintile.tail()
```

In [22]:
```
df_heart_dx_mort.head()
```

Out[22]:

	YEAR	STATE	RATE	DEATHS	URL
0	2022	AL	234.2	14958	/nchs/pressroom/states/alabama/al.htm
1	2022	AK	145.7	1013	/nchs/pressroom/states/alaska/ak.htm
2	2022	AZ	148.5	14593	/nchs/pressroom/states/arizona/az.htm
3	2022	AR	224.1	8664	/nchs/pressroom/states/arkansas/ar.htm
4	2022	CA	142.4	66340	/nchs/pressroom/states/california/ca.htm

In [23]:
```
#df_heart_dx_mort.tail()
```

In [24]:
```
df_htn_dx_mort['YEAR'].unique()
```

Out[24]:
```
array([2022, 2021, 2020, 2019, 2018, 2017, 2016, 2015, 2014, 2005])
```

In [25]:
```
df_htn_dx_mort.head(5)
```

Out[25]:

	YEAR	STATE	RATE	DEATHS	URL
0	2022	AL	13.2	849	/nchs/pressroom/states/alabama/al.htm
1	2022	AK	8.6	56	/nchs/pressroom/states/alaska/ak.htm
2	2022	AZ	11.3	1109	/nchs/pressroom/states/arizona/az.htm
3	2022	AR	12.1	454	/nchs/pressroom/states/arkansas/ar.htm
4	2022	CA	14.4	6727	/nchs/pressroom/states/california/ca.htm

In [26]:
```
df_heart_dx_mort['YEAR'].unique()
```

Out[26]:
```
array([2022, 2021, 2020, 2019, 2018, 2017, 2016, 2015, 2014, 2005])
```

In [27]:
```
df_heart_dx_mort.head()
```

Out[27]:

	YEAR	STATE	RATE	DEATHS	URL
0	2022	AL	234.2	14958	/nchs/pressroom/states/alabama/al.htm
1	2022	AK	145.7	1013	/nchs/pressroom/states/alaska/ak.htm
2	2022	AZ	148.5	14593	/nchs/pressroom/states/arizona/az.htm
3	2022	AR	224.1	8664	/nchs/pressroom/states/arkansas/ar.htm
4	2022	CA	142.4	66340	/nchs/pressroom/states/california/ca.htm

In [28]:
```
#df_htn_dx_mort.tail()
```

In [29]:
```
# Display first ten rows of the dataframe
df_acs_2009_2010_states.head()
```

Out[29]:

	state	median_income	total_population_poverty	poverty_count	total_population_u
0	AL	40489	4588899	804683	
1	AK	66953	682412	61653	
2	AZ	48745	6475485	1069897	
3	AR	37823	2806056	527378	
4	CA	58931	36202780	5128708	3

In [30]:
```
# Display last ten rows of the dataframe
#df_acs_2009_2010_states.tail()
```

In [31]:
```
df_annualcounty_pm25_cmr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 44772 entries, 0 to 44771
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  44772 non-null  int64  
 1   FIPS        44772 non-null  int64  
 2   Year        44772 non-null  int64  
 3   PM2.5       44772 non-null  float64
 4   CMR         44772 non-null  float64
 5   fip_state   44772 non-null  int64  
 6   state       44772 non-null  object 
dtypes: float64(2), int64(4), object(1)
memory usage: 2.4+ MB
```

In [32]:
```
# This is the number of rows and columns in the data
df_annualcounty_pm25_cmr.shape
```

Out[32]: (44772, 7)

The dataframe has 44772 rows and 7 columns. The total number of datapoints expected is 313404

In [34]:
```
df_county_sespm25_index_quintile.shape
```

Out[34]: (2132, 10)

The dataframe has 2132 rows and 10 columns. The total number of datapoints expected is 21320

In [36]:
```
df_county_sespm25_index_quintile.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2132 entries, 0 to 2131
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Unnamed: 0         2132 non-null   int64
 1   FIPS               2132 non-null   int64
 2   SES_index_1990     2132 non-null   float64
 3   SES_index_2000     2132 non-null   float64
 4   SES_index_2010     2132 non-null   float64
 5   SES_quintile_1990  2132 non-null   object
 6   SES_quintile_2000  2132 non-null   object
 7   SES_quintile_2010  2132 non-null   object
 8   fip_state          2132 non-null   int64
 9   state              2132 non-null   object
dtypes: float64(3), int64(3), object(4)
memory usage: 166.7+ KB
```

In [37]:
```
df_heart_dx_mort.shape
```

Out[37]: (501, 5)

The dataframe has 501 rows and 5 columns. The total number of datapoints expected is 2505

In [39]:
```
df_heart_dx_mort.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 501 entries, 0 to 500
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   YEAR    501 non-null    int64
 1   STATE   501 non-null    object
 2   RATE    501 non-null    float64
 3   DEATHS  501 non-null    object
 4   URL     501 non-null    object
dtypes: float64(1), int64(1), object(3)
memory usage: 19.7+ KB
```

In [40]:
```
df_htn_dx_mort.shape
```

Out[40]: (501, 5)

The dataframe has 501 rows and 5 columns. The total number of datapoints expected is 2505

In [42]:
```
df_htn_dx_mort.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 501 entries, 0 to 500
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   YEAR    501 non-null    int64  
 1   STATE   501 non-null    object 
 2   RATE    501 non-null    float64
 3   DEATHS  501 non-null    object 
 4   URL     501 non-null    object 
dtypes: float64(1), int64(1), object(3)
memory usage: 19.7+ KB
```

In [43]:
```
df_acs_2009_2010_states.shape
```

Out[43]: (104, 20)

The dataframe has 104 rows and 20 columns. The total number of datapoints expected is 2080

In [45]:
```
df_acs_2009_2010_states.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 104 entries, 0 to 103
Data columns (total 20 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   state                           104 non-null    object 
 1   median_income                   104 non-null    int64  
 2   total_population_poverty        104 non-null    int64  
 3   poverty_count                   104 non-null    int64  
 4   total_population_uninsured      104 non-null    int64  
 5   uninsured_count                 104 non-null    int64  
 6   total_population_education_18   104 non-null    int64  
 7   high_school_diploma             104 non-null    int64  
 8   ged_alternative                 104 non-null    int64  
 9   associates_degree               104 non-null    int64  
 10  bachelors_degree                104 non-null    int64  
 11  masters_degree                  104 non-null    int64  
 12  professional_degree             104 non-null    int64  
 13  doctorate_degree                104 non-null    int64  
 14  fip                             104 non-null    int64  
 15  poverty_rate                    104 non-null    float64
 16  uninsured_rate                  104 non-null    float64
 17  educated_adults                 104 non-null    int64  
 18  education_percent_educated_18   104 non-null    float64
 19  year                            104 non-null    int64  
dtypes: float64(3), int64(16), object(1)
memory usage: 16.4+ KB
```

In [46]:
```
df_annualcounty_pm25_cmr['state'].unique()
```

Out[46]:
```
array(['AL', 'AZ', 'AR', 'CA', 'CO', 'CT', 'DE', 'DC', 'FL', 'GA', 'ID',
       'IL', 'IN', 'IA', 'KS', 'KY', 'LA', 'ME', 'MD', 'MA', 'MI', 'MN',
       'MS', 'MO', 'MT', 'NE', 'NV', 'NH', 'NJ', 'NM', 'NY', 'NC', 'ND',
       'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VT',
       'VA', 'WA', 'WV', 'WI', 'WY'], dtype=object)
```

In [47]:
```
#create a list of the columns in the dataset
df_annualcounty_pm25_cmrCol = df_annualcounty_pm25_cmr.columns
df_annualcounty_pm25_cmrCol
```

Out[47]:
```
Index(['Unnamed: 0', 'FIPS', 'Year', 'PM2.5', 'CMR', 'fip_state', 'state'],
dtype='object')
```

In [48]:
```
# Update the Headers for Consistency

df_annualcounty_pm25_cmrCol = df_annualcounty_pm25_cmr.rename(columns = {'Ur

# view the new columns and update the variable
df_annualcounty_pm25_cmr = df_annualcounty_pm25_cmrCol

df_annualcounty_pm25_cmr.head()
```

Out[48]:

	indexes	FIPS	Year	PM2.5	CMR	fip_state	state
0	1	1001	1990	9.749792	471.758888	1	AL
1	2	1001	1991	9.069443	456.869651	1	AL
2	3	1001	1992	9.105352	520.014377	1	AL
3	4	1001	1993	8.752873	454.436425	1	AL
4	5	1001	1994	9.024049	415.035332	1	AL

Renamed the column "Unnamed:0' to indexes for a more explanatory dataset.

In [50]:
```
df_annualcounty_pm25_cmr_filtered = df_annualcounty_pm25_cmr[(df_annualcount
```

In [51]:
```
df_annualcounty_pm25_cmr_filtered.tail()
```

Out[51]:

	indexes	FIPS	Year	PM2.5	CMR	fip_state	state
44729	44730	56029	2010	2.571525	170.765285	56	WY
44749	44750	56033	2009	2.566431	235.312525	56	WY
44750	44751	56033	2010	2.642380	175.671813	56	WY
44770	44771	56037	2009	3.119896	242.828987	56	WY
44771	44772	56037	2010	3.230996	254.860863	56	WY

Dropped rows with year 1990 to 2008 for a matching analysis of timeline with the ACS 2009 and 2010 dataset. Dropping the rows narrowed the number of states in the dataset to 49 from 50.

In [53]:
```
df_annualstate_county_pm25_cmr = df_annualcounty_pm25_cmr_filtered

df_annualstate_county_pm25_cmr['state'].unique()
```

Out[53]:
```
array(['AL', 'AZ', 'AR', 'CA', 'CO', 'CT', 'DE', 'DC', 'FL', 'GA', 'ID',
       'IL', 'IN', 'IA', 'KS', 'KY', 'LA', 'ME', 'MD', 'MA', 'MI', 'MN',
       'MS', 'MO', 'MT', 'NE', 'NV', 'NH', 'NJ', 'NM', 'NY', 'NC', 'ND',
       'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VT',
       'VA', 'WA', 'WV', 'WI', 'WY'], dtype=object)
```

In [54]:
```
# Determine the number of missing values
df_annualstate_county_pm25_cmr.isnull().sum()
```

Out[54]:
```
indexes      0
FIPS         0
Year         0
PM2.5        0
CMR          0
fip_state    0
state        0
dtype: int64
```

In [55]:
```
# Determine the percentage of missing values
# Typically less than five percent missing values may not affect the results
# More than 5% can be dropped, replaced with existing data, or imputed using

def missing(Dataframe):
    print('Percentage of missing values in the dataset:\n',
          round((Dataframe.isnull().sum() *100/len(Dataframe)), 2).sort_valu


missing(df_annualstate_county_pm25_cmr)
```

```
Percentage of missing values in the dataset:
 indexes      0.0
FIPS         0.0
Year         0.0
PM2.5        0.0
CMR          0.0
fip_state    0.0
state        0.0
dtype: float64
```

I have no missing values in this dataset which is good for my analysis as it allows for a faster and complete statistical analysis, exploration and visualization

In [57]:
```
# create a list of the columns in the dataset
df_county_sespm25_index_quintileCol = df_county_sespm25_index_quintile.colum
df_county_sespm25_index_quintileCol
```

Out[57]:
```
Index(['Unnamed: 0', 'FIPS', 'SES_index_1990', 'SES_index_2000',
       'SES_index_2010', 'SES_quintile_1990', 'SES_quintile_2000',
       'SES_quintile_2010', 'fip_state', 'state'],
      dtype='object')
```

In [58]:
```
# Update the Headers for Syntax Consistency

df_county_sespm25_index_quintileCol = df_county_sespm25_index_quintile.renam

# view the new columns and update the variable

df_county_sespm25_index_quintile = df_county_sespm25_index_quintileCol

df_county_sespm25_index_quintile.head()
```

Out[58]:

	indexes	FIPS	SES_index_1990	SES_index_2000	SES_index_2010	SES_quintile_1
0	1	1001	-0.079387	-0.322846	-0.405150	
1	2	1003	-0.187240	-0.467794	-0.403987	
2	3	1005	1.279538	2.013751	1.740142	
3	4	1009	0.124421	-0.375181	-0.405849	
4	5	1011	2.877256	3.519681	2.617074	

Renamed the column "Unnamed:0' to indexes for a more explanatory dataset.

In [60]:
```
# Determine the number of missing values

df_county_sespm25_index_quintile.isnull().sum()
```

Out[60]:
```
indexes              0
FIPS                 0
SES_index_1990       0
SES_index_2000       0
SES_index_2010       0
SES_quintile_1990    0
SES_quintile_2000    0
SES_quintile_2010    0
fip_state            0
state                0
dtype: int64
```

In [61]:
```
#  function to determine the percentage of missing values
# Typically less than five percent missing values may not affect the results
# More than 5% can be dropped, replaced with existing data, or imputed using

def missing(Dataframe):
    print('Percentage of missing values in the dataset:\n',
          round((Dataframe.isnull().sum() *100/len(Dataframe)), 2).sort_valu


missing(df_county_sespm25_index_quintile)
```

```
Percentage of missing values in the dataset:
 indexes              0.0
FIPS                 0.0
SES_index_1990       0.0
SES_index_2000       0.0
SES_index_2010       0.0
SES_quintile_1990    0.0
SES_quintile_2000    0.0
SES_quintile_2010    0.0
fip_state            0.0
state                0.0
dtype: float64
```

In [62]:
```
#create a list of the columns in the dataset
df_heart_dx_mortCol = df_heart_dx_mort.columns
df_heart_dx_mortCol
```

Out[62]:
```
Index(['YEAR', 'STATE', 'RATE', 'DEATHS', 'URL'], dtype='object')
```

In [63]:
```
#create a list of the columns in the dataset
df_heart_dx_mortCol = df_heart_dx_mort.columns
df_heart_dx_mortCol
```

Out[63]:
```
Index(['YEAR', 'STATE', 'RATE', 'DEATHS', 'URL'], dtype='object')
```

In [64]:
```
# Update the Headers for Consistency

df_heart_dx_mortCol = df_heart_dx_mort.rename(columns = {'STATE':'state'})

# view the new columns and update the variable

df_heart_dx_mort = df_heart_dx_mortCol

df_heart_dx_mort.head()
```

Out[64]:

	YEAR	state	RATE	DEATHS	URL
0	2022	AL	234.2	14958	/nchs/pressroom/states/alabama/al.htm
1	2022	AK	145.7	1013	/nchs/pressroom/states/alaska/ak.htm
2	2022	AZ	148.5	14593	/nchs/pressroom/states/arizona/az.htm
3	2022	AR	224.1	8664	/nchs/pressroom/states/arkansas/ar.htm
4	2022	CA	142.4	66340	/nchs/pressroom/states/california/ca.htm

Changed the column name 'STATE' to 'state' in this cardiovascular disease rate dataset to allign with similar column names in the other datasets for easier manipulation and merging if needed.

In [66]:
```
df_heart_dx_mort.head()
```

Out[66]:

	YEAR	state	RATE	DEATHS	URL
0	2022	AL	234.2	14958	/nchs/pressroom/states/alabama/al.htm
1	2022	AK	145.7	1013	/nchs/pressroom/states/alaska/ak.htm
2	2022	AZ	148.5	14593	/nchs/pressroom/states/arizona/az.htm
3	2022	AR	224.1	8664	/nchs/pressroom/states/arkansas/ar.htm
4	2022	CA	142.4	66340	/nchs/pressroom/states/california/ca.htm

In [67]:

```
# Load the dataset
df = df_heart_dx_mort

df['state'] = df['state'].replace({
    'District of Columbia' : 'DC',

})

# Save the updated dataset
df_heart_dx_mort = df


df_heart_dx_mort.head(5)
```

Out[67]:

	YEAR	state	RATE	DEATHS	URL
0	2022	AL	234.2	14958	/nchs/pressroom/states/alabama/al.htm
1	2022	AK	145.7	1013	/nchs/pressroom/states/alaska/ak.htm
2	2022	AZ	148.5	14593	/nchs/pressroom/states/arizona/az.htm
3	2022	AR	224.1	8664	/nchs/pressroom/states/arkansas/ar.htm
4	2022	CA	142.4	66340	/nchs/pressroom/states/california/ca.htm

Changed the variable 'District of columbia' to 'DC' in the state column for conformity with the rest of the dataset.

In [69]:

```
print(df['state'].unique())
```

```
['AL' 'AK' 'AZ' 'AR' 'CA' 'CO' 'CT' 'DE' 'DC' 'FL' 'GA' 'HI' 'ID' 'IL'
 'IN' 'IA' 'KS' 'KY' 'LA' 'ME' 'MD' 'MA' 'MI' 'MN' 'MS' 'MO' 'MT' 'NE'
 'NV' 'NH' 'NJ' 'NM' 'NY' 'NC' 'ND' 'OH' 'OK' 'OR' 'PA' 'RI' 'SC' 'SD'
 'TN' 'TX' 'UT' 'VT' 'VA' 'WA' 'WV' 'WI' 'WY']
```

In [70]:

```
# Determine the number of missing values

df_heart_dx_mort.isnull().sum()
```

Out[70]:
```
YEAR      0
state     0
RATE      0
DEATHS    0
URL       0
dtype: int64
```

In [71]:
```
def missing(Dataframe):
    print('Percentage of missing values in the dataset:\n',
          round((Dataframe.isnull().sum() *100/len(Dataframe)), 2).sort_valu


missing(df_heart_dx_mort)
```

```
Percentage of missing values in the dataset:
 YEAR      0.0
state     0.0
RATE      0.0
DEATHS    0.0
URL       0.0
dtype: float64
```

I have no missing values in this dataset which is also good for my analysis as it allows for a faster and complete statistical analysis, exploration and visualization

In [73]:
```
#create a list of the columns in the dataset
df_htn_dx_mortCol = df_htn_dx_mort.columns
df_htn_dx_mortCol
```

Out[73]:
```
Index(['YEAR', 'STATE', 'RATE', 'DEATHS', 'URL'], dtype='object')
```

Changed the column name 'STATE' to 'state' in this hypertensive disease rate dataset to allign with similar column names in the other datasets for easier manipulation and merging if needed.

In [75]:
```
# Update the Headers for Consistency

df_htn_dx_mortCol = df_htn_dx_mort.rename(columns = {'STATE':'state'})

# view the new columns and update the variable

df_htn_dx_mort = df_htn_dx_mortCol

df_htn_dx_mort.head()
```

Out[75]:

	YEAR	state	RATE	DEATHS	URL
0	2022	AL	13.2	849	/nchs/pressroom/states/alabama/al.htm
1	2022	AK	8.6	56	/nchs/pressroom/states/alaska/ak.htm
2	2022	AZ	11.3	1109	/nchs/pressroom/states/arizona/az.htm
3	2022	AR	12.1	454	/nchs/pressroom/states/arkansas/ar.htm
4	2022	CA	14.4	6727	/nchs/pressroom/states/california/ca.htm

In [76]:

```
# Load the dataset
df = df_htn_dx_mort

df['state'] = df['state'].replace({
    'District of Columbia' : 'DC',

})

# Save the updated dataset
df_htn_dx_mort = df


df_htn_dx_mort.head()
```

Out[76]:

	YEAR	state	RATE	DEATHS	URL
0	2022	AL	13.2	849	/nchs/pressroom/states/alabama/al.htm
1	2022	AK	8.6	56	/nchs/pressroom/states/alaska/ak.htm
2	2022	AZ	11.3	1109	/nchs/pressroom/states/arizona/az.htm
3	2022	AR	12.1	454	/nchs/pressroom/states/arkansas/ar.htm
4	2022	CA	14.4	6727	/nchs/pressroom/states/california/ca.htm

Changed the variable 'District of columbia' to 'DC' in the state column for conformity with the rest of the dataset.

In [78]:

```
# number of missing values

df_htn_dx_mort.isnull().sum()
```

Out[78]:

```
YEAR      0
state     0
RATE      0
DEATHS    0
URL       0
dtype: int64
```

In [79]:

```
def missing(Dataframe):
    print('Percentage of missing values in the dataset:\n',
          round((Dataframe.isnull().sum() *100/len(Dataframe)), 2).sort_valu
```

```
missing(df_htn_dx_mort)
```

```
Percentage of missing values in the dataset:
 YEAR      0.0
state     0.0
RATE      0.0
DEATHS    0.0
URL       0.0
dtype: float64
```

I have no missing values in this dataset which is also good for my analysis as it allows for a faster and complete statistical analysis, exploration and visualization

In [81]:
```
#create a list of the columns in the dataset
df_acs_2009_2010_statesCol = df_acs_2009_2010_states.columns
df_acs_2009_2010_statesCol
```

Out[81]:
```
Index(['state', 'median_income', 'total_population_poverty', 'poverty_coun
t',
       'total_population_uninsured', 'uninsured_count',
       'total_population_education_18', 'high_school_diploma',
       'ged_alternative', 'associates_degree', 'bachelors_degree',
       'masters_degree', 'professional_degree', 'doctorate_degree', 'fip',
       'poverty_rate', 'uninsured_rate', 'educated_adults',
       'education_percent_educated_18', 'year'],
      dtype='object')
```

The column names in this collated ACS rate dataset allign with research goals so i will keep them as they are.

In [83]:
```
df_acs_2009_2010_states.head()
```

Out[83]:

	state	median_income	total_population_poverty	poverty_count	total_population_u
0	AL	40489	4588899	804683	
1	AK	66953	682412	61653	
2	AZ	48745	6475485	1069897	
3	AR	37823	2806056	527378	
4	CA	58931	36202780	5128708	3

In [84]:
```
# number of missing values

df_acs_2009_2010_states.isnull().sum()
```

Out[84]:
```
state                           0
median_income                   0
total_population_poverty        0
poverty_count                   0
total_population_uninsured      0
uninsured_count                 0
total_population_education_18   0
high_school_diploma             0
ged_alternative                 0
associates_degree               0
bachelors_degree                0
masters_degree                  0
professional_degree             0
doctorate_degree                0
fip                             0
poverty_rate                    0
uninsured_rate                  0
educated_adults                 0
education_percent_educated_18   0
year                            0
dtype: int64
```

In [85]:
```
def missing(Dataframe):
    print('Percentage of missing values in the dataset:\n',
          round((Dataframe.isnull().sum() *100/len(Dataframe)), 2).sort_valu


missing(df_acs_2009_2010_states)
```

```
Percentage of missing values in the dataset:
 state                           0.0
median_income                   0.0
education_percent_educated_18   0.0
educated_adults                 0.0
uninsured_rate                  0.0
poverty_rate                    0.0
fip                             0.0
doctorate_degree                0.0
professional_degree             0.0
masters_degree                  0.0
bachelors_degree                0.0
associates_degree               0.0
ged_alternative                 0.0
high_school_diploma             0.0
total_population_education_18   0.0
uninsured_count                 0.0
total_population_uninsured      0.0
poverty_count                   0.0
total_population_poverty        0.0
year                            0.0
dtype: float64
```

I have no missing values in this collated ACS dataset which is also good for my analysis as it allows for a faster and complete statistical analysis, exploration and visualization

In [87]:
```
df_annualstate_county_pm25_cmr.head()
```

Out[87]:

	indexes	FIPS	Year	PM2.5	CMR	fip_state	state
19	20	1001	2009	6.402091	330.876172	1	AL
20	21	1001	2010	6.942778	316.911479	1	AL
40	41	1003	2009	5.419087	270.402216	1	AL
41	42	1003	2010	5.837704	276.377191	1	AL
61	62	1005	2009	5.840124	383.159080	1	AL

Exploratory Data Analysis and Feature Engineering

Descriptive Statistics

In [89]:
```
df_annualstate_county_pm25_cmr.describe()
```

Out[89]:

	indexes	FIPS	Year	PM2.5	CMR	fip_s
count	4264.000000	4264.000000	4264.000000	4264.000000	4264.000000	4264.0
mean	22396.000000	30599.787992	2009.500000	6.171229	257.605458	30.5
std	12926.077525	15142.415588	0.500059	1.396911	56.675549	15.1
min	20.000000	1001.000000	2009.000000	2.192728	106.135757	1.0
25%	11208.000000	18162.500000	2009.000000	5.521922	216.515285	18.0
50%	22396.000000	29164.000000	2009.500000	6.391946	250.385485	29.0
75%	33584.000000	45019.500000	2010.000000	7.126114	291.266376	45.0
max	44772.000000	56037.000000	2010.000000	9.384544	557.426037	56.0

The minimum and maximum values for the pm2.5 are 2.19 µg/m³ and 9.38 µg/m³ while the minimum and maximum values for the cardiovascular mortality rate are 106.1 per 100,000 and 557.4 per 100,000.

The mean PM2.5 of 6.17 and median of 6.39 suggests a relatively normal distribution for particulate matter of size 2.5

The mean CMR of 257.6 and median of 250.4 suggests a near symmetric distribution as well.

The quartile ranges are 25th percentile of 5.5 and 216.5 for PM2.5 and CMR respectively. The 75th percentile are 7.12 and 291.26 for PM2.5 and CMR respectively.

The standard deviation of PM2.5 at 1.39 indicates small variability across counties and states.However the standard deviation of CMR at 56.7 shows a high spread in cardiovascular mortality rates across states.

In [91]:
```
df_heart_dx_mort.describe()
```

Out[91]:

	YEAR	RATE
count	501.000000	501.000000
mean	2016.710579	172.287425
std	4.611515	32.655107
min	2005.000000	114.900000
25%	2015.000000	149.300000
50%	2018.000000	163.400000
75%	2020.000000	192.000000
max	2022.000000	306.400000

The minimum and maximum values for this dataframe are 114.9 and 306.4 per 100,000.

The mean of 172.3 and median of 163.4 suggests a right-skewed distribution.

The quartile ranges are 25th percentile of 149.3. and 75th percentile of 192.0.

The standard deviation of 32.7 is high and could allude to significant differences in heart disease mortality rates across states in the USA.

In [93]:
```
df_htn_dx_mort.describe()
```

Out[93]:

	YEAR	RATE
count	501.000000	501.000000
mean	2016.710579	8.628343
std	4.611515	2.518634
min	2005.000000	0.000000
25%	2015.000000	6.900000
50%	2018.000000	8.300000
75%	2020.000000	10.100000
max	2022.000000	20.400000

The minimum and maximum values for this dataframe are 0.0 and 20.4 deaths per 100,000.

The mean of 8.63 and median of 8.30 suggests a right-skewed distribution.

The quartile ranges are 25th percentile of 6.9 and 75th percentile of 10.1.

The standard deviation of 2.51 indicates moderate variability in hypertension mortality rates across states in the USA.

In [95]:
```
df_county_sespm25_index_quintile.describe()
```

Out[95]:

	indexes	FIPS	SES_index_1990	SES_index_2000	SES_index_201
count	2132.000000	2132.000000	2.132000e+03	2.132000e+03	2.132000e+0
mean	1066.500000	30599.787992	-7.332054e-17	8.998431e-17	1.999651e-
std	615.599708	15144.191928	9.641826e-01	9.837311e-01	9.556947e-0
min	1.000000	1001.000000	-2.535586e+00	-1.646289e+00	-1.836970e+0
25%	533.750000	18162.500000	-6.293172e-01	-6.843596e-01	-6.735622e-0
50%	1066.500000	29164.000000	-1.083418e-01	-2.034422e-01	-1.362228e-0
75%	1599.250000	45019.500000	5.120400e-01	4.586209e-01	4.726322e-0
max	2132.000000	56037.000000	5.645396e+00	6.646980e+00	6.456330e+0

The mean index of 1066 and median of 1066 indicates a normal distribution.

In [97]:
```
df_acs_2009_2010_states.describe()
```

Out[97]:

	median_income	total_population_poverty	poverty_count	total_population_unin
count	104.000000	1.040000e+02	1.040000e+02	1.04000
mean	49604.144231	5.847806e+06	8.895052e+05	5.89802
std	9270.377961	6.565761e+06	1.035133e+06	6.60977
min	18314.000000	5.299820e+05	5.214400e+04	5.33716
25%	43628.000000	1.689948e+06	2.264710e+05	1.71014
50%	48258.000000	4.056070e+06	6.204850e+05	4.08210
75%	55437.250000	6.489280e+06	9.851172e+05	6.51268
max	69272.000000	3.659337e+07	5.783043e+06	3.68155

The dataset shows considerable variability across several socioeconomic indicators. Median income ranges from a low of 18,314 to a high of 69,272, reflecting significant economic disparities. The number of individuals without health insurance also varies widely, from as few as 2,532 to as many as 914,426 people, highlighting potential disparities in healthcare access. Educational attainment, specifically the percentage of the state population with only higher education, spans from 25.2% to 36.25%. The average rate of higher education is 32.1%, closely aligned with the median of 32.44%, suggesting a relatively symmetric distribution with minimal skewness. In contrast, poverty rates exhibit a broader spread, ranging from 8.3% to 45.03%. The mean poverty rate is 14.9%, while the median is slightly lower at 14.24%, indicating a right-skewed distribution where a smaller number of states experience significantly higher poverty levels. Supporting this, the interquartile range (IQR) for poverty is 5.25, signifying notable dispersion within the central 50% of the data. Moreover, the variance in poverty rate exceeds the mean, highlighting substantial variability across observations. Health uninsurance rates, while generally lower, still display meaningful variation—from 0.31% to 4.65%. The mean rate stands at 1.82%, compared to a median of 1.54%, again suggesting a mild right-skew in the distribution. However, the variance here is relatively low (0.88), indicating that the data is more clustered around the central tendency than other variables. Overall, the patterns suggest that while some indicators like educational attainment show consistency across states, others—particularly poverty and income—reveal significant inequality. The skewed distributions and wide IQRs in these domains may require further investigation into structural and regional factors influencing these disparities.

In [99]:
```
#Merge SES index quintile data and PM25/CMR data
#Read SES data with 'FIPS' as str and load
df_county_ses_quintile_index = df_county_sespm25_index_quintile
df_county_ses_quintile_index['FIPS'] = df_county_ses_quintile_index['FIPS'].

# Ensure df_pm25_cmr is also a string
df_pm25_cmr = df_annualstate_county_pm25_cmr
df_pm25_cmr['FIPS'] = df_pm25_cmr['FIPS'].astype(str)

# Merge  on 'FIPS'
df_merged_state_county = pd.merge(df_pm25_cmr, df_county_ses_quintile_index,


# View merged DataFrame
df_merged_state_county.head()
```

Out[99]:

	indexes_x	FIPS	Year	PM2.5	CMR	fip_state_x	state_x	indexes_y	SE
0	20	1001	2009	6.402091	330.876172	1	AL	1	
1	21	1001	2010	6.942778	316.911479	1	AL	1	
2	41	1003	2009	5.419087	270.402216	1	AL	2	
3	42	1003	2010	5.837704	276.377191	1	AL	2	
4	62	1005	2009	5.840124	383.159080	1	AL	3	

In [100…
```
# Feature Engineering
# Drop only existing columns
df_merged_state_county = df_merged_state_county.drop(columns=['fip_state_y',

# Rename columns
df_merged_state_county = df_merged_state_county.rename(columns={ 'fip_state_
```

```
# View merged DataFrame
df_merged_state_county.head()
```

Out[100…

	FIPS	Year	PM2.5	CMR	fip	state	SES_index_1990	SES_index_2000
0	1001	2009	6.402091	330.876172	1	AL	-0.079387	-0.322846
1	1001	2010	6.942778	316.911479	1	AL	-0.079387	-0.322846
2	1003	2009	5.419087	270.402216	1	AL	-0.187240	-0.467794
3	1003	2010	5.837704	276.377191	1	AL	-0.187240	-0.467794
4	1005	2009	5.840124	383.159080	1	AL	1.279538	2.013751

In [101…

```
#df_annualstate_county_pm25_cmr.head()
```

In [102…

```
# Feature Engineering

df1 = df_acs_2009_2010_states

df2 = df_annualstate_county_pm25_cmr

# second dataset has state-level FIPS in a different column, rename it to 'f


df2.rename(columns={'fip_state': 'fip'}, inplace=True)

df_acs_pm25_cmr_ses_index_state_combined = pd.merge(df1, df2, how='inner', o

df_acs_pm25_cmr_ses_index_state_combined.rename(columns={'state_x': 'state'}

df_acs_pm25_cmr_ses_index_state_combined.drop(columns=['state_y'], inplace=T

df_acs_pm25_cmr_ses_index_state_combined.head(5)
```

Out[102…

	state	median_income	total_population_poverty	poverty_count	total_population_u
0	AL	40489	4588899	804683	
1	AL	40489	4588899	804683	
2	AL	40489	4588899	804683	
3	AL	40489	4588899	804683	
4	AL	40489	4588899	804683	

In [103…

```
# Feature engineering
# Merge on 'state' and 'YEAR' for alignment
df_cvd_htn_mort_combined = pd.merge(df_heart_dx_mort, df_htn_dx_mort, on=['s


# View merged DataFrame
#df_cvd_htn_mort_combined.head()
```

```
df_cvd_htn_mort_combined_reup = df_cvd_htn_mort_combined.rename(columns={'RA

df_cvd_htn_mort_combined_reup.head()

# Save as csv if needed
#df_cvd_htn_mort_combined_reup.to_csv('cvd_htn_mort_rate_combined_data.csv',
```

Out[103…

	YEAR	state	Cvdmortrate	Cvddeathcount	URL_cvdmort	H
0	2022	AL	234.2	14958	/nchs/pressroom/states/alabama/al.htm	
1	2022	AK	145.7	1013	/nchs/pressroom/states/alaska/ak.htm	
2	2022	AZ	148.5	14593	/nchs/pressroom/states/arizona/az.htm	
3	2022	AR	224.1	8664	/nchs/pressroom/states/arkansas/ar.htm	
4	2022	CA	142.4	66340	/nchs/pressroom/states/california/ca.htm	

In [104…

```
df_acs_pm25_cmr_ses_index_state_combined.drop(columns=['Year'], inplace=True
df_acs_pm25_cmr_ses_index_state_combined.head(5)
```

Out[104…

	state	median_income	total_population_poverty	poverty_count	total_population_u
0	AL	40489	4588899	804683	
1	AL	40489	4588899	804683	
2	AL	40489	4588899	804683	
3	AL	40489	4588899	804683	
4	AL	40489	4588899	804683	

Correlation Analysis

Plots and Correlation map: These visualizations illustrate the relationships between socioeconomic factor variables, pm2.5 and CMR.

In [106…

```
df_acs_pm25_cmr_ses_index_state_combinedCorr = df_acs_pm25_cmr_ses_index_sta
#df_acs_pm25_cmr_ses_index_state_combinedCorr #view output
```

In [107…

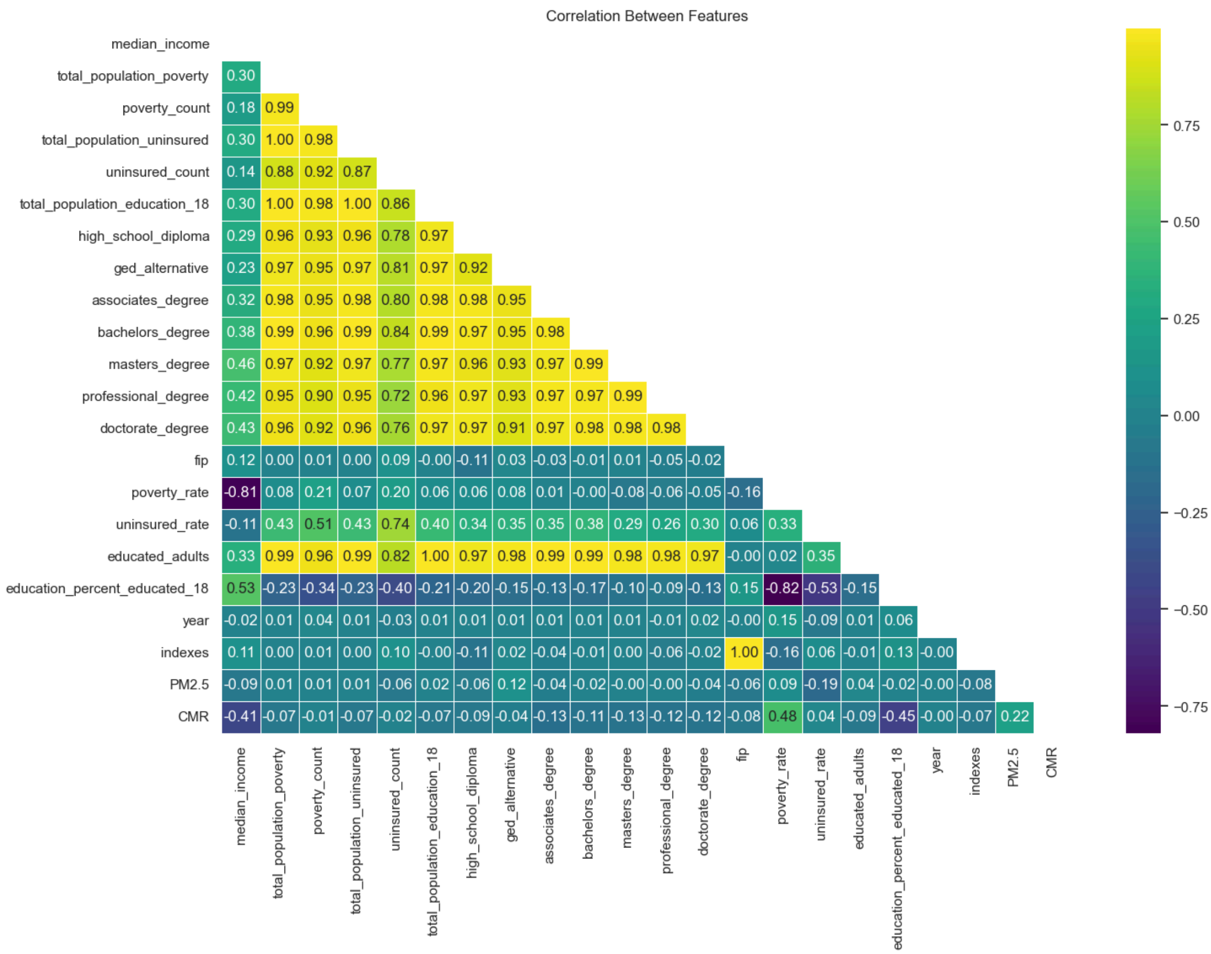
```
# Set seaborn themes
sns.set_theme(style='white')
sns.color_palette('viridis', as_cmap=True)
```

Out[107…

In [108…

```
# Create the plot
plt.figure(figsize=(15,10))
```

```
matrix = df_acs_pm25_cmr_ses_index_state_combinedCorr
mask = np.triu(np.ones_like(matrix, dtype=float))
sns.heatmap(df_acs_pm25_cmr_ses_index_state_combinedCorr,
            annot=True,
            linewidths=.5,
            cmap='viridis',
            fmt= '.2f',
            mask=mask)

# Specify the name of the plot
plt.title('Correlation Between Features')
plt.show()
```



In [109…
```
df_annualcounty_pm25_cmrCorr = df_annualcounty_pm25_cmr.corr(numeric_only=Tr
#df_annualcounty_pm25_cmrCorr #view output
```

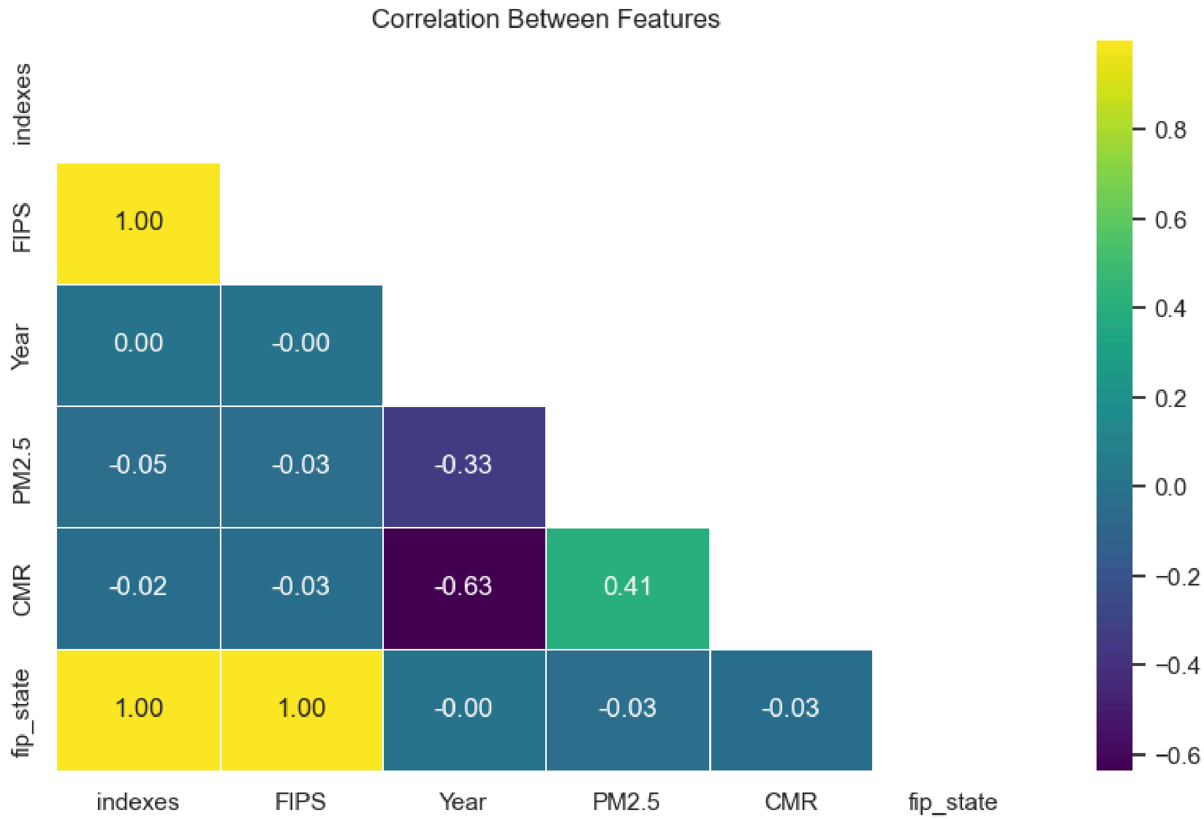
In [110…
```
# Set seaborn themes
sns.set_theme(style='white')
sns.color_palette('viridis', as_cmap=True)
```

Out[110…

viridis

In [111…

```
# Create the plot
plt.figure(figsize=(10,6))
matrix = df_annualcounty_pm25_cmrCorr
mask = np.triu(np.ones_like(matrix, dtype=float))
sns.heatmap(df_annualcounty_pm25_cmrCorr,
            annot=True,
            linewidths=.5,
            cmap='viridis',
            fmt= '.2f',
            mask=mask)

# Specify the name of the plot
plt.title('Correlation Between Features')
plt.show()
```



There is a weak positive correlation between PM2.5 levels and cardiovascular mortality risk (CMR), with a correlation coefficient (r) of 0.41. This suggests that higher levels of air pollution, specifically fine particulate matter (PM2.5), are modestly associated with increased cardiovascular mortality. Additionally, there is a moderately strong negative correlation between the year and CMR (r = –0.63), indicating a possible declining trend in cardiovascular mortality over time.

In [113…

```
df_acs_2009_2010_statesCorr = df_acs_2009_2010_states.corr(numeric_only=True
#df_acs_2009_2010_statesCorr #view output
```

In [114…

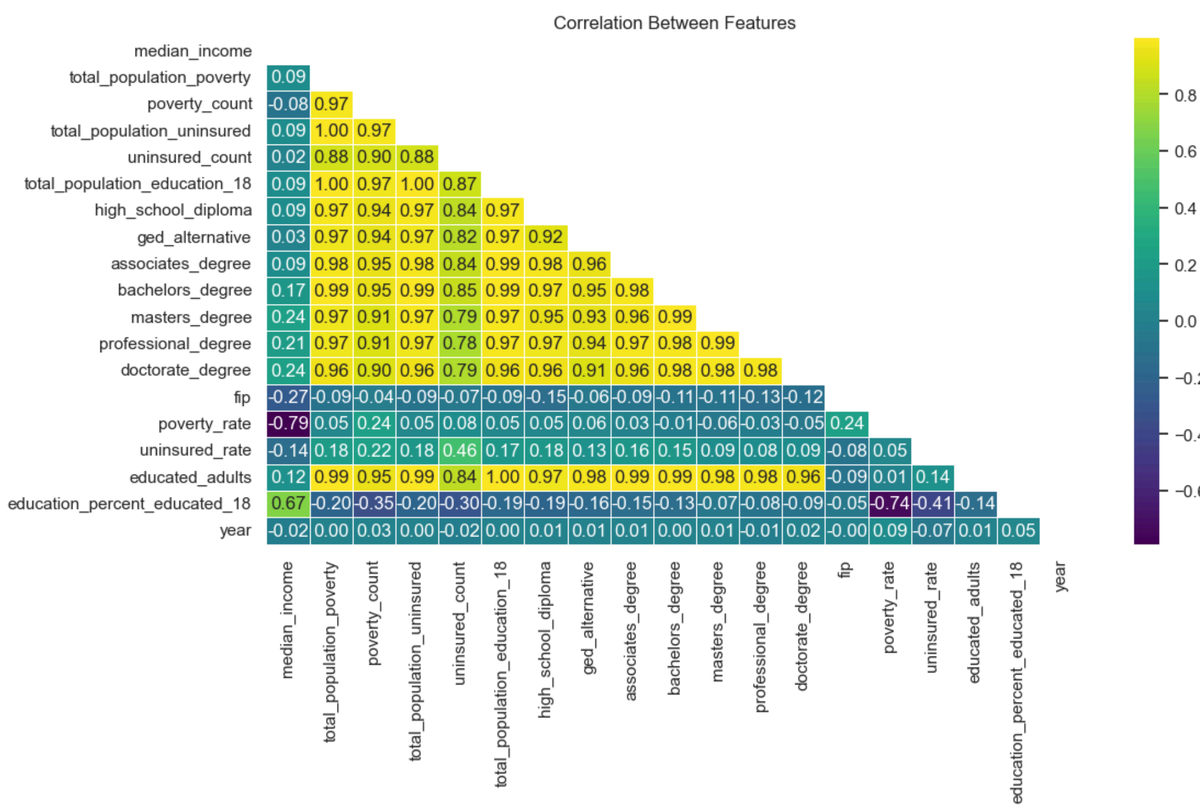
```
# Set seaborn themes
sns.set_theme(style='white')
```

```python
sns.color_palette('viridis', as_cmap=True)
```

Out[114…

In [115…

```python
# Create the plot
plt.figure(figsize=(12,6))
matrix = df_acs_2009_2010_statesCorr
mask = np.triu(np.ones_like(matrix, dtype=float))
sns.heatmap(df_acs_2009_2010_statesCorr,
            annot=True,
            linewidths=.5,
            cmap='viridis',
            fmt= '.2f',
            mask=mask)

# Specify the name of the plot
plt.title('Correlation Between Features')
plt.show()
```



This suggests a strong negative correlation between the poverty rate and median income, with a correlation coefficient (r) of –0.79, indicating that higher poverty levels may be associated with lower income. There is also a strong negative correlation between the percentage of the population with only a high school education and the poverty rate (r = –0.74), suggesting that higher education levels may be linked to lower poverty rates. In contrast, there is a weak positive correlation between the health uninsurance rate and the total population in poverty (r = 0.18), implying a slight increase in the uninsurance rate as the number of people in poverty rises. Additionally, a strong positive correlation exists between median income and the rate of higher education attainment (r = 0.67), suggesting that higher levels of education may be associated with higher income.

In [117…

```
df_merged_state_countyCorr = df_merged_state_county.corr(numeric_only=True)
#df_merged_state_countyCorr #view output
```

In [118…

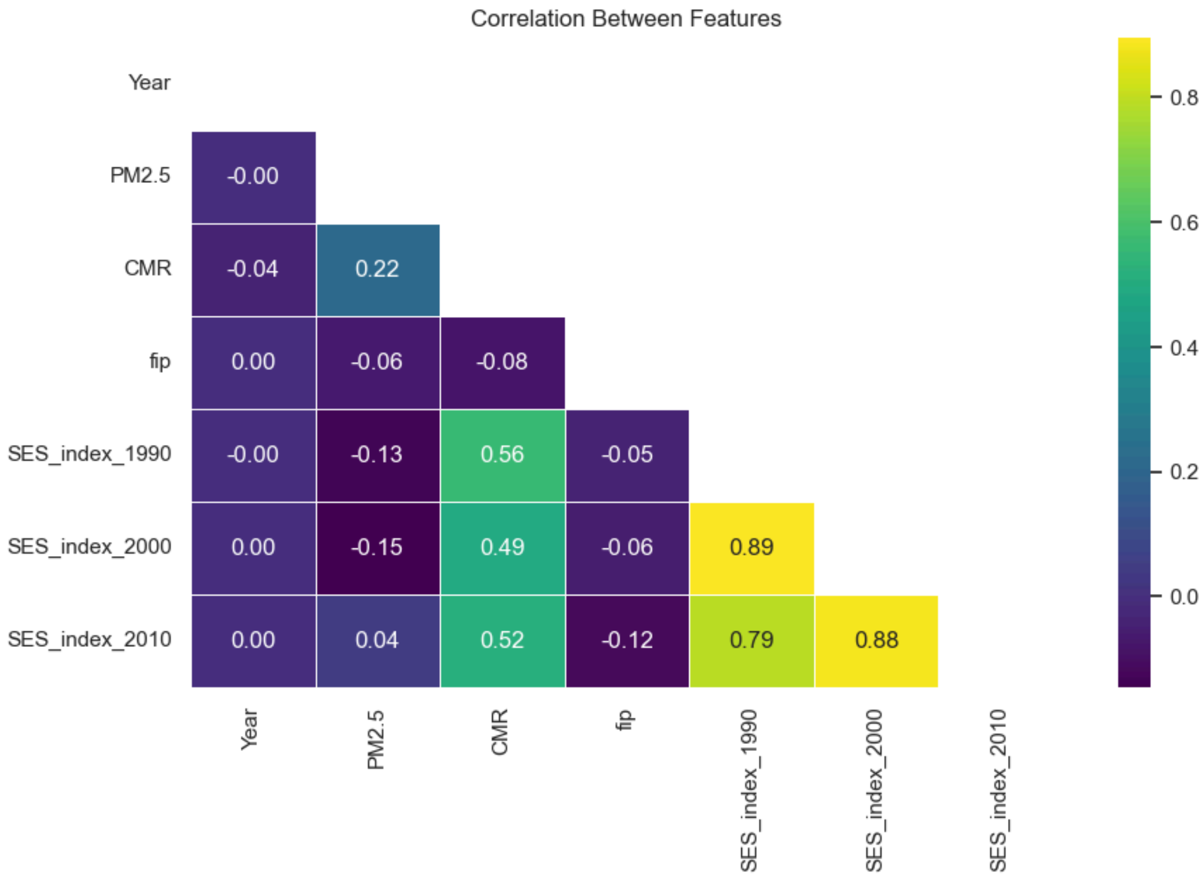
```
# Set seaborn themes
sns.set_theme(style='white')
sns.color_palette('viridis', as_cmap=True)
```

Out[118…

viridis

In [119…

```
# Create the plot
plt.figure(figsize=(10,6))
matrix = df_merged_state_countyCorr
mask = np.triu(np.ones_like(matrix, dtype=float))
sns.heatmap(df_merged_state_countyCorr,
           annot=True,
           linewidths=.5,
           cmap='viridis',
           fmt= '.2f',
           mask=mask)

# Specify the name of the plot
plt.title('Correlation Between Features')
plt.show()
```



Its worth noting that this heat map suggests from the correlation values socio-economic index and cardiomortality rate that the the cardiomortality rate increases as socioeconomic status index increases and this is in contrast to research that suggests that a higher socioeconomic status is associated with a lower CMR due to better health habits and healthcare access. Some possible reasons for this correlation may be due to confounding by region or other variables and could also be due SES indices capturing complexities such as counties with much older poupulation etc.

In [121…

```
# Hypothesis test
from scipy.stats import ttest_ind

# Hypothesis: States/Counties with higher PM2.5 levels have higher CMR
high_pm25 = df_merged_state_county[df_merged_state_county['PM2.5'] > df_merg
low_pm25 = df_merged_state_county[df_merged_state_county['PM2.5'] <= df_merg

# Perform t-test
t_stat, p_value = ttest_ind(high_pm25, low_pm25)
print(f"t-statistic: {t_stat}, p-value: {p_value}")
```

```
t-statistic: 7.1673660317328, p-value: 8.96756932006852e-13
```

There is a statistically significant difference in the mean CMR between states/counties with high PM2.5 levels and those with low PM2.5 levels.

Associations between socioeconomic factors (poverty, education, and health insurance) and cardiovascular mortality rates across some U.S. states

In [124…

```
#Correlation Analysis
df_acs_pm25_cmr_ses_index_state_combinedCorr = df_acs_pm25_cmr_ses_index_sta
#df_acs_pm25_cmr_ses_index_state_combinedCorr #view output
```

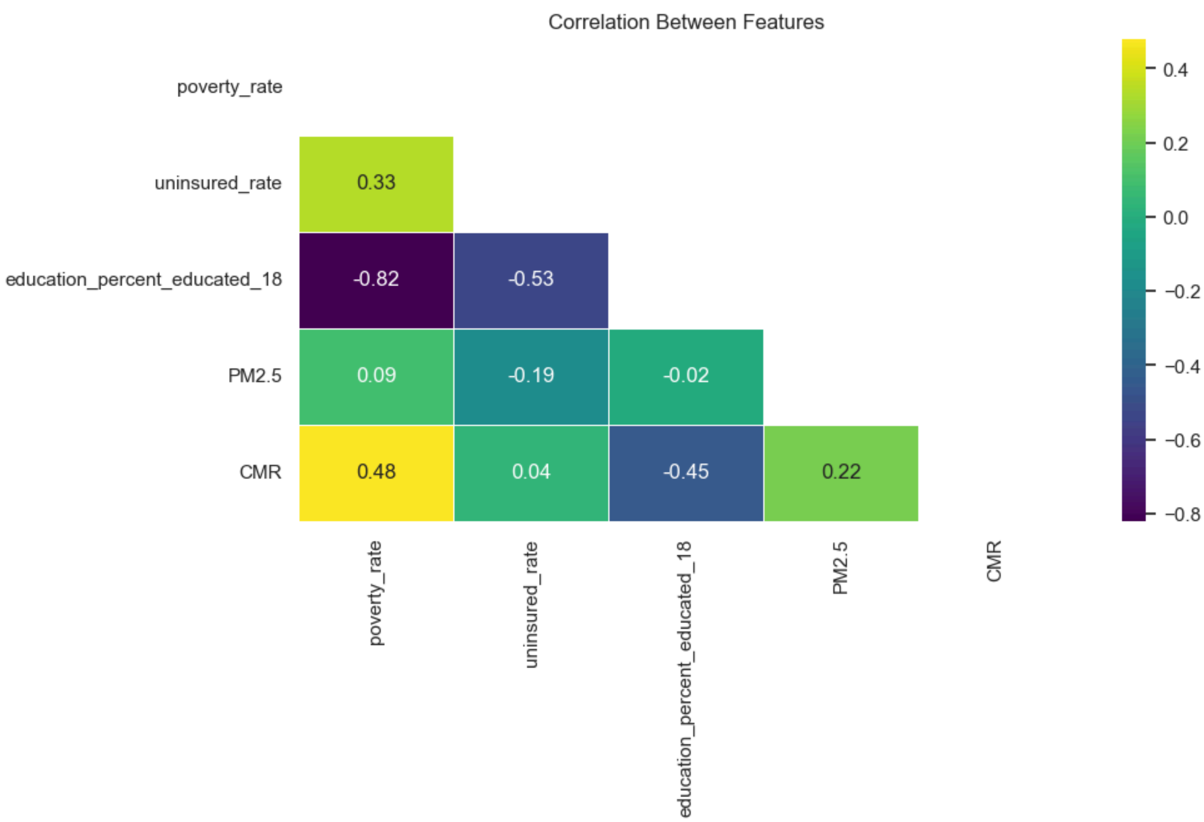
In [125…

```
# Set seaborn themes
sns.set_theme(style='white')
sns.color_palette('viridis', as_cmap=True)
```

Out[125…

In [126…

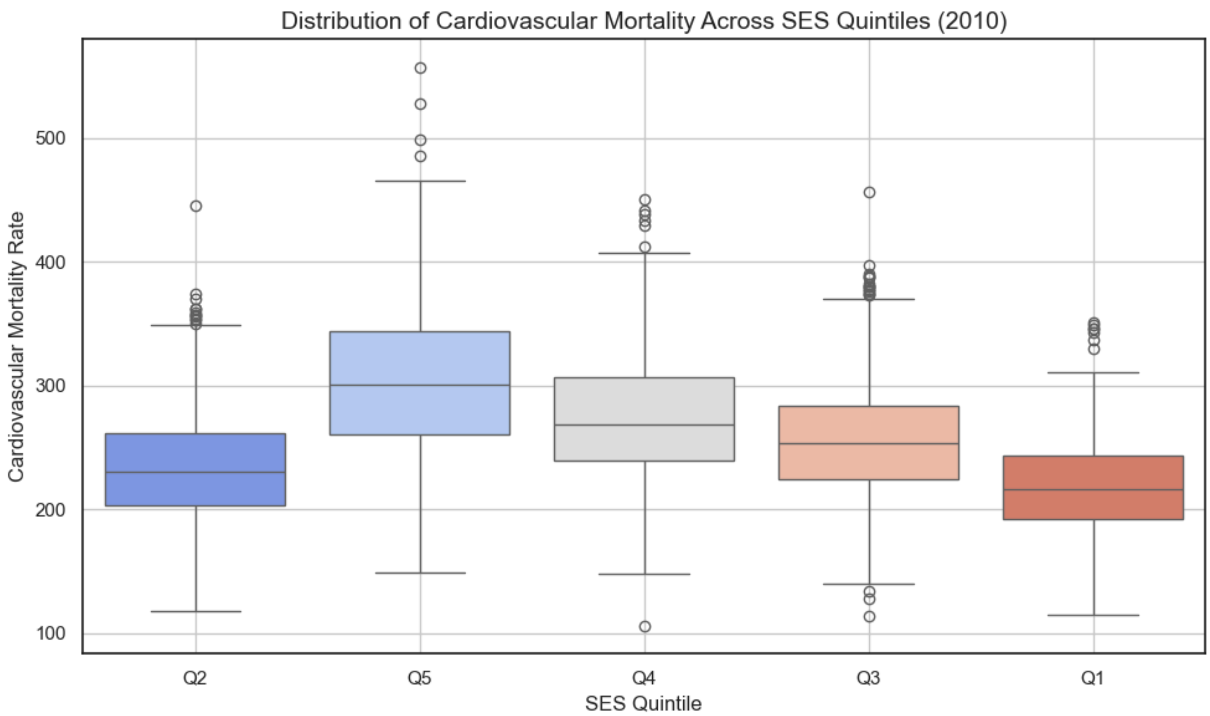
```
# Create the plot
plt.figure(figsize=(10,5))
matrix = df_acs_pm25_cmr_ses_index_state_combinedCorr
mask = np.triu(np.ones_like(matrix, dtype=float))
sns.heatmap(df_acs_pm25_cmr_ses_index_state_combinedCorr,
            annot=True,
            linewidths=.5,
            cmap='viridis',
            fmt= '.2f',
            mask=mask)

# Specify the name of the plot
plt.title('Correlation Between Features')
plt.show()
```



Boxplots on SES and CMR: They reveal systematic differences in CMR across socioeconomic groups.

This suggests a modestly positive correlation between the poverty rate,pm2.5 and cardiovascularmortality rate, with a correlation coefficient (r) of 0.48 and 0.22 indicating that increasing poverty levels and pm2.5 levels may be associated with higher CMR.

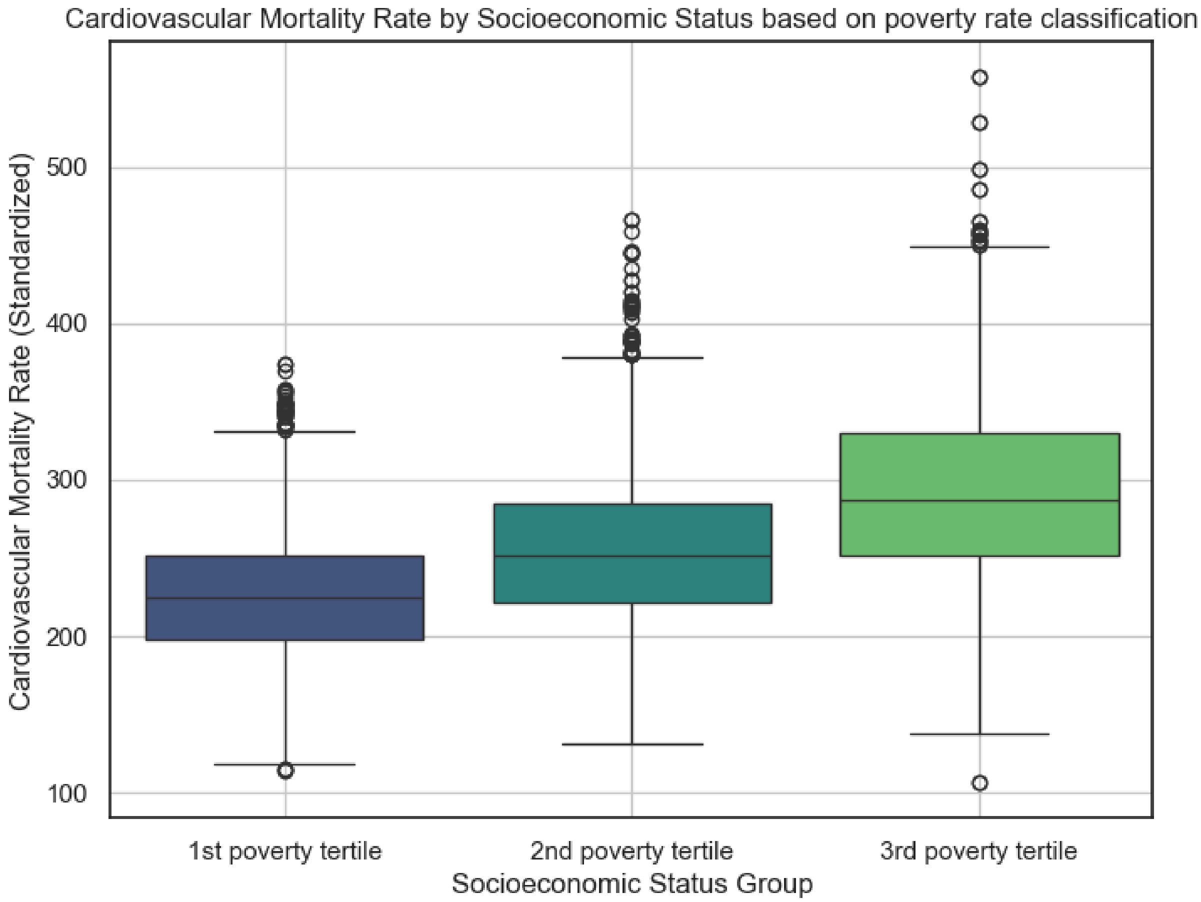
In [129…
```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df_merged_state_county, x='SES_quintile_2010', y='CMR', pal
plt.title('Distribution of Cardiovascular Mortality Across SES Quintiles (20
plt.xlabel('SES Quintile')
plt.ylabel('Cardiovascular Mortality Rate')
plt.grid(True)
plt.tight_layout()
plt.show()
```



The contrast noticed in the boxplots between the influence of social classification based on socioeconomic status on cardiovascular mortality and the influence of poverty levels classified into tertiles on cardiovascular mortality suggests that while socioeconomic status and poverty are related, their impacts on cardiovascular health may be distinct. Socioeconomic status likely captures broader factors, such as access to quality education, stable employment, and social support networks, whereas poverty levels focus more narrowly on income deprivation. Further statistical analysis is important to determine the significance of the observed differences.

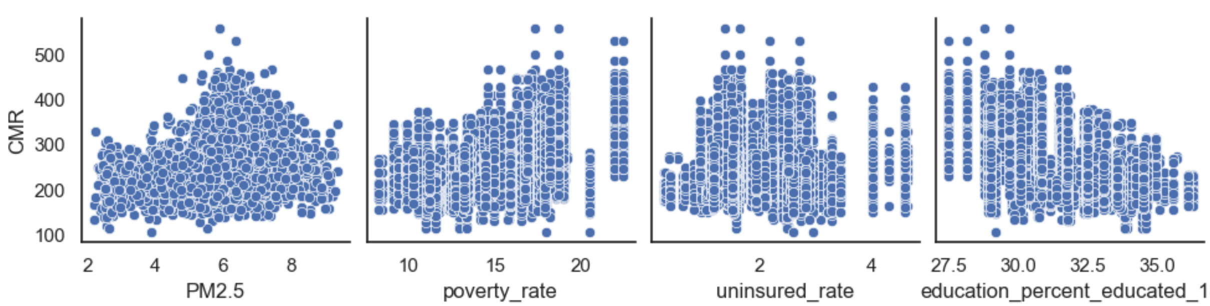
In [131…
```
# Categorize states into Low, Medium, High SES Groups
df_acs_pm25_cmr_ses_index_state_combined['SES_Group'] = pd.qcut(df_acs_pm25_

# Boxplot
plt.figure(figsize=(8,6))
sns.boxplot(data=df_acs_pm25_cmr_ses_index_state_combined, x="SES_Group", y=
```

```
plt.title("Cardiovascular Mortality Rate by Socioeconomic Status based on po
plt.xlabel("Socioeconomic Status Group")
plt.ylabel("Cardiovascular Mortality Rate (Standardized)")
plt.grid(True)
plt.show()
```



In [132…

```
sns.pairplot(
    df_acs_pm25_cmr_ses_index_state_combined,
    x_vars=["PM2.5", "poverty_rate", "uninsured_rate","education_percent_edu
    y_vars=["CMR"]
)



plt.show()
```



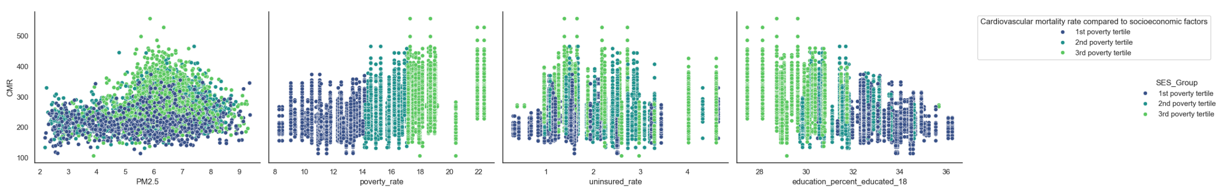
This pairplot provides a matrix of scatter plots, examining how different socioeconomic factors (poverty, education, insurance) relate to

Cardiomortality rate (CMR).

The pairwise relationships shows that higher pm2.5 rates may be associated with increased CMR and shows that lower education and higher poverty rates may be associated with increased CMR.

In [134…

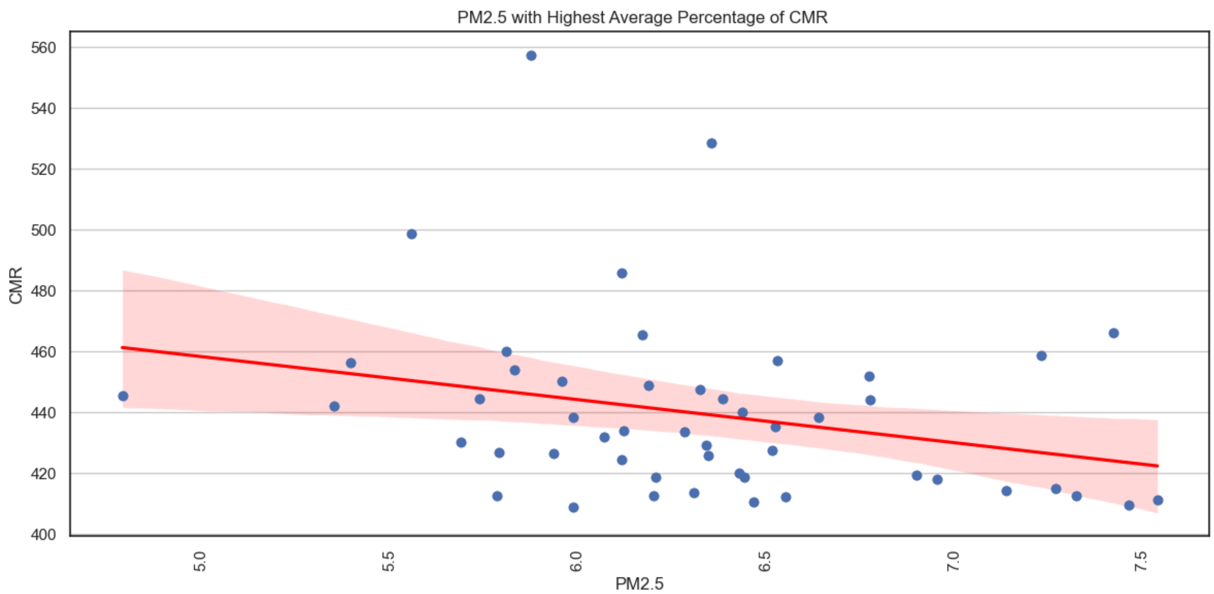
```
sns.pairplot(
    df_acs_pm25_cmr_ses_index_state_combined,
    x_vars=["PM2.5", "poverty_rate", "uninsured_rate","education_percent_edu
    y_vars=["CMR"],
    hue="SES_Group",
    palette="viridis" ,
    height=4,
    aspect=1.5
)

# Add a legend
plt.legend(title="Cardiovascular mortality rate compared to socioeconomic fa

# Adjust layout
plt.tight_layout()

# Show the plot
plt.show()
```



Scatter Plots of CMR in relation to PM2.5 and Socioeconomic Indicators: These plots demonstrate that environmental and social determinants impact health.

In [136…

```
# Group by variable1 and calculate the average percentage of variable2 for e
averageVariable1 = df_acs_pm25_cmr_ses_index_state_combined.groupby('PM2.5')

# Sort variable1 based on the highest average percentage of variable2
maxVariable1 = averageVariable1.sort_values(ascending=False).head(50)

# Create a scatter plot
plt.figure(figsize=(12, 6))
sns.regplot(x=maxVariable1.index, y=maxVariable1.values, scatter=True, line_
plt.scatter(maxVariable1.index, maxVariable1.values)
plt.xlabel('PM2.5')
plt.ylabel('CMR')
plt.title(' PM2.5 with Highest Average Percentage of CMR')
plt.xticks(rotation=90)
plt.grid(axis='y')
plt.tight_layout()
# Show the visualization
plt.show()
```

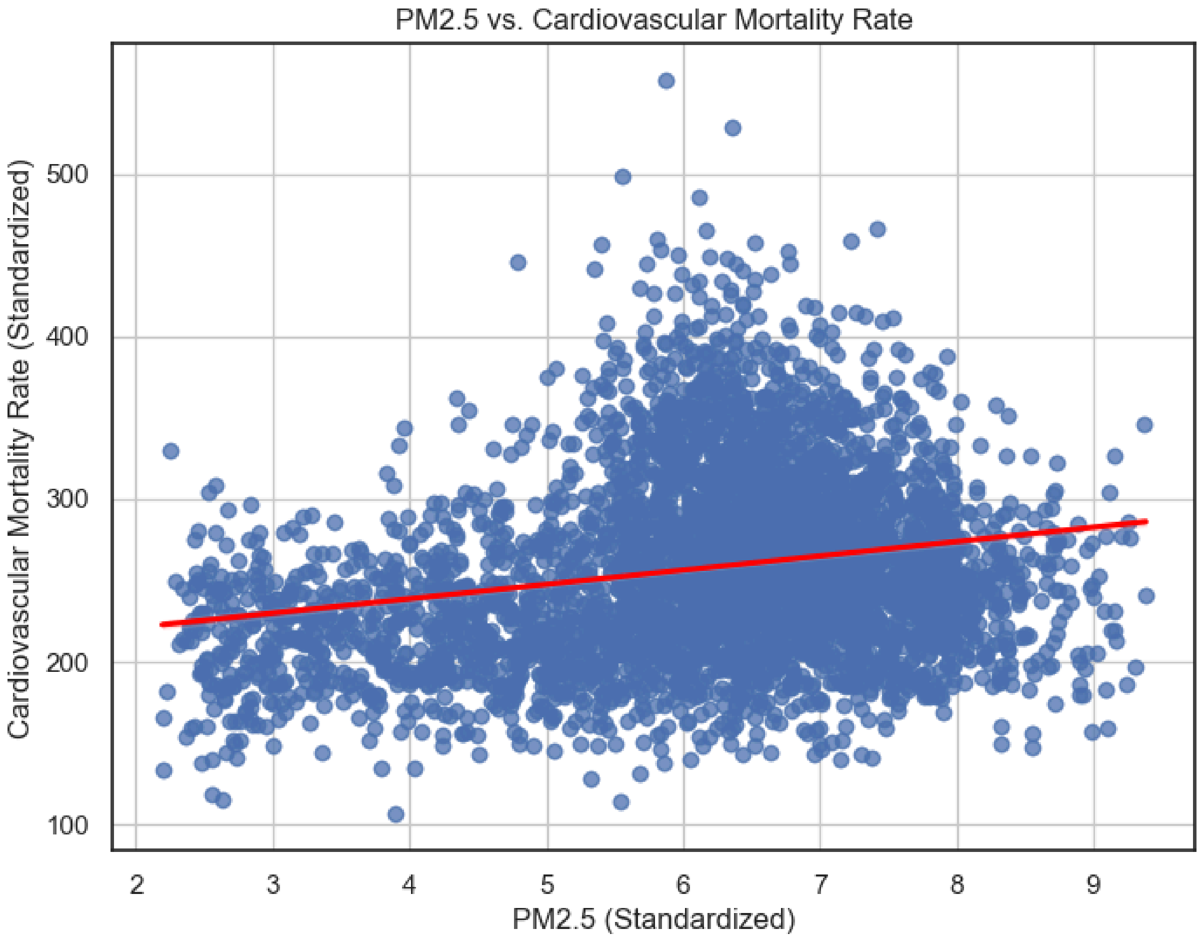


This plot explores the impact of air pollution (PM2.5) on cardiovascular mortality.

Higher PM2.5 levels appear to be linked to an increase in CMR when variables are standardized, reinforcing environmental concerns in cardiovascular health. But higher PM2.5 levels appear to be linked to a decrease in CMR when variables are averaged.

In [138…

```
# Scatter Plot: PM2.5 vs Cardiovascular Mortality Rate
plt.figure(figsize=(8,6))
sns.regplot(data=df_acs_pm25_cmr_ses_index_state_combined, x="PM2.5", y="CMR
plt.title("PM2.5 vs. Cardiovascular Mortality Rate")
plt.xlabel("PM2.5 (Standardized)")
plt.ylabel("Cardiovascular Mortality Rate (Standardized)")
plt.grid(True)
plt.show()
```

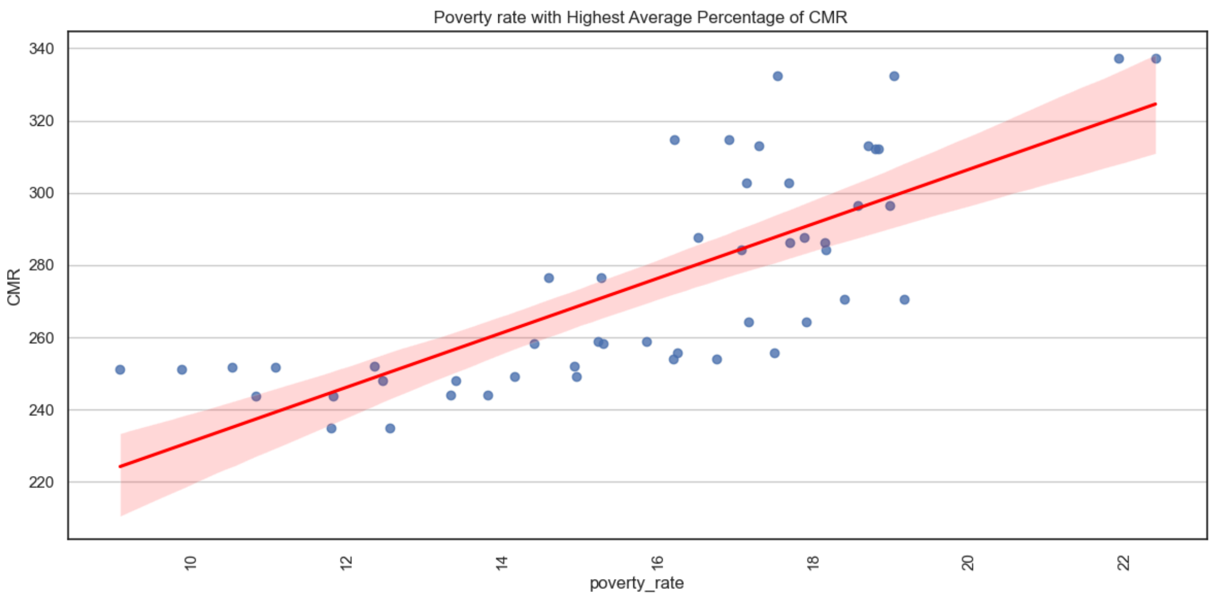


This scatter plot examines the correlation between the poverty rate and cardiovascular mortality rates.

The plot shows a positive correlation with standardized and non-standardized variables, indicating that states with higher poverty rates may have some influence on increased cardiovascular mortality rates.

In [140…

```
# Group by variable1 and calculate the average percentage of variable2 for e
averageVariable1 = df_acs_pm25_cmr_ses_index_state_combined.groupby('poverty

# Sort variable1 based on the highest average percentage of variable2
maxVariable1 = averageVariable1.sort_values(ascending=False).head(50)

# Create a scatter plot
plt.figure(figsize=(12, 6))
sns.regplot(x=maxVariable1.index, y=maxVariable1.values, scatter=True, line_
plt.xlabel('poverty_rate')
plt.ylabel('CMR')
plt.title(' Poverty rate with Highest Average Percentage of CMR')
plt.xticks(rotation=90)
plt.grid(axis='y')
plt.tight_layout()
# Show the visualization
plt.show()
```

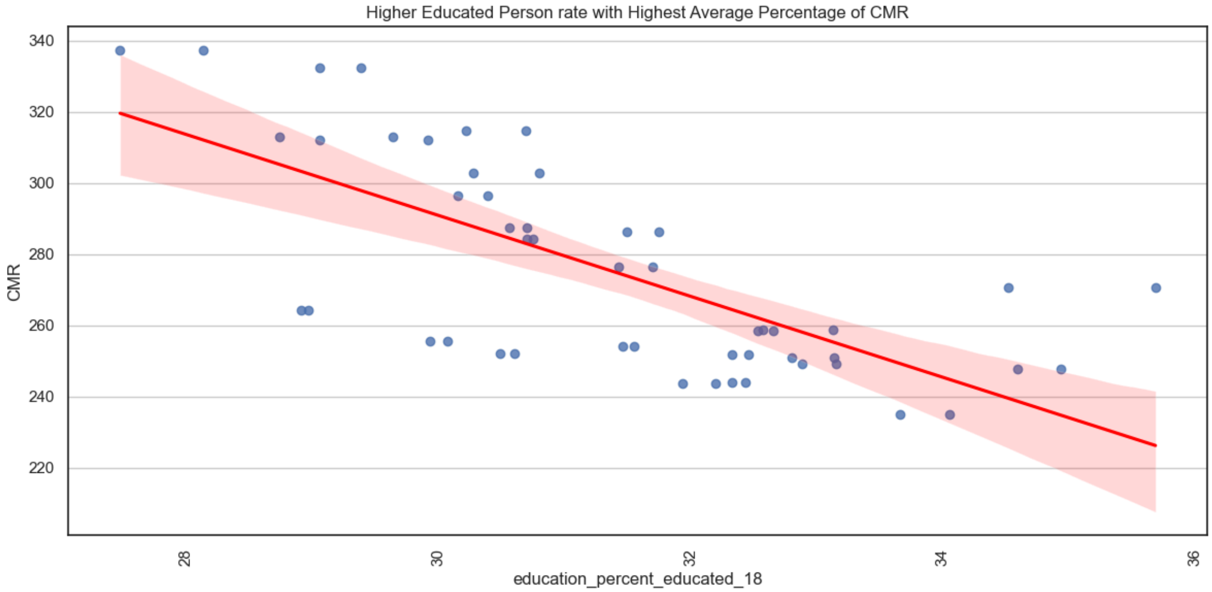


This scatter plot examines the correlation between the higher education rates and cardiovascular mortality rates.

The plot shows a potential negative correlation with standardized and non_standardized variables, indicating that states with higher educated citizen rates may have some influence on decreased cardiovascular mortality rates.

In [142…

```
# Group  by variable1 and calculate the average percentage of variable2 for
averageVariable1 = df_acs_pm25_cmr_ses_index_state_combined.groupby('educati

# Sort variable1 based on the highest average percentage of variable2
maxVariable1 = averageVariable1.sort_values(ascending=False).head(50)

# Create a scatter plot
plt.figure(figsize=(12, 6))
sns.regplot(x=maxVariable1.index, y=maxVariable1.values, scatter=True, line_
plt.xlabel('education_percent_educated_18')
plt.ylabel('CMR')
plt.title('Higher Educated Person rate with Highest Average Percentage of CM
plt.xticks(rotation=90)
plt.grid(axis='y')
plt.tight_layout()
# Show the visualization
plt.show()
```

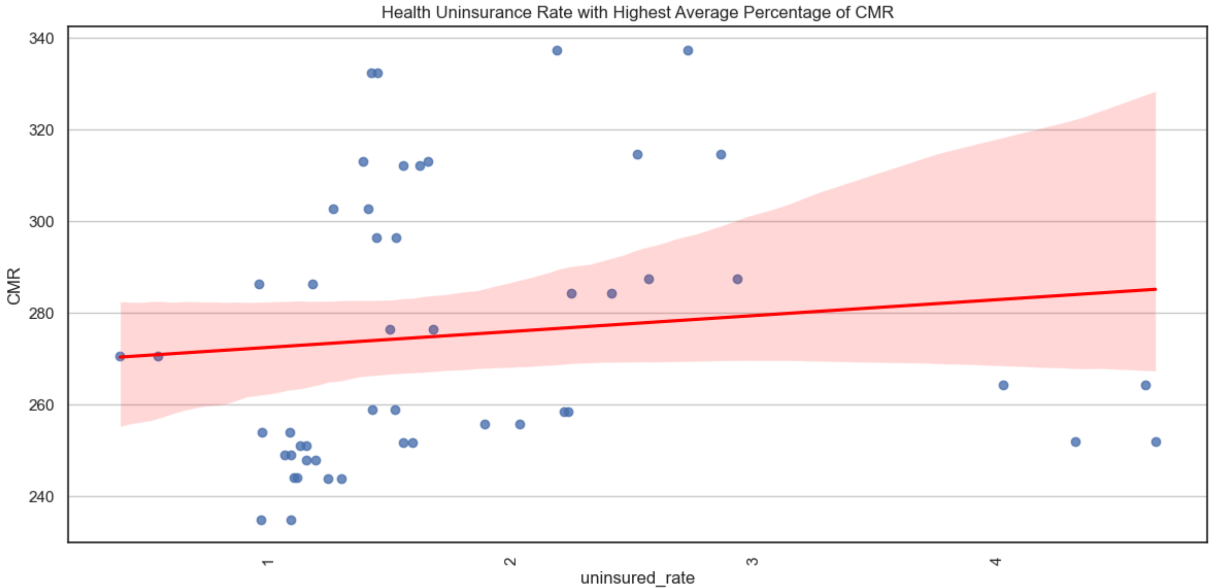


This scatter plot examines the correlation between the uninsured rate and cardiovascular mortality rates.

The plot shows a potential positive correlation, indicating that states with higher uninsured rates may have some influence on increased cardiovascular mortality rates.

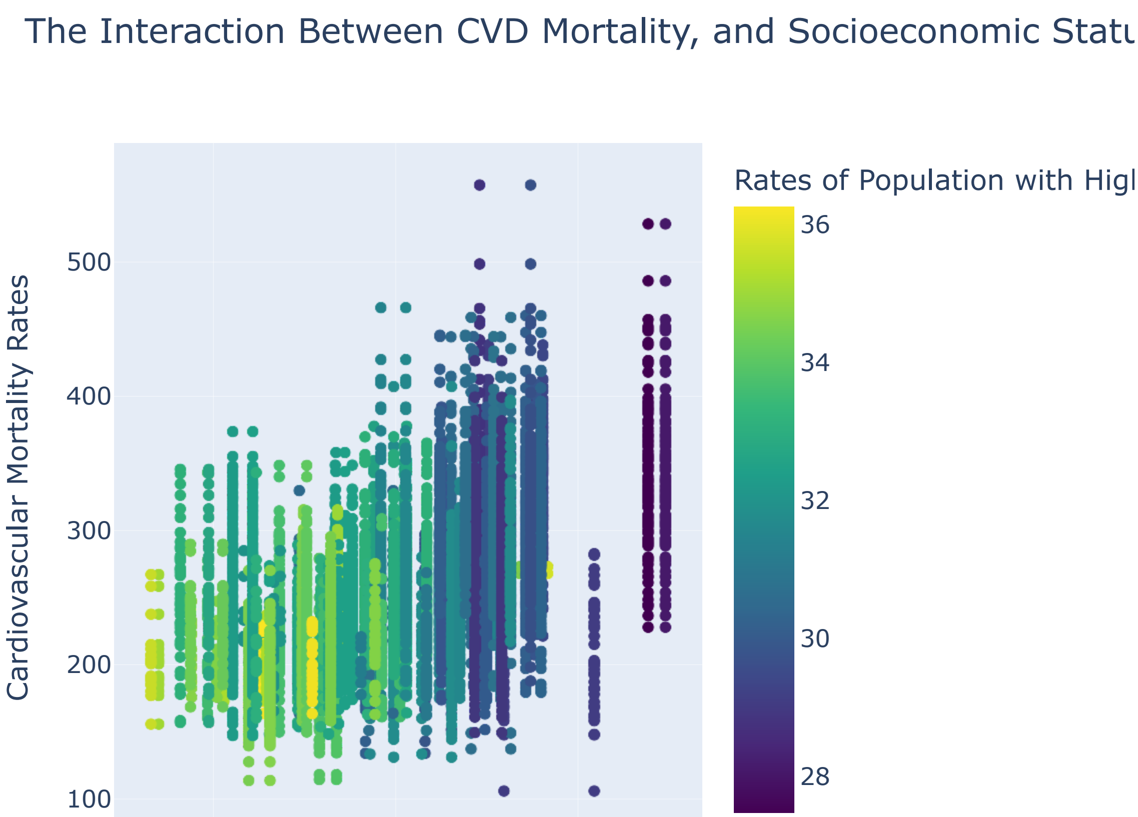
In [144…

```
#Group by variable1 and calculate the average percentage of variable2 for ea
averageVariable1 = df_acs_pm25_cmr_ses_index_state_combined.groupby('uninsur

# Sort variable1 based on the highest average percentage of variable2
maxVariable1 = averageVariable1.sort_values(ascending=False).head(50)

# Create a scatter plot
plt.figure(figsize=(12, 6))
sns.regplot(x=maxVariable1.index, y=maxVariable1.values, scatter=True, line_
plt.xlabel('uninsured_rate')
plt.ylabel('CMR')
plt.title('Health Uninsurance Rate with Highest Average Percentage of CMR')
plt.xticks(rotation=90)
plt.grid(axis='y')
plt.tight_layout()
# Show the visualization
plt.show()
```



In [145…

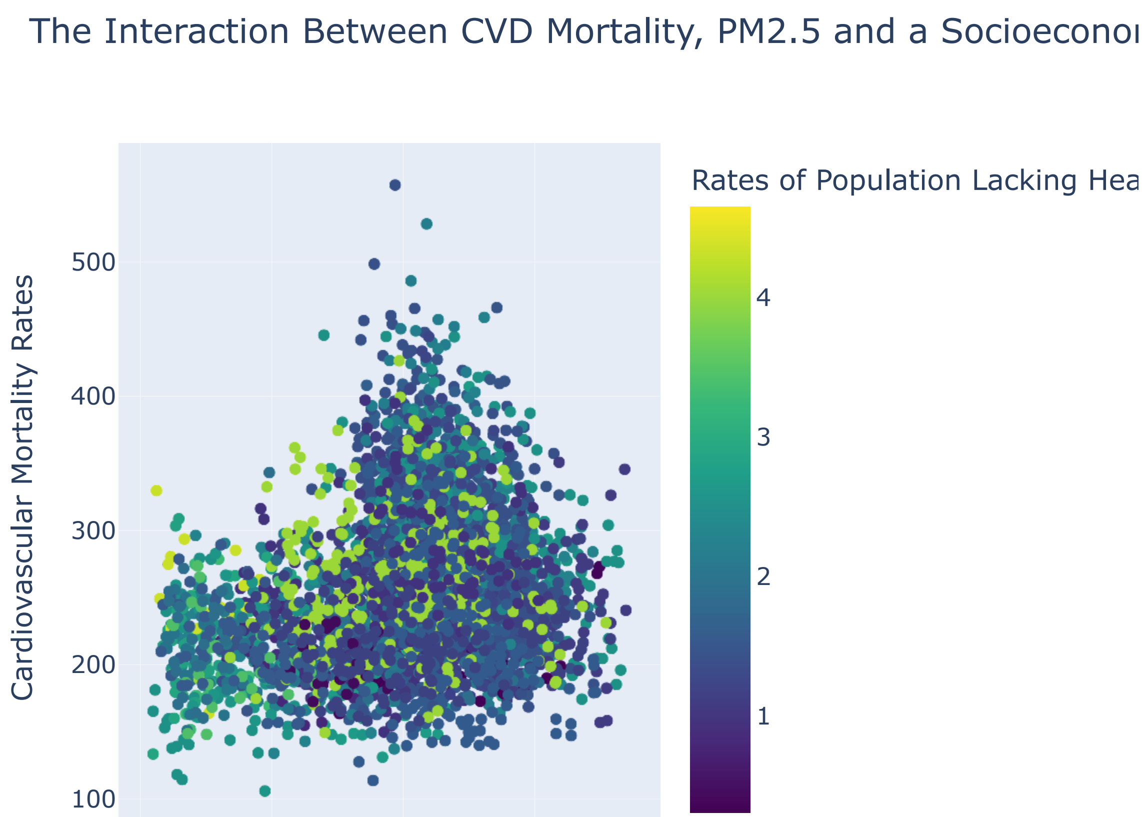
```
#  Plotly Scatter chart
import plotly.express as px

fig = px.scatter (df_acs_pm25_cmr_ses_index_state_combined,
x='poverty_rate',
y = 'CMR' ,
color = 'education_percent_educated_18',
title = 'The Interaction Between CVD Mortality, and Socioeconomic Status Fac
labels={
"poverty_rate": "Poverty rate",
"CMR": "Cardiovascular Mortality Rates",
"education_percent_educated_18": "Rates of Population with Higher Education
},
color_continuous_scale=px. colors. sequential.Viridis)
fig. show()
```



This visualization provides a matrix of plots, examining how different socioeconomic factors (Poverty, Higher education, Health insurance) relate to CMR. The relationships suggest that higher education and lower poverty rates may be associated with decreased CMR.

In [147…

```
#  Plotly Scatter chart
import plotly.express as px

fig = px.scatter (df_acs_pm25_cmr_ses_index_state_combined,
x='PM2.5',
y = 'CMR' ,
color = 'uninsured_rate',
title = 'The Interaction Between CVD Mortality, PM2.5 and a Socioeconomic St
labels={
"PM2.5": "Particulate Matter 2.5 levels",
"CMR": "Cardiovascular Mortality Rates",
"uninsured_rate": "Rates of Population Lacking Health Insurance "
},
color_continuous_scale=px. colors. sequential.Viridis)
fig. show()
```



This visualization provides a plot, examining how a socioeconomic factor(Health insurance) and PM2.5 relates to Cardiovascular Mortality. The relationships subtlely suggest that as PM2.5 Pollutant levels rise in combination with higher rates of lack of health insurance Cardiovascular Mortality may also rise.

How does hypertension prevalence impact cardiovascular mortality rates?

In [152…

```
df_cvd_htn_mort_combined_reup_clean=df_cvd_htn_mort_combined_reup.drop(colum

df_cvd_htn_mort_combined_reup_clean.tail()
```

Out[152…

	YEAR	state	Cvdmortrate	Cvddeathcount	Htndxdeathrate	Htndxdeathcount
496	2005	VA	203.0	14192	7.9	549
497	2005	WA	180.5	10985	7.5	452
498	2005	WV	253.6	5538	11.6	253
499	2005	WI	190.6	11842	7.1	451
500	2005	WY	188.3	952	3.9	20

In [153…

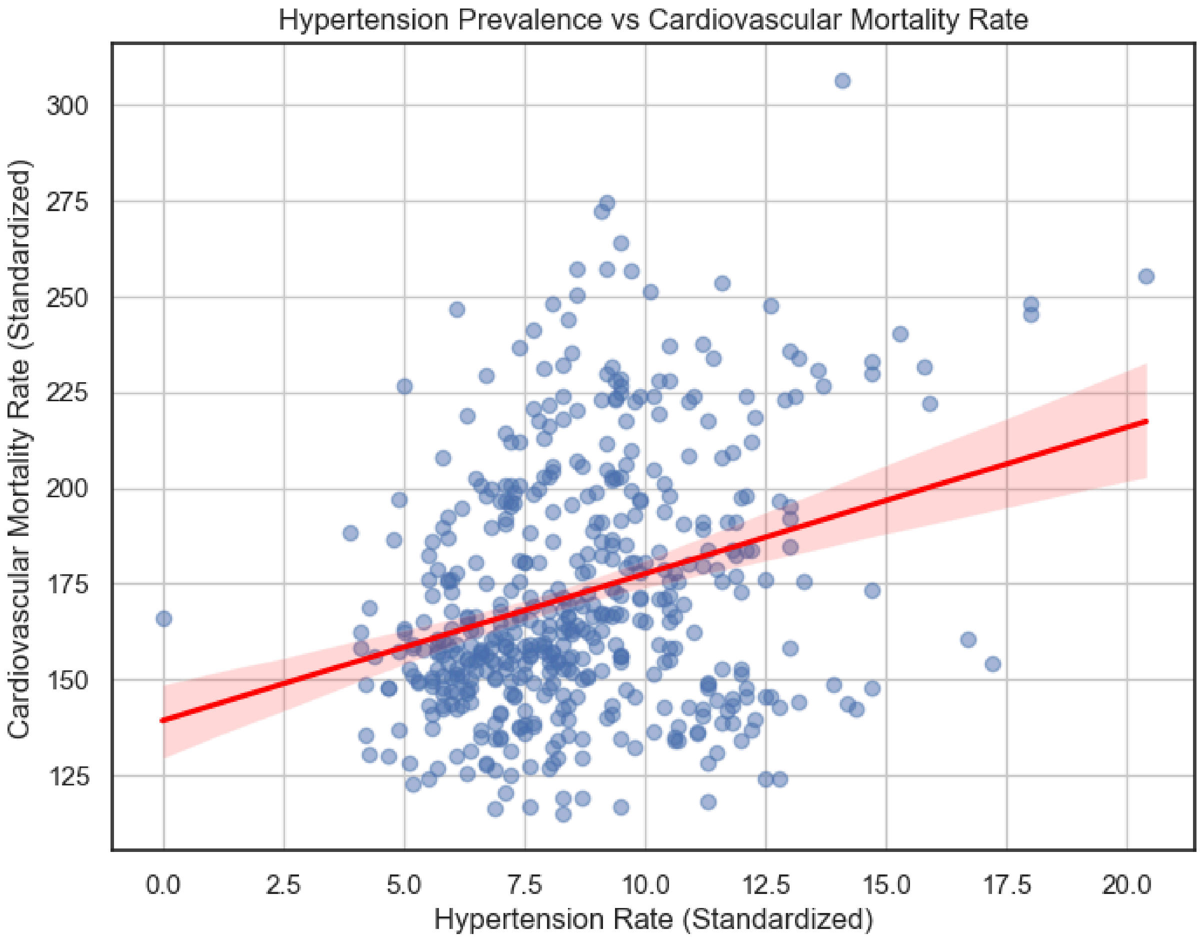
```
#Correlation Analysis
df_cvd_htn_mort_combined_reup_cleanCorr = df_cvd_htn_mort_combined_reup_clea
df_cvd_htn_mort_combined_reup_cleanCorr

from scipy.stats import pearsonr
#Pearson correlation and p-value
corr_coef, p_value = pearsonr(df_cvd_htn_mort_combined_reup_clean['Cvdmortra

corr_coef,p_value
```

Out[153… (0.2952786039775407, 1.5444236806717292e-11)

This suggests a weak positive relationship between Particulate matter 2.5ug levels and Cardiomortality rate with higher PM2.5 levels being associated with higher cardiovascular mortality rates though other factors may also have a strong influence too on CMR.

In [155…

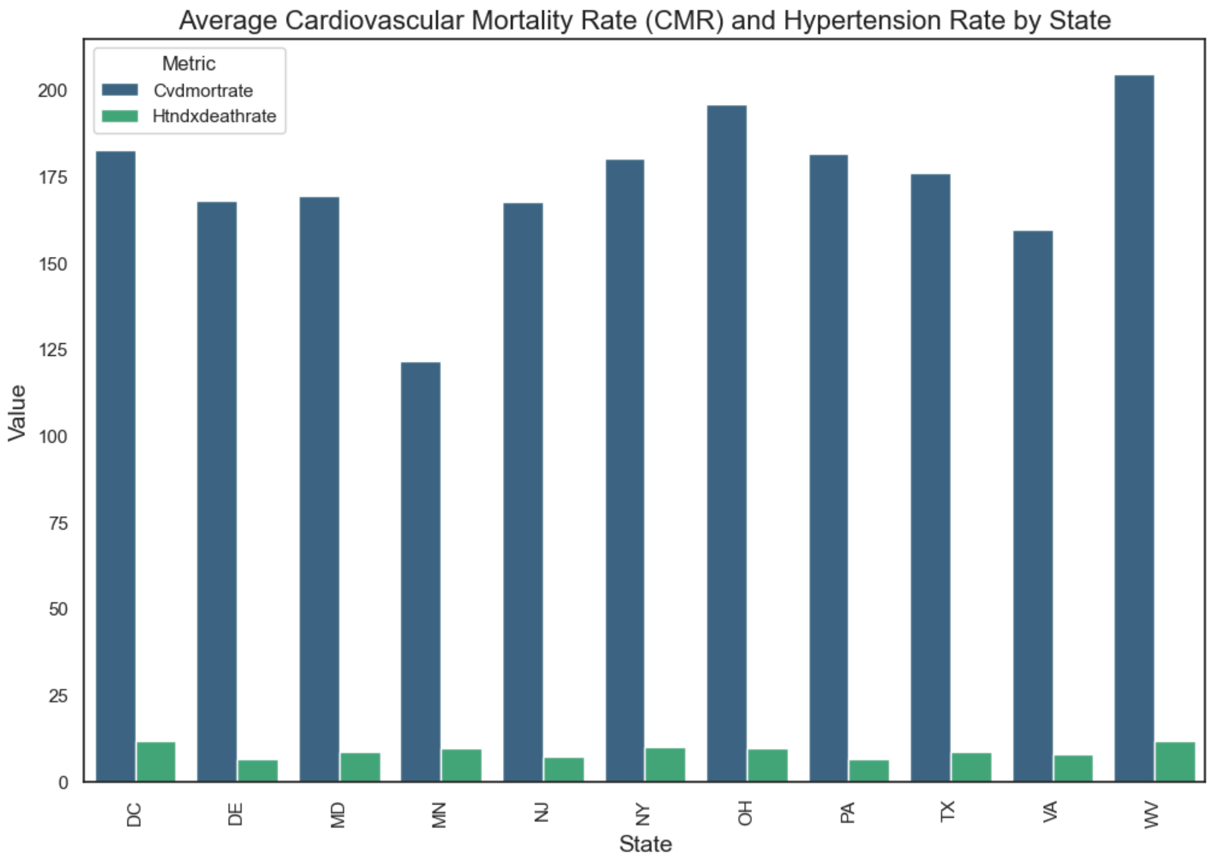
```
# Scatter Plot: Hypertension Prevalence vs Cardiovascular Mortality Rate
plt.figure(figsize=(8,6))
sns.regplot(data=df_cvd_htn_mort_combined_reup_clean, x="Htndxdeathrate", y=
plt.title("Hypertension Prevalence vs Cardiovascular Mortality Rate")
plt.xlabel("Hypertension Rate (Standardized)")
plt.ylabel("Cardiovascular Mortality Rate (Standardized)")
plt.grid(True)
plt.show()
```



In [156…

```
# States of interest
states = ['DC', 'MD', 'VA', 'WV', 'PA', 'DE', 'MN', 'NY', 'NJ', 'TX', 'OH']
df_filtered = df_cvd_htn_mort_combined_reup[df_cvd_htn_mort_combined_reup['s

# Group by state and calculate mean CMR and Hypertension Rate
df_grouped = df_filtered.groupby('state')[['Cvdmortrate', 'Htndxdeathrate']]

# Join the grouped dataframe for plotting
df_melted = df_grouped.melt(
    id_vars=['state'],
    value_vars=['Cvdmortrate', 'Htndxdeathrate'],
    var_name='Metric',
    value_name='Value'
)

# Create a side-by-side bar plot
plt.figure(figsize=(12, 8))
sns.barplot(
    x='state',
    y='Value',
    hue='Metric',
    data=df_melted,
    palette='viridis'
)
plt.title('Average Cardiovascular Mortality Rate (CMR) and Hypertension Rate
plt.xlabel('State', fontsize=14)
plt.ylabel('Value', fontsize=14)
```

```
plt.xticks(rotation=90)
plt.legend(title='Metric')
plt.show()
```



The visualizations in the figure above are expected for Cardiovascular disease mortality and hypertensive disease rates considering that hypertension can is a high risk factor for CMR and Death but Cardiovascular disease mortality can be due to a vast number of conditions.

Regression Analysis: The regression results quantify how various factors contribute to CMR.

In [159…

```
# independent and dependent variables
X = df_cvd_htn_mort_combined_reup_clean[['Htndxdeathrate']]
y = df_cvd_htn_mort_combined_reup_clean['Cvdmortrate']

import statsmodels.api as sm
# intercept
X = sm.add_constant(X)

# Fit the regression model
model = sm.OLS(y, X).fit()

# Display summary statistics
model.summary()
```

Out[159…

OLS Regression Results

Dep. Variable:	Cvdmortrate	**R-squared:**	0.087
Model:	OLS	**Adj. R-squared:**	0.085
Method:	Least Squares	**F-statistic:**	47.66
Date:	Fri, 18 Apr 2025	**Prob (F-statistic):**	1.54e-11
Time:	18:17:29	**Log-Likelihood:**	-2434.0
No. Observations:	501	**AIC:**	4872.
Df Residuals:	499	**BIC:**	4880.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	139.2546	4.984	27.940	0.000	129.462	149.047
Htndxdeathrate	3.8284	0.555	6.904	0.000	2.739	4.918

Omnibus:	30.390	**Durbin-Watson:**	1.994
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	34.272
Skew:	0.629	**Prob(JB):**	3.61e-08
Kurtosis:	3.248	**Cond. No.**	32.5

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

This model shows a statistically significant relationship between Hypertension-related death rate and Cardiovascular mortality rate. The positive and significant coefficient for Hypertension-related death rates suggests that higher hypertension-related death rates are associated with higher cardiovascular mortality rates and this gives some insight on the impact of hypertension prevalence on cardiovascular mortality rates albeit the low r-squared value (0.087) indicates that while hypertension-related death rates are significant, they explain only a small portion (8.7%) of the variation in cardiovascular mortality rates.

This could mean that other factors like PM2.5 levels, socioeconomic factors are important and should be included as strong influences.

The visualizations in this paper effectively reflect the relationship between cardiovascular mortality rates (CMR), air pollution (PM2.5), and socioeconomic factors such as poverty, education, and healthcare access(health uninsurance rate). The correlation map and regression analysis confirm that both environmental and social determinants significantly contribute to variations in CMR across different U.S. states. Higher PM2.5 exposure is associated with increased cardiovascular mortality, reinforcing concerns about air pollution's impact on heart disease. Lower socioeconomic status (SES) groups experience higher CMR, highlighting the role of poverty and education disparities in cardiovascular health.

In [162…

```
# X and Y variables
X_variable = 'CMR'
y_variables = ['PM2.5']

# Add a intercept to the independent variables
X = sm.add_constant(df_acs_pm25_cmr_ses_index_state_combined[y_variables])
y = df_acs_pm25_cmr_ses_index_state_combined[X_variable]

# Fit the OLS model
model = sm.OLS(y, X).fit()

# Print the model summary
print(model.summary())
```

```
                            OLS Regression Results                            
==============================================================================
==
Dep. Variable:                    CMR   R-squared:                       0.0
47
Model:                            OLS   Adj. R-squared:                  0.0
47
Method:                 Least Squares   F-statistic:                     42
2.0
Date:                Fri, 18 Apr 2025   Prob (F-statistic):           1.47e-
91
Time:                        18:17:29   Log-Likelihood:                -4632
4.
No. Observations:                8528   AIC:                         9.265e+
04
Df Residuals:                    8526   BIC:                         9.267e+
04
Df Model:                           1                                         
Covariance Type:            nonrobust                                         
==============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
const        203.2342      2.714     74.888      0.000     197.914     208.5
54
PM2.5          8.8104      0.429     20.541      0.000       7.970       9.6
51
==============================================================================
==
Omnibus:                      690.091   Durbin-Watson:                   0.8
32
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              908.6
27
Skew:                           0.704   Prob(JB):                    4.95e-1
98
Kurtosis:                       3.759   Cond. No.                          2
9.3
==============================================================================
==

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
```

In [163…

```
# X and Y variables
X_variable = 'CMR'
y_variables = ["uninsured_rate",'PM2.5']

# Add a intercept to the independent variables
X = sm.add_constant(df_acs_pm25_cmr_ses_index_state_combined[y_variables])
y = df_acs_pm25_cmr_ses_index_state_combined[X_variable]

# Fit the OLS model
model = sm.OLS(y, X).fit()
```

```python
# Print the model summary
print(model.summary())
```

```
                            OLS Regression Results                            
==============================================================================
==
Dep. Variable:                    CMR   R-squared:                       0.0
54
Model:                            OLS   Adj. R-squared:                  0.0
54
Method:                 Least Squares   F-statistic:                     24
2.2
Date:                Fri, 18 Apr 2025   Prob (F-statistic):          4.74e-1
03
Time:                        18:17:29   Log-Likelihood:                -4629
4.
No. Observations:                8528   AIC:                         9.259e+
04
Df Residuals:                    8525   BIC:                         9.262e+
04
Df Model:                           2                                         
Covariance Type:            nonrobust                                         
==============================================================================
======
                     coef    std err          t      P>|t|      [0.025      
0.975]
------------------------------------------------------------------------------
------
const            189.3197      3.250     58.255      0.000     182.949     1
95.690
uninsured_rate     5.1492      0.667      7.722      0.000       3.842       
6.456
PM2.5              9.4450      0.435     21.699      0.000       8.592      
10.298
==============================================================================
==
Omnibus:                      703.300   Durbin-Watson:                   0.8
43
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              937.8
13
Skew:                           0.707   Prob(JB):                    2.27e-2
04
Kurtosis:                       3.801   Cond. No.                        3
6.8
==============================================================================
==

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
```

In [164…

```python
# X and Y variables
X_variable = 'CMR'
y_variables = ['poverty_rate', 'PM2.5']
```

```
# Add a intercept to the independent variables
X = sm.add_constant(df_acs_pm25_cmr_ses_index_state_combined[y_variables])
y = df_acs_pm25_cmr_ses_index_state_combined[X_variable]

# Fit the OLS model
model = sm.OLS(y, X).fit()

# Print the model summary
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
==
Dep. Variable:                    CMR   R-squared:                       0.2
62
Model:                            OLS   Adj. R-squared:                  0.2
62
Method:                 Least Squares   F-statistic:                     151
1.
Date:                Fri, 18 Apr 2025   Prob (F-statistic):               0.
00
Time:                        18:17:30   Log-Likelihood:                -4523
6.
No. Observations:                8528   AIC:                         9.048e+
04
Df Residuals:                    8525   BIC:                         9.050e+
04
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
====
                   coef    std err          t      P>|t|      [0.025      0.
975]
------------------------------------------------------------------------------
----
const           69.7983      3.591     19.439      0.000      62.760      7
6.837
poverty_rate     9.3889      0.189     49.780      0.000       9.019
9.759
PM2.5            7.1237      0.379     18.792      0.000       6.381
7.867
==============================================================================
==
Omnibus:                      487.993   Durbin-Watson:                   1.0
59
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              645.3
38
Skew:                           0.540   Prob(JB):                    7.36e-1
41
Kurtosis:                       3.807   Cond. No.                         11
4.
==============================================================================
==

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
```

In [165…

```
# X and Y variables
X_variable = 'CMR'
y_variables = ['poverty_rate', "uninsured_rate", 'PM2.5']

# Add a intercept to the independent variables
X = sm.add_constant(df_acs_pm25_cmr_ses_index_state_combined[y_variables])
y = df_acs_pm25_cmr_ses_index_state_combined[X_variable]
```

```
# Fit the OLS model
model = sm.OLS(y, X).fit()

# Print the model summary
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
==
Dep. Variable:                    CMR   R-squared:                       0.2
70
Model:                            OLS   Adj. R-squared:                  0.2
70
Method:                 Least Squares   F-statistic:                     105
1.
Date:                Fri, 18 Apr 2025   Prob (F-statistic):               0.
00
Time:                        18:17:30   Log-Likelihood:                -4518
8.
No. Observations:                8528   AIC:                         9.038e+
04
Df Residuals:                    8524   BIC:                         9.041e+
04
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
======
                     coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
------
const             76.3900      3.633     21.026      0.000      69.268
83.512
poverty_rate      10.0973      0.201     50.250      0.000       9.703
10.491
uninsured_rate    -6.1647      0.627     -9.824      0.000      -7.395
-4.935
PM2.5              6.2367      0.388     16.089      0.000       5.477
6.997
==============================================================================
==
Omnibus:                      510.461   Durbin-Watson:                   1.0
65
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              669.6
38
Skew:                           0.561   Prob(JB):                    3.89e-1
46
Kurtosis:                       3.790   Cond. No.                         11
7.
==============================================================================
==

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
```

In [166…

```
# X and Y variables
X_variable = 'CMR'
y_variables = ['poverty_rate', 'education_percent_educated_18', "uninsured_r

# Add a intercept to the independent variables
X = sm.add_constant(df_acs_pm25_cmr_ses_index_state_combined[y_variables])
y = df_acs_pm25_cmr_ses_index_state_combined[X_variable]

# Fit the OLS model
model = sm.OLS(y, X).fit()

# Print the model summary
print(model.summary())
```

```
                               OLS Regression Results
==============================================================================
==
Dep. Variable:                    CMR   R-squared:                       0.3
03
Model:                            OLS   Adj. R-squared:                  0.3
03
Method:                 Least Squares   F-statistic:                     92
6.8
Date:                Fri, 18 Apr 2025   Prob (F-statistic):               0.
00
Time:                        18:17:30   Log-Likelihood:               -4499
0.
No. Observations:                8528   AIC:                       8.999e+
04
Df Residuals:                    8523   BIC:                       9.003e+
04
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
=====================
                                     coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
---------------------
const                            512.5388     21.964     23.336      0.000
469.485     555.593
poverty_rate                       4.8573      0.326     14.894      0.000
4.218       5.497
education_percent_educated_18   -10.8550      0.539    -20.122      0.000
-11.912      -9.798
uninsured_rate                   -12.5539      0.690    -18.182      0.000
-13.907     -11.200
PM2.5                              6.1614      0.379     16.267      0.000
5.419       6.904
==============================================================================
==
Omnibus:                      395.441   Durbin-Watson:                   1.1
11
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              508.5
45
Skew:                           0.475   Prob(JB):                    3.72e-1
11
Kurtosis:                       3.727   Cond. No.                     1.53e+
03
==============================================================================
==

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
[2] The condition number is large, 1.53e+03. This might indicate that there
are
strong multicollinearity or other numerical problems.
```

Particulate matter 2.5 consistently shows a significant positive association with cardiovascular mortality rate across all models. While socioeconomic factors are important predictors of cardiovascular mortality rate, poverty rate and education level, appear to have a substantial impact. The models explanatory power varies, with the model including poverty rate and PM2.5 having the highest R-squared in addition to a statistically significant relationships between all three independent variables and cardiovascular mortality rate. While higher poverty rates and pm2.5 levels are associated with higher cardiovascular mortality rate, higher education levels are associated with lower cardiovascular mortality rate with the model accounting for 30.3% of the variance in cardiovascular mortality rate. The model also presents possible multi-collinearity issues probable due to inter-relationship between the variables.