Computation and Visualization

Group 8

Half-Coders

Project 3

In [1]:
```
import pandas as pd

# Load the dataset
file_path = '500_Cities__Local_Data_for_Better_Health__2019_release.csv'
data = pd.read_csv(file_path)

data.head()
```

Out[1]:

	Year	StateAbbr	StateDesc	CityName	GeographicLevel	DataSource	Category	UniqueID	Measure	Data_Value_Unit	...	High_Confidence_Limit	Data_Value_Footn
0	2017	CA	California	Hawthorne	Census Tract	BRFSS	Health Outcomes	0632548-06037602504	Arthritis among adults aged >=18 Years	%	...	15.2	
1	2017	CA	California	Hawthorne	City	BRFSS	Unhealthy Behaviors	0632548	Current smoking among adults aged >=18 Years	%	...	15.9	
2	2017	CA	California	Hayward	City	BRFSS	Health Outcomes	0633000	Coronary heart disease among adults aged >=18 ...	%	...	4.8	
3	2017	CA	California	Hayward	City	BRFSS	Unhealthy Behaviors	0633000	Obesity among adults aged >=18 Years	%	...	24.4	
4	2017	CA	California	Hemet	City	BRFSS	Prevention	0633182	Cholesterol screening among adults aged >=18 Y...	%	...	78.3	

5 rows × 24 columns

Data Cleaning

In [2]:
```
# Check for missing values
missing_values = data.isnull().sum()

# Check for duplicates
duplicates = data.duplicated().sum()

# Inspect GeoLocation column format and missing values
geolocation_missing = data['GeoLocation'].isnull().sum()

print("missing_values",missing_values)
print("duplicates",duplicates)
print("geolocation_missing",geolocation_missing)
```

```
missing_values Year                              0
StateAbbr                        0
StateDesc                        0
CityName                        56
GeographicLevel                  0
DataSource                       0
Category                         0
UniqueID                         0
Measure                          0
Data_Value_Unit                  0
DataValueTypeID                  0
Data_Value_Type                  0
Data_Value                   22792
Low_Confidence_Limit         22792
High_Confidence_Limit        22792
Data_Value_Footnote_Symbol  787309
Data_Value_Footnote         787309
PopulationCount                  0
GeoLocation                     56
CategoryID                       0
MeasureId                        0
CityFIPS                        56
TractFIPS                    28056
Short_Question_Text              0
dtype: int64
duplicates 0
geolocation_missing 56
```

The initial data cleaning and preparation analysis reveals the following:

- Missing Values: There are no missing values for most of the columns, but there are significant gaps in Data_Value, Low_Confidence_Limit, High_Confidence_Limit, and extensive missing data in Data_Value_Footnote and Data_Value_Footnote_Symbol. Notably, GeoLocation, CityName, and CityFIPS each have 56 missing entries, and TractFIPS has 28,056 missing entries, indicating some records lack specific location details.
- Duplicates: There are no duplicate rows in the dataset, which is positive for the integrity of the data.
- GeoLocation Column: The GeoLocation column, crucial for mapping, has 56 missing entries. This column needs to be in a suitable format for geospatial analysis, typically as separate latitude and longitude columns or as a single column in a tuple format that can be directly used with mapping libraries.

Given these findings, the next steps will involve:

- We need to reduce dimensionality of the dataset since it is too large.
- Addressing missing GeoLocation entries. Given the context, it might be best to exclude these records from mapping analyses since their exact locations are unknown.
- Extracting latitude and longitude from the GeoLocation column into separate, usable columns for mapping.
- Proceeding with the cleaned dataset to create interactive maps and visualizations.

Data Preparation

To reduce the dataset to 20,000 rows without creating a highly skewed dataset with respect to geospatial columns and CityName, we can consider the following steps:

Columns to Drop: Identify and drop columns that may not be critical for interactive visualization and advanced analysis, like footnotes, confidence limits, and potentially redundant identifiers. Rows to Sample: Since we aim for geographic and city-based diversity, stratified sampling based on CityName could be a good approach, ensuring representation from all cities.

In [3]:
```
# Initial data summary
initial_summary = {
    'total_rows': data.shape[0],
    'total_columns': data.shape[1],
    'column_names': data.columns.tolist()
}

initial_summary
```

Out[3]:
```
{'total_rows': 810103,
 'total_columns': 24,
 'column_names': ['Year',
  'StateAbbr',
  'StateDesc',
  'CityName',
  'GeographicLevel',
  'DataSource',
  'Category',
  'UniqueID',
  'Measure',
  'Data_Value_Unit',
  'DataValueTypeID',
  'Data_Value_Type',
  'Data_Value',
  'Low_Confidence_Limit',
  'High_Confidence_Limit',
  'Data_Value_Footnote_Symbol',
  'Data_Value_Footnote',
  'PopulationCount',
  'GeoLocation',
  'CategoryID',
  'MeasureId',
  'CityFIPS',
  'TractFIPS',
  'Short_Question_Text']}
```

In [4]:
```
# Columns that might not be critical for analysis (based on description)
columns_to_drop = [
    'Data_Value_Unit', # Assuming all data values are in a consistent unit
    'DataValueTypeID', # Technical ID, not necessary for analysis
    'Data_Value_Type', # Assuming all values are of the same type or this isn't critical for visualizations
    'Low_Confidence_Limit', # Confidence limits might not be necessary for broad visualizations
    'High_Confidence_Limit', # Same as above
    'Data_Value_Footnote_Symbol', # Footnotes are not essential for initial analysis
    'Data_Value_Footnote', # Same as above
    'CategoryID', # Category should be sufficient without the need for an ID
    'MeasureId' # Measure should be sufficient without the need for an ID
]

# Drop identified columns
reduced_data = data.drop(columns=columns_to_drop)

# New data summary after dropping columns
new_summary = {
    'reduced_rows': reduced_data.shape[0],
    'reduced_columns': reduced_data.shape[1],
    'remaining_column_names': reduced_data.columns.tolist()
}

new_summary
```

Out[4]:
```
{'reduced_rows': 810103,
 'reduced_columns': 15,
 'remaining_column_names': ['Year',
  'StateAbbr',
  'StateDesc',
  'CityName',
  'GeographicLevel',
  'DataSource',
  'Category',
  'UniqueID',
  'Measure',
  'Data_Value',
  'PopulationCount',
  'GeoLocation',
  'CityFIPS',
  'TractFIPS',
  'Short_Question_Text']}
```

In [5]:
```
# Calculate the number of records per city to understand distribution
city_record_distribution = reduced_data['CityName'].value_counts().reset_index()
city_record_distribution.columns = ['CityName', 'RecordCount']

# Determine the number of cities
num_cities = city_record_distribution.shape[0]

# Summary of city distribution and the plan for manual stratification approach
city_distribution_summary = {
    'total_cities': num_cities,
    'city_records_distribution': city_record_distribution.head(), # Show a preview
}

city_distribution_summary
```

Out[5]:
```
{'total_cities': 474,
 'city_records_distribution':         CityName  RecordCount
 0      New York        59911
 1   Los Angeles        28119
 2       Chicago        22369
 3       Houston        16787
 4  Philadelphia        10707}
```

In [6]:
```
import pandas as pd

# Assuming 'reduced_data' is your DataFrame after removing null values
city_counts = reduced_data['CityName'].value_counts()

# Initialize an empty DataFrame to hold the sampled data
final_sampled_df = pd.DataFrame()

# Determine the number of records to sample per city to reach approximately 20,000 entries
total_target_samples = 20000
samples_per_city = max(1, total_target_samples // len(city_counts))

for city in city_counts.index:
    city_subset = reduced_data[reduced_data['CityName'] == city]
    # If the city subset is larger than 'samples_per_city', sample down to that number
    if len(city_subset) > samples_per_city:
        sampled_subset = city_subset.sample(n=samples_per_city, random_state=42)
    else:
        # If the city subset is smaller than the target sample size, use all records
        sampled_subset = city_subset
    # Append the sampled subset (or all records for that city) to the final DataFrame
    final_sampled_df = pd.concat([final_sampled_df, sampled_subset])

# If the final DataFrame is larger than the target, sample it down
if len(final_sampled_df) > total_target_samples:
    final_sampled_df = final_sampled_df.sample(n=total_target_samples, random_state=42)
```

In [7]:
```
final_sampled_df
```

Out[7]:

	Year	StateAbbr	StateDesc	CityName	GeographicLevel	DataSource	Category	UniqueID	Measure	Data_Value	PopulationCount	GeoLocation	City
516408	2017	NY	New York	New York	Census Tract	BRFSS	Health Outcomes	3651000-36047034200	Diagnosed diabetes among adults aged >=18 Years	19.4	6502	(40.5741076886, -73.9968626717)	3651(
521580	2017	NY	New York	New York	Census Tract	BRFSS	Unhealthy Behaviors	3651000-36047028502	Current smoking among adults aged >=18 Years	23.5	2802	(40.6974507413, -73.9408852337)	3651(
481677	2017	NY	New York	New York	Census Tract	BRFSS	Prevention	3651000-36047044200	Current lack of health insurance among adults ...	9.6	2658	(40.6123965321, -73.9655935668)	3651(

539731	2017	NY	New York	New York	Census Tract	BRFSS	Unhealthy Behaviors	3651000-36081049200	Obesity among adults aged >=18 Years	27.5	4750	(40.7229658389, -73.7584694313)	3651(
543089	2016	NY	New York	New York	Census Tract	BRFSS	Prevention	3651000-36081009400	Papanicolaou smear use among adult women aged ...	76.1	2834	(40.6814105586, -73.8365114137)	3651(
...	...	...	...	...	...	...	...	...	...	...	...	...	
282561	2016	ID	Idaho	Meridian	Census Tract	BRFSS	Prevention	1652120-16001010335	Mammography use among women aged 50–74 Years	69.6	18954	(43.6434512706, -116.437023046)	1652
289219	2016	ID	Idaho	Meridian	Census Tract	BRFSS	Prevention	1652120-16001010321	Papanicolaou smear use among adult women aged ...	80.4	5474	(43.6077830863, -116.36782846)	1652
292035	2017	ID	Idaho	Meridian	City	BRFSS	Unhealthy Behaviors	1652120	No leisure-time physical activity among adults...	20.1	75092	(43.6185195383, -116.39758487)	1652
285856	2017	ID	Idaho	Meridian	Census Tract	BRFSS	Health Outcomes	1652120-16001010321	Chronic kidney disease among adults aged >=18 ...	2.9	5474	(43.6077830863, -116.36782846)	1652
296946	2017	ID	Idaho	Meridian	Census Tract	BRFSS	Health Outcomes	1652120-16001010313	Mental health not good for >=14 days among adu...	11.3	12255	(43.5813084062, -116.37902549)	1652

19908 rows × 15 columns

In [8]:
```
# Saving the sampled dataset as a csv
final_sampled_df.to_csv('Project3_data.csv', index=False)
```

Reimporting dataset

In [9]:
```
# Load the dataset
file_path = 'Project3_data.csv'
data = pd.read_csv(file_path)

data.head()
```

Out[9]:

	Year	StateAbbr	StateDesc	CityName	GeographicLevel	DataSource	Category	UniqueID	Measure	Data_Value	PopulationCount	GeoLocation	CityFIPS	
0	2017	NY	New York	New York	Census Tract	BRFSS	Health Outcomes	3651000-36047034200	Diagnosed diabetes among adults aged >=18 Years	19.4	6502	(40.5741076886, -73.9968626717)	3651000.0	3
1	2017	NY	New York	New York	Census Tract	BRFSS	Unhealthy Behaviors	3651000-36047028502	Current smoking among adults aged >=18 Years	23.5	2802	(40.6974507413, -73.9408852337)	3651000.0	3
2	2017	NY	New York	New York	Census Tract	BRFSS	Prevention	3651000-36047044200	Current lack of health insurance among adults ...	9.6	2658	(40.6123965321, -73.9655935668)	3651000.0	3
3	2017	NY	New York	New York	Census Tract	BRFSS	Unhealthy Behaviors	3651000-36081049200	Obesity among adults aged >=18 Years	27.5	4750	(40.7229658389, -73.7584694313)	3651000.0	3
4	2016	NY	New York	New York	Census Tract	BRFSS	Prevention	3651000-36081009400	Papanicolaou smear use among adult women aged ...	76.1	2834	(40.6814105586, -73.8365114137)	3651000.0	3

In [10]:
```python
# Check for missing values
missing_values = data.isnull().sum()

# Check for duplicates
duplicates = data.duplicated().sum()

# Inspect GeoLocation column format and missing values
geolocation_missing = data['GeoLocation'].isnull().sum()

print("missing_values",missing_values)
print("duplicates",duplicates)
print("geolocation_missing",geolocation_missing)
```

```
missing_values Year                      0
StateAbbr              0
StateDesc              0
CityName               0
GeographicLevel        0
DataSource             0
Category               0
UniqueID               0
Measure                0
Data_Value           645
PopulationCount        0
GeoLocation            0
CityFIPS               0
TractFIPS           1301
Short_Question_Text    0
dtype: int64
duplicates 3
geolocation_missing 0
```

Addressing missing and duplicate entries and extracting latitude and longitude

In [11]:
```
# Removing duplicates
data_cleaned = data.drop_duplicates()

# Handling missing values in 'Data_Value' by removing rows with missing 'Data_Value'
data_cleaned = data_cleaned.dropna(subset=['Data_Value'])

# Splitting the 'GeoLocation' into two separate columns 'Latitude' and 'Longitude'
data_cleaned[['Latitude', 'Longitude']] = data_cleaned['GeoLocation'].str.extract(r'\((.*), (.*)\)').astype(float)

# Dropping the original 'GeoLocation' column
data_cleaned = data_cleaned.drop(columns=['GeoLocation'])


# Check for missing values
missing_values = data_cleaned.isnull().sum()

# Check for duplicates
duplicates = data_cleaned.duplicated().sum()

print("missing_values",missing_values)
print("duplicates",duplicates)
print("geolocation_missing",geolocation_missing)
```

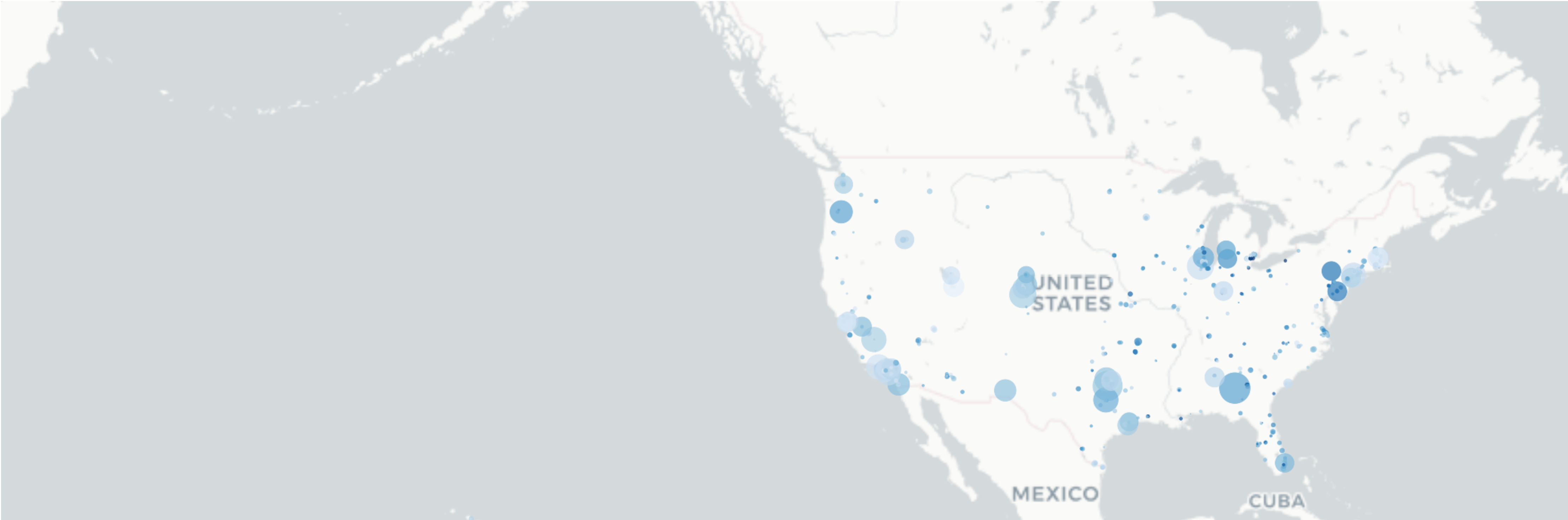
```
missing_values Year                      0
StateAbbr              0
StateDesc              0
CityName               0
GeographicLevel        0
DataSource             0
Category               0
UniqueID               0
Measure                0
Data_Value             0
PopulationCount        0
CityFIPS               0
TractFIPS           1292
Short_Question_Text    0
Latitude               0
Longitude              0
dtype: int64
duplicates 0
geolocation_missing 0
```

Geospatial Data Analysis and Interactive Visualization

Interactive map visualization showing the locations of the data points, colored by a selected measure, for diabetes and obesity

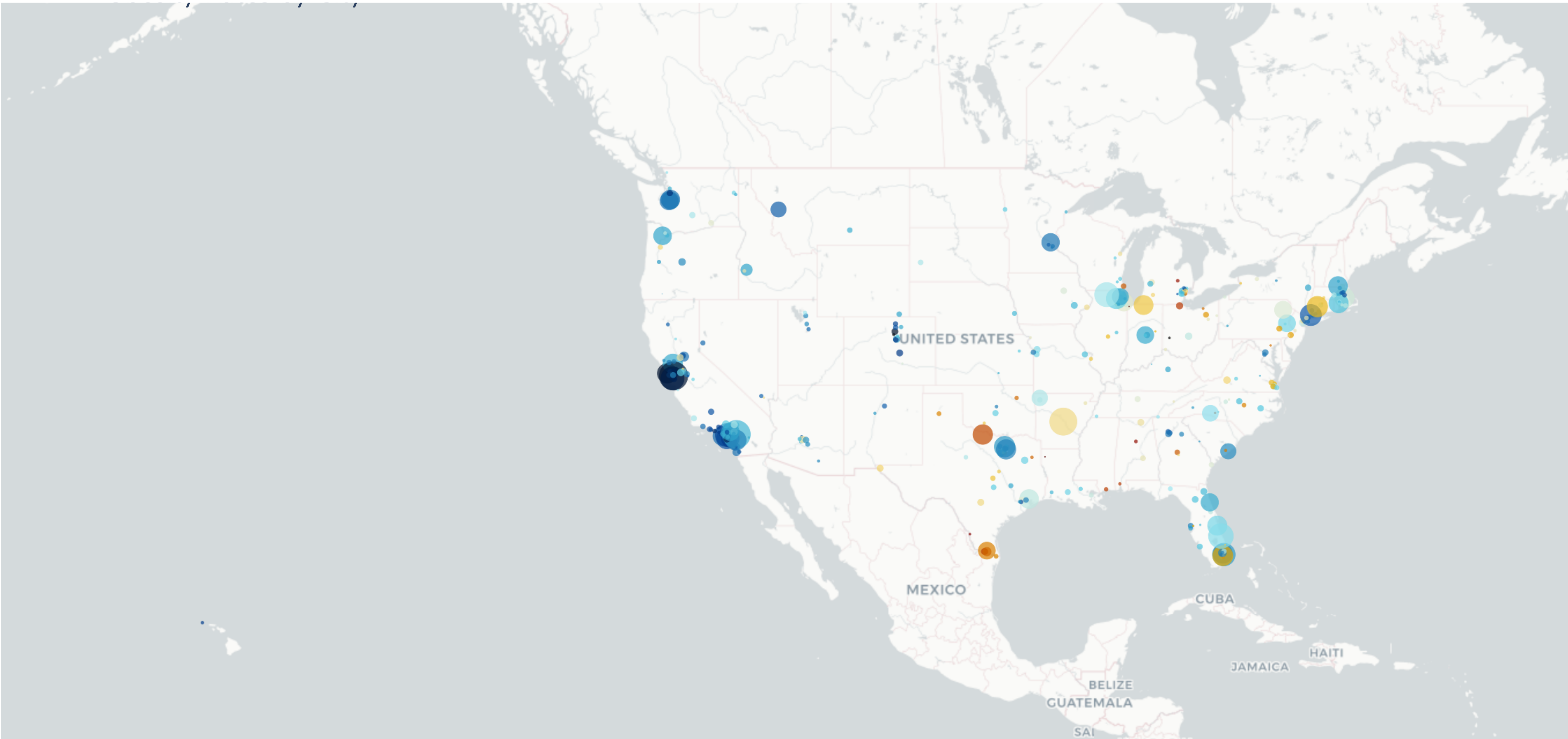
In [12]:
```python
#Trends in Unhealthy Behaviors: Smoking Rates

import plotly.express as px
# Filter the dataset for smoking-related measures
smoking_data = data_cleaned[data_cleaned['Measure'].str.contains('Current smoking')]

# Create an interactive map using Plotly Express for smoking data
fig_smoking = px.scatter_mapbox(smoking_data,
                                lat="Latitude",
                                lon="Longitude",
                                color="Data_Value",
                                size="PopulationCount",
                                color_continuous_scale=px.colors.sequential.Blues,
                                size_max=15,
                                zoom=10,
                                mapbox_style="carto-positron",
                                title="Smoking Rates")

# To save the visualization as an HTML file
fig_smoking.show()
fig_smoking.write_html("smoking_rates_nyc.html")
```

Smoking Rates



Map of Health Outcomes by City: Plot latitude and longitude on a map to show health outcomes (like Diabetes or Obesity rates) across different cities. Users can hover over locations to see detailed data.

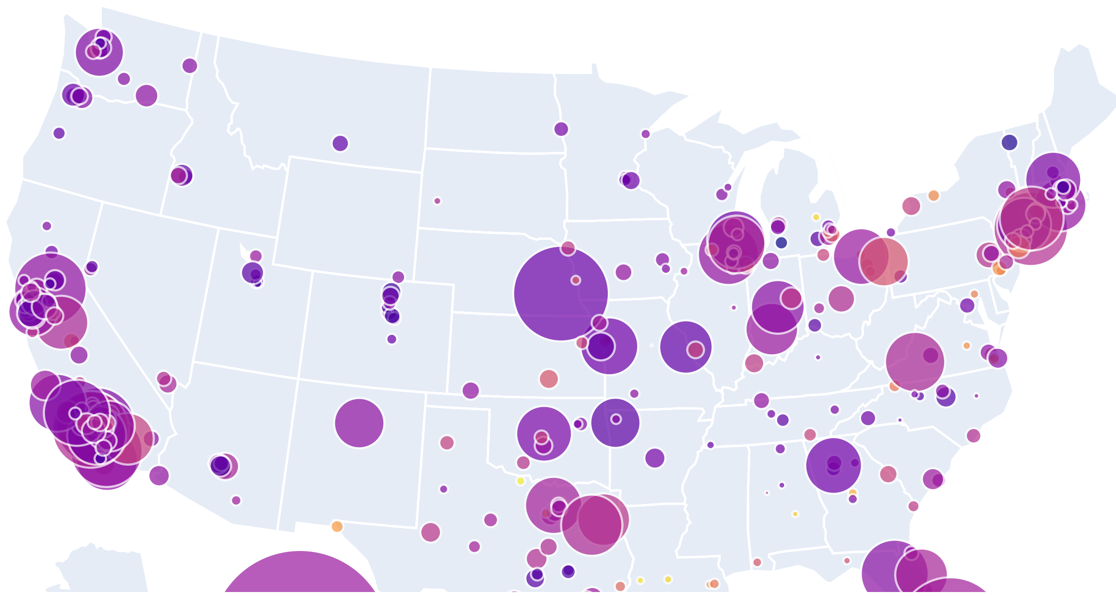
In [13]:
```
# We'll need to extract latitude and longitude from the GeoLocation column
data[['Latitude', 'Longitude']] = data['GeoLocation'].str.strip('()').str.split(', ', expand=True).astype(float)

# For demonstration, let's focus on Obesity rates as the health outcome
obesity_data = data[data['Short_Question_Text'] == 'Obesity']

# Assuming there might be multiple entries for a city, we calculate the average Obesity rate
obesity_avg = obesity_data.groupby('CityName').agg({'Latitude': 'first', 'Longitude': 'first', 'Data_Value': 'mean', 'PopulationCount': 'sum'})

# Creating the map
fig = px.scatter_mapbox(obesity_avg,
                        lat="Latitude",
                        lon="Longitude",
                        size="PopulationCount",
                        color="Data_Value",
                        hover_name="CityName",
                        hover_data=["Data_Value", "PopulationCount"],
                        color_continuous_scale=px.colors.cyclical.IceFire,
                        size_max=15,
                        zoom=3,
                        mapbox_style="carto-positron")

fig.update_layout(title='Obesity Rates by City',
                  geo=dict(scope='usa'),
                  margin={"r":0,"t":0,"l":0,"b":0})

fig.show()
fig.write_html("Obesity Rates by City.html")
```



Bubble Map for Population and Health Metrics: Utilize a bubble map where the size of each bubble represents the population of the city, and the color represents a health metric (e.g., diabetes rate). This can help visualize how health outcomes correlate with population size.

In [14]:
```
# Extract latitude and longitude from the 'GeoLocation' column
data[['Latitude', 'Longitude']] = data['GeoLocation'].str.strip('()').str.split(', ', expand=True).astype(float)

# Choose a health metric for the visualization, e.g., 'Diabetes'
health_metric = 'Diabetes'
diabetes_data = data[data['Short_Question_Text'] == health_metric]

# Calculate the average rate of diabetes and the total population for each city
city_diabetes_data = diabetes_data.groupby('CityName').agg({
    'Latitude': 'mean',
    'Longitude': 'mean',
    'Data_Value': 'mean',
    'PopulationCount': 'sum'
}).reset_index()

# Create a bubble map
fig = px.scatter_geo(city_diabetes_data,
                     lat='Latitude',
                     lon='Longitude',
                     size='PopulationCount',
                     color='Data_Value',
                     hover_name='CityName',
                     hover_data=['Data_Value', 'PopulationCount'],
                     size_max=60,
                     title='Bubble Map for Population and Diabetes Rates in USA')

# Update the layout
fig.update_layout(geo=dict(scope='usa'))

# Show the plot
fig.show()
fig.write_html("Bubble Map for Population and Diabetes Rates in USA.html")
```

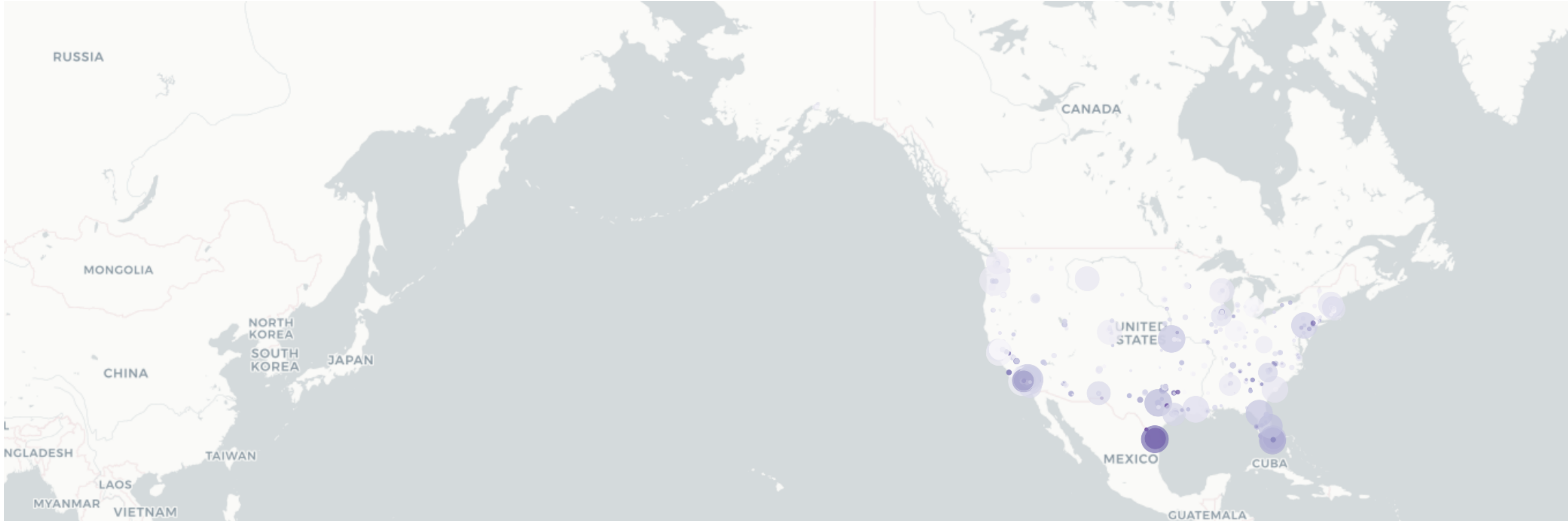
Bubble Map for Population and Diabetes Rates in USA



In [15]:

```
#Prevention Measures Analysis: Health Insurance Coverage
# Filter the dataset for health insurance coverage
insurance_data = data_cleaned[data_cleaned['Measure'].str.contains('health insurance')]

# Create an interactive map using Plotly Express for health insurance coverage data
fig_insurance = px.scatter_mapbox(insurance_data,
                                  lat="Latitude",
                                  lon="Longitude",
                                  color="Data_Value",
                                  size="PopulationCount",
                                  color_continuous_scale=px.colors.sequential.Purples,
                                  size_max=15,
                                  zoom=10,
                                  mapbox_style="carto-positron",
                                  title="Health Insurance Coverage")
fig_insurance.show()
# To save the visualization as an HTML file
fig_insurance.write_html("health_insurance_coverage_nyc.html")
```

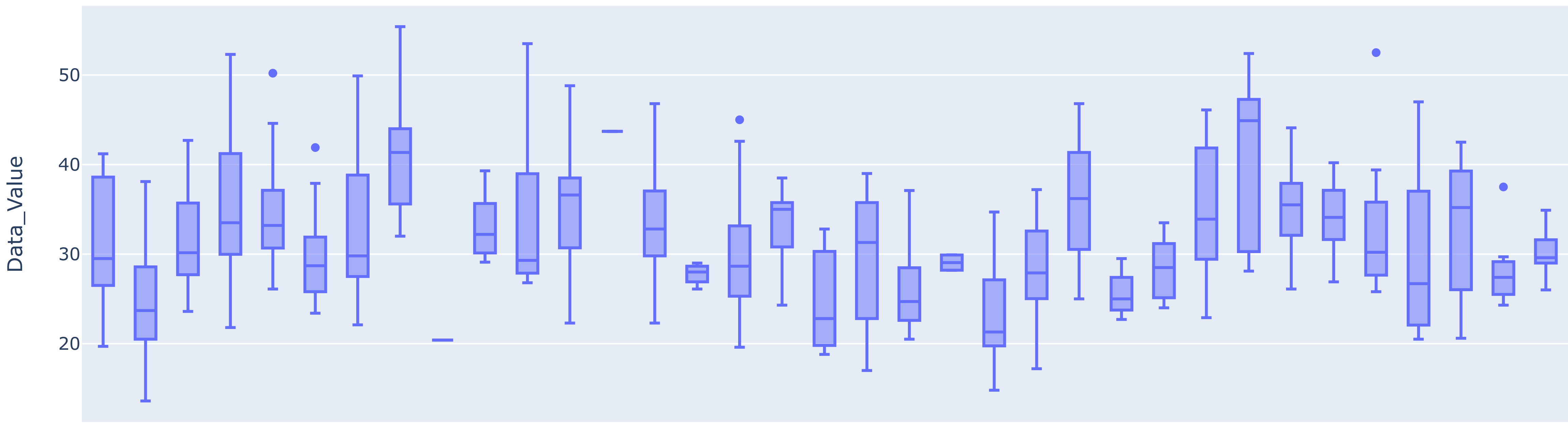
Health Insurance Coverage



Box Plot for State-wise Health Data Distribution: Use box plots to show the distribution of a specific health metric (like obesity rates) across different states. This allows for comparisons and identification of outliers.

In [16]:
```
# Filter for a specific health metric, e.g., 'Obesity'
# Replace 'Obesity' with your specific health metric of interest
health_metric = 'Obesity'
obesity_data = data[data['Short_Question_Text'] == health_metric]

# Create a box plot to show the distribution of the health metric across different states
fig = px.box(obesity_data, x='StateDesc', y='Data_Value',
             title=f'State-wise Health Data Distribution of {health_metric}')

fig.show()  # To show the plot
fig.write_html("State-wise Health Data Distribution.html")
# Note: You may want to ensure that 'StateDesc' is the correct column name for state descriptions in your dataset.
# Also, ensure 'Data_Value' is the column that contains the health metric values you want to plot.
```

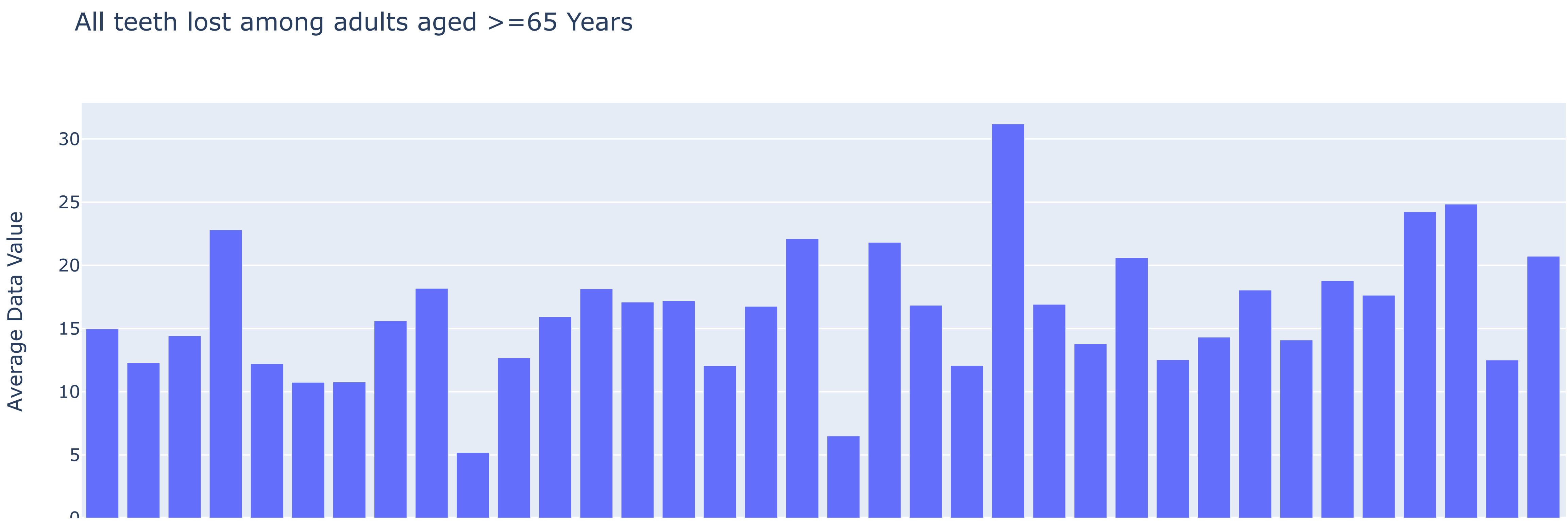
State-wise Health Data Distribution of Obesity



Interactive Bar Charts for Category Comparison: Create bar charts comparing health outcomes or behaviors across different categories for a selected city or state. Users can select different measures from a dropdown menu.

In [17]:
```
import pandas as pd
import plotly.graph_objects as go

# Aggregate data for each measure across all states
aggregated_data = data.groupby(['Measure', 'StateDesc'])['Data_Value'].mean().reset_index()

# Create abbreviations for each measure
measures = aggregated_data['Measure'].unique()
abbreviations = {measure: f"M{index+1}" for index, measure in enumerate(measures)}

# Reverse mapping for using in callbacks
full_names = {abbr: measure for measure, abbr in abbreviations.items()}

# Define the figure
fig = go.Figure()

# Initial measure to display
initial_measure = measures[0]
initial_abbr = abbreviations[initial_measure]

# Filter data for the initial measure
measure_data = aggregated_data[aggregated_data['Measure'] == initial_measure]

# Add the bar chart for the initial measure
fig.add_trace(go.Bar(x=measure_data['StateDesc'], y=measure_data['Data_Value'], name=initial_abbr))

# Update layout
fig.update_layout(title_text=f'{initial_measure}',
                  xaxis_title="State",
                  yaxis_title="Average Data Value")

# Dropdown menu for selecting measures
buttons = [
    dict(label=abbreviations[measure],
         method="update",
         args=[{"y": [aggregated_data[aggregated_data['Measure'] == measure]['Data_Value']],
                "x": [aggregated_data[aggregated_data['Measure'] == measure]['StateDesc']],
                "name": abbreviations[measure]},
               {"title": f"{measure}"}])
    for measure in measures]

# Add dropdown to the figure
fig.update_layout(
    updatemenus=[dict(buttons=buttons,
```

```
                      direction="down",
                      pad={"r": 10, "t": 10},
                      showactive=True,
                      x=1,
                      xanchor="left",
                      y=1.1,
                      yanchor="top")])

fig.show()
fig.write_html("abc.html")
```



In [ ]: