EDA of GIS Final Project

US Map at State Level

Data Setup

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

In [ ]:
```
# County Level Data
# Link: https://data.cdc.gov/500-Cities-Places/PLACES-County-Data-GIS-Friendly-Format-2023-releas/i46a-9kgh/about_data
df = pd.read_csv("C:/Users/user/Desktop/GIS/Final Project/PLACES__County_Data__GIS_Friendly_Format___2023_release_2024050
```

In [ ]:
```
df.head(5)
```

Out[ ]:

	StateAbbr	StateDesc	CountyName	CountyFIPS	TotalPopulation	ACCESS2_CrudePrev	ACCESS2_Crude95CI	ACCESS2_AdjPrev	ACCESS2_Adj95
0	AL	Alabama	Autauga	1001	59095	10.0	(7.7, 12.6)	10.4	(8.0, 13
1	AL	Alabama	Bullock	1011	10320	18.7	(15.3, 23.0)	19.2	(15.7, 23
2	AL	Alabama	Chilton	1021	45274	13.5	(10.8, 16.9)	14.1	(11.2, 17
3	AL	Alabama	Cleburne	1029	15103	11.8	(8.9, 15.1)	12.5	(9.4, 16
4	AL	Alabama	DeKalb	1049	71813	15.9	(12.5, 20.1)	16.7	(13.2, 21

5 rows × 154 columns

Import USA Shapefile

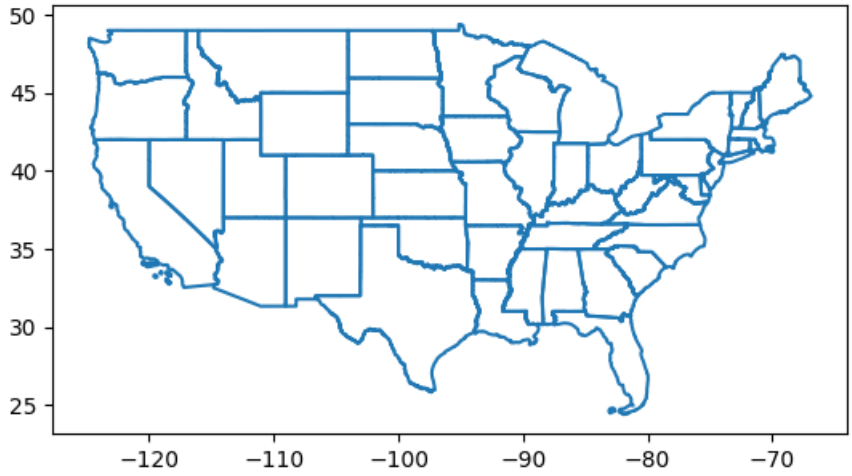
In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import contextily as ctx
import geopandas as gpd
import os
from mpl_toolkits.axes_grid1 import make_axes_locatable

path = "C:/Users/user/Desktop/GIS/Final Project/tl_2023_us_state/tl_2023_us_state.shp"
usa = gpd.read_file(path)
usa = usa.to_crs("EPSG:4326")
```

In [ ]:
```
usa.iloc[0,:]
```

Out[ ]:
```
REGION                                                     3
DIVISION                                                   5
STATEFP                                                   54
STATENS                                             01779805
GEOID                                                     54
GEOIDFQ                                          0400000US54
STUSPS                                                   WV
NAME                                           West Virginia
LSAD                                                      00
MTFCC                                                 G4000
FUNCSTAT                                                  A
ALAND                                           62266499712
AWATER                                            489003081
INTPTLAT                                        +38.6472854
INTPTLON                                       -080.6183274
geometry    POLYGON ((-77.754376 39.333461, -77.754219 39....
Name: 0, dtype: object
```

In [ ]:
```
# Exclude non-continental states
non_continental = ['HI','VI','MP','GU','AK','AS','PR']
for n in non_continental:
    usa = usa[usa.STUSPS != n]
```

In [ ]:
```
usa.boundary.plot()
plt.show()
```



Define Plotting Function

In [ ]:
```
# Define function to plot states
def StatesPlot(df, data, cmap):
    f, ax = plt.subplots(1, 1, figsize=(15, 10), sharex=True, sharey=True, dpi=300)
    f.tight_layout()
    plt.title('United States Map - Variable = ' + data)
    ax.set_axis_off()
    divider = make_axes_locatable(ax)
    cax = divider.append_axes("right", size="3%", pad=0.5, alpha=0.5)
    df.plot(data, ax=ax, alpha=0.5, cmap=cmap, edgecolor='k', legend=True, cax=cax, linewidth=0.5)
    plt.ylabel('Number', fontsize=12)
    plt.show()
```

Aggregate to State-Level

In [ ]:
```
# Filter columns that end with '_CrudePrev'
columns_to_group = [col for col in df.columns if col.endswith('_CrudePrev')] + ['TotalPopulation']

# Group by StateAbbr and aggregate the columns
state_df = df.groupby('StateAbbr')[columns_to_group].sum().reset_index()
```

In [ ]:
```
print("Columns in usa GeoDataFrame:", usa.columns)
print("Columns in state_df DataFrame:", state_df.columns)
```

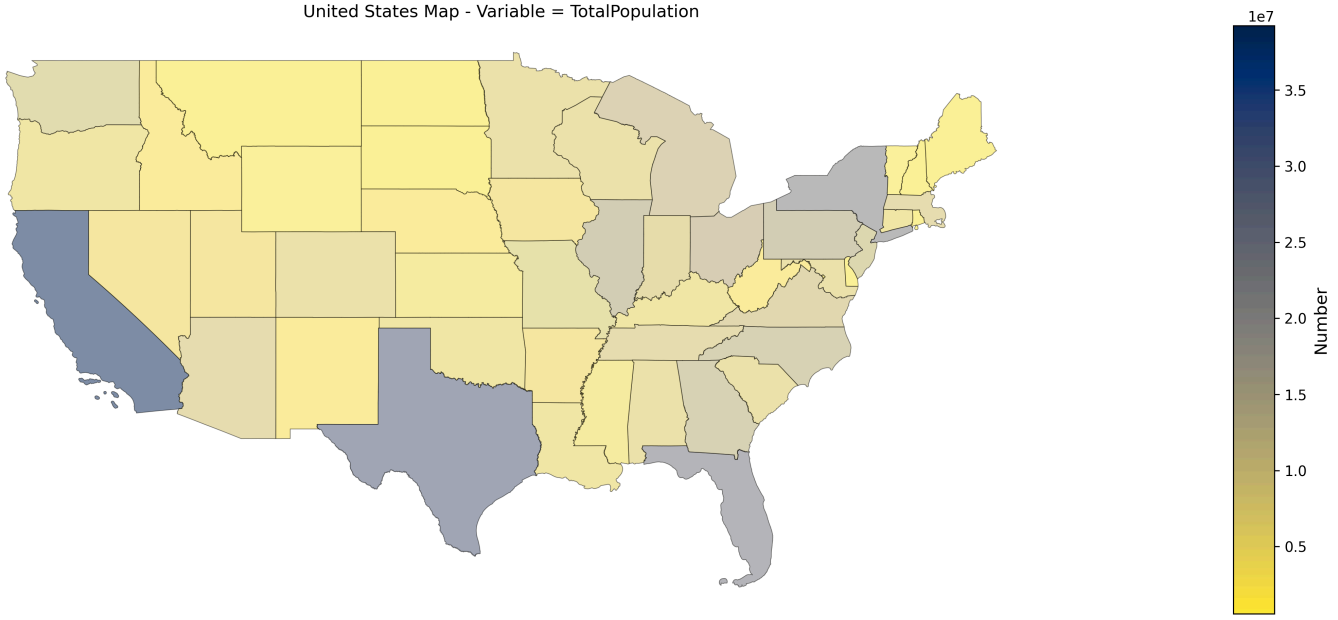
```
Columns in usa GeoDataFrame: Index(['REGION', 'DIVISION', 'STATEFP', 'STATENS', 'GEOID', 'GEOIDFQ',
       'STUSPS', 'NAME', 'LSAD', 'MTFCC', 'FUNCSTAT', 'ALAND', 'AWATER',
       'INTPTLAT', 'INTPTLON', 'geometry'],
      dtype='object')
Columns in state_df DataFrame: Index(['StateAbbr', 'ACCESS2_CrudePrev', 'ARTHRITIS_CrudePrev',
       'BINGE_CrudePrev', 'BPHIGH_CrudePrev', 'BPMED_CrudePrev',
       'CANCER_CrudePrev', 'CASTHMA_CrudePrev', 'CERVICAL_CrudePrev',
       'CHD_CrudePrev', 'CHECKUP_CrudePrev', 'CHOLSCREEN_CrudePrev',
       'COLON_SCREEN_CrudePrev', 'COPD_CrudePrev', 'COREM_CrudePrev',
       'COREW_CrudePrev', 'CSMOKING_CrudePrev', 'DENTAL_CrudePrev',
       'DEPRESSION_CrudePrev', 'DIABETES_CrudePrev', 'GHLTH_CrudePrev',
       'HIGHCHOL_CrudePrev', 'KIDNEY_CrudePrev', 'LPA_CrudePrev',
       'MAMMOUSE_CrudePrev', 'MHLTH_CrudePrev', 'OBESITY_CrudePrev',
       'PHLTH_CrudePrev', 'SLEEP_CrudePrev', 'STROKE_CrudePrev',
       'TEETHLOST_CrudePrev', 'HEARING_CrudePrev', 'VISION_CrudePrev',
       'COGNITION_CrudePrev', 'MOBILITY_CrudePrev', 'SELFCARE_CrudePrev',
       'INDEPLIVE_CrudePrev', 'DISABILITY_CrudePrev', 'TotalPopulation'],
      dtype='object')
```

Merge Shapefile with Selected Dataframe

In [ ]:
```
# Merge aggregated DataFrame with the GeoDataFrame
merged_gdf = usa.merge(state_df, left_on='STUSPS', right_on='StateAbbr', how='left')
```

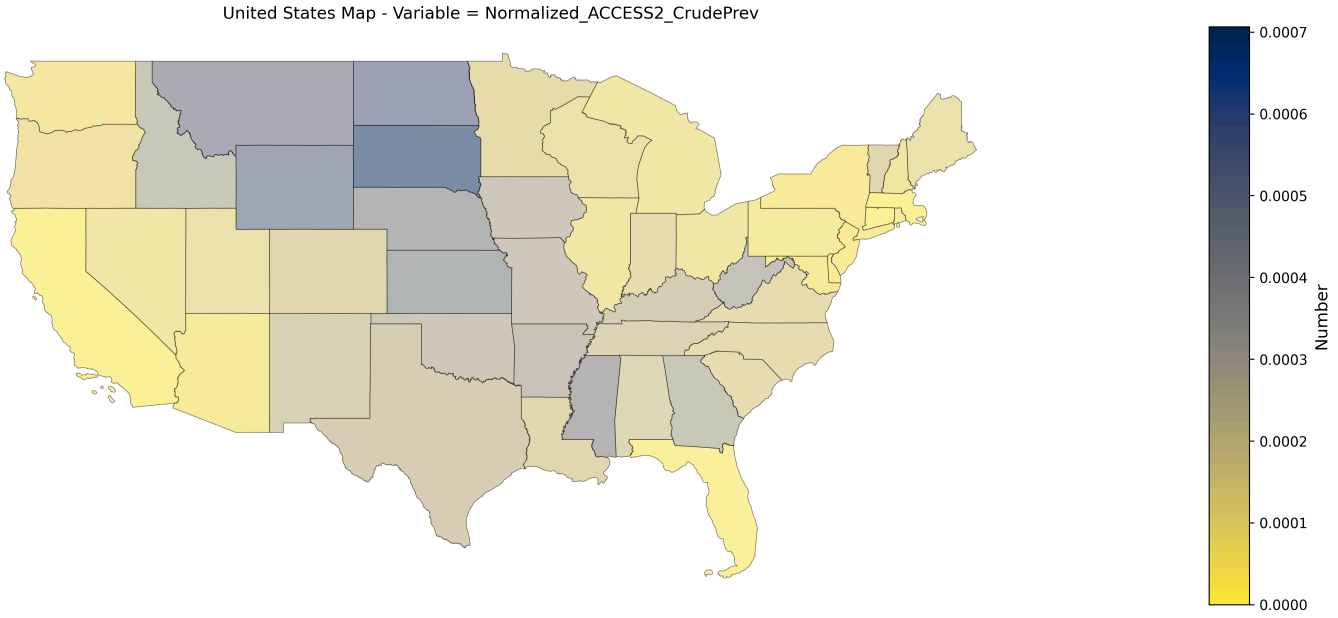
Population by State (Non-Normalized)

In [ ]:
```
StatesPlot(merged_gdf,'TotalPopulation', 'cividis_r')
```



Normalized Heatmap

In [ ]:
```
# Calculate normalized insurance data by dividing by population
merged_gdf['Normalized_ACCESS2_CrudePrev'] = merged_gdf['ACCESS2_CrudePrev'] / merged_gdf['TotalPopulation']

StatesPlot(merged_gdf,'Normalized_ACCESS2_CrudePrev', 'cividis_r')
```



Normalized Every Columns that End with "_CrudePrev" and Save for Further Use

In [ ]:
```
# Get columns ending with '_CrudePrev'
columns_to_plot = [col for col in df.columns if col.endswith('_CrudePrev')]

def calculate_normalized_data(geo_df, data_columns, population_column):
    for column in data_columns:
        # Calculate normalized data by dividing by population
        normalized_column = f'Normalized_{column}'
        geo_df[normalized_column] = geo_df[column] / geo_df[population_column]

    # Drop the original data columns
    geo_df.drop(columns=data_columns, inplace=True)

    return geo_df
```

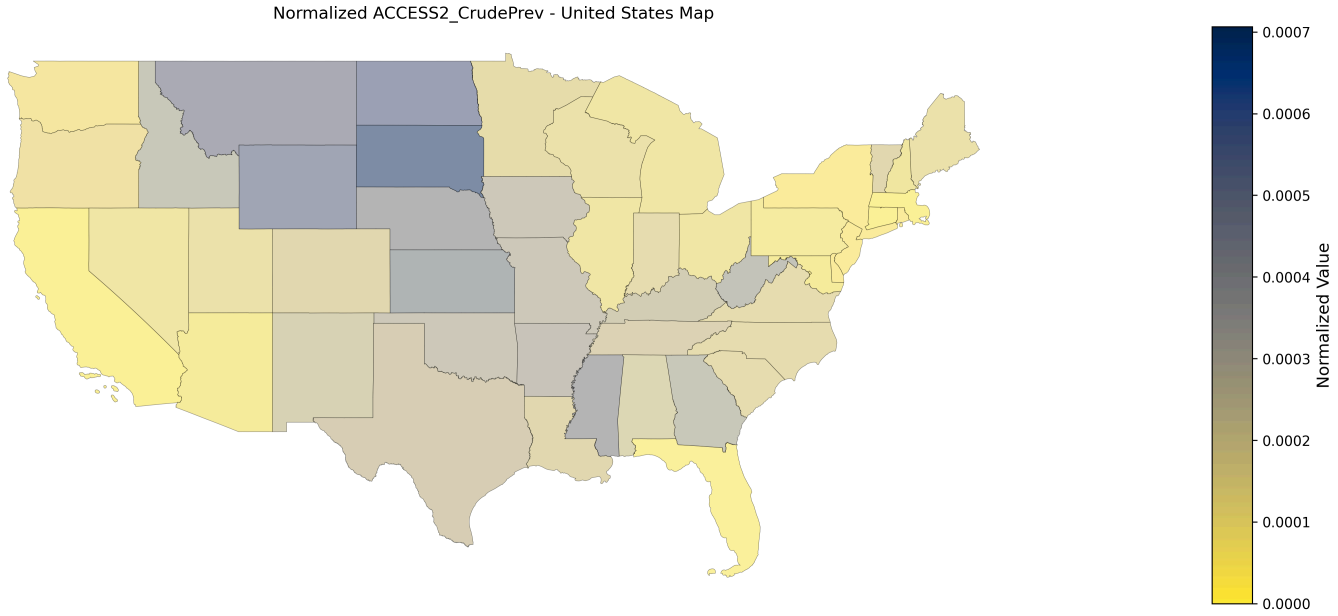
```
# Assuming merged_gdf is your DataFrame
df2 = calculate_normalized_data(merged_gdf, columns_to_plot, 'TotalPopulation')
```

In [ ]:
```
# Specify the output file path
output_file = "normalized_state.csv"

# Save the DataFrame to a CSV file for further use
df2.to_csv(output_file, index=False)
```

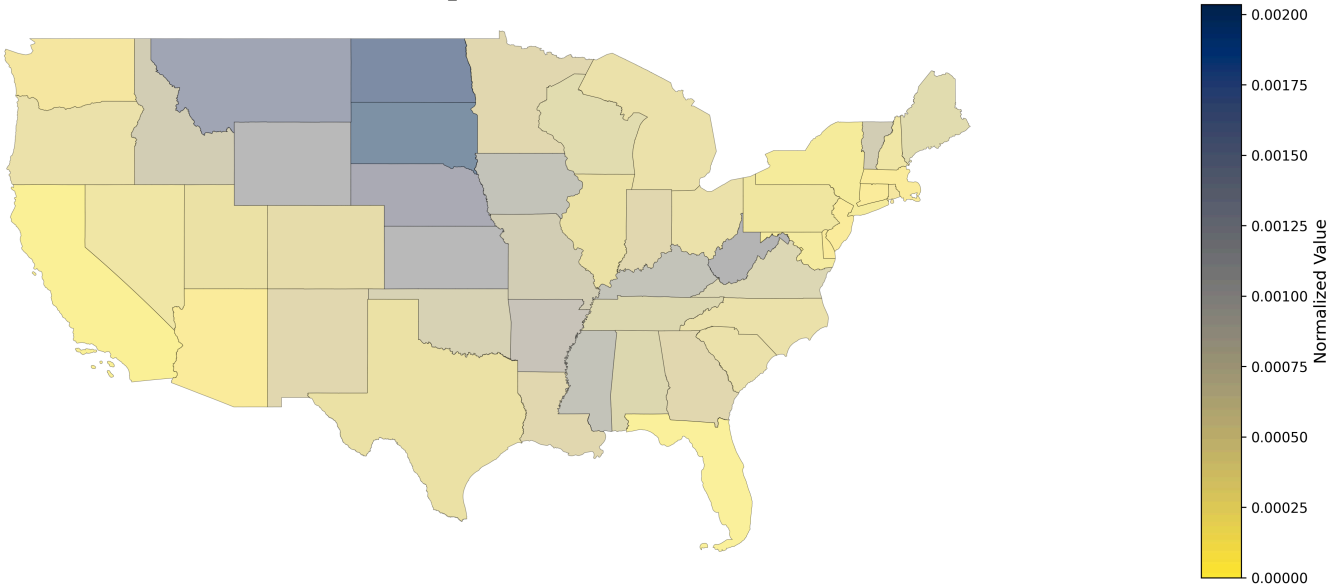
Function to Plot Multiple Columns at Once

In [ ]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable

def plot_normalized_data_multiple_columns(geo_df, data_columns, population_column, colormap):
    for column in data_columns:
        # Calculate normalized data by dividing by population
        normalized_column = f'Normalized_{column}'
        geo_df[normalized_column] = geo_df[column] / geo_df[population_column]

        # Plotting
        f, ax = plt.subplots(1, 1, figsize=(15, 10), sharex=True, sharey=True, dpi=300)
        f.tight_layout()
        plt.title(f'Normalized {column} - United States Map')
        ax.set_axis_off()
        divider = make_axes_locatable(ax)
        cax = divider.append_axes("right", size="3%", pad=0.5, alpha=0.5)
        geo_df.plot(normalized_column, ax=ax, alpha=0.5, cmap=colormap, edgecolor='k', legend=True, cax=cax, linewidth=0.
        plt.ylabel('Normalized Value', fontsize=12)
        plt.show()
```

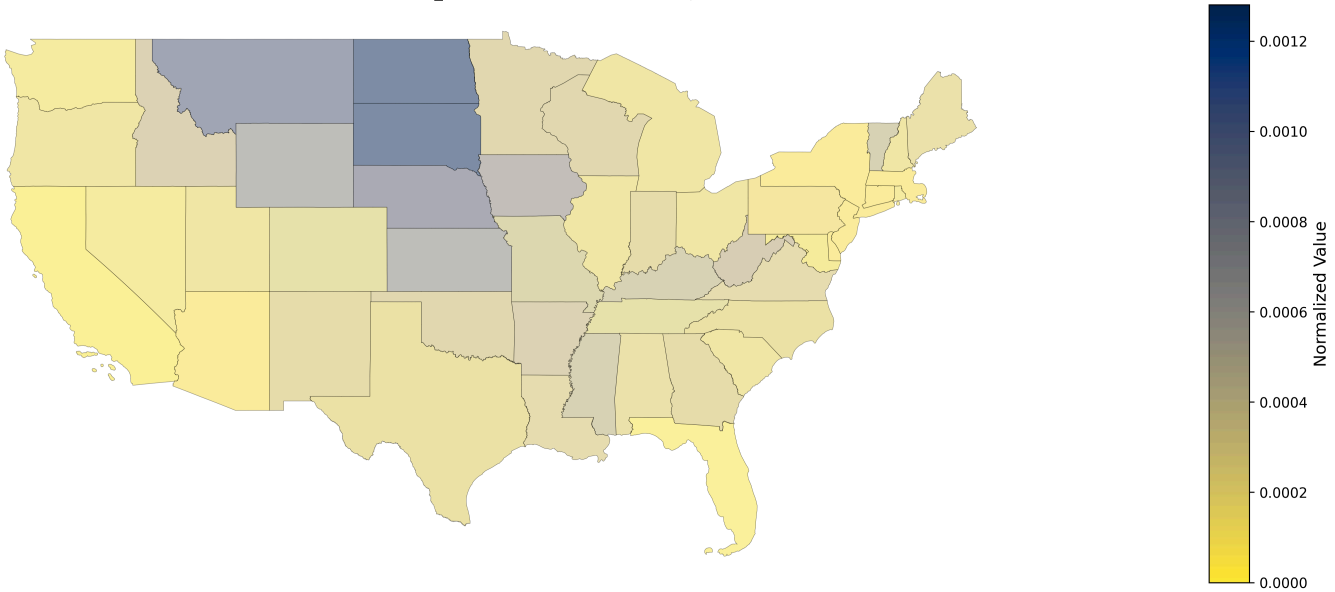
In [ ]:
```
# Get columns ending with '_CrudePrev'
columns_to_plot = [col for col in df.columns if col.endswith('_CrudePrev')]

plot_normalized_data_multiple_columns(merged_gdf, columns_to_plot, 'TotalPopulation', 'cividis_r')
```



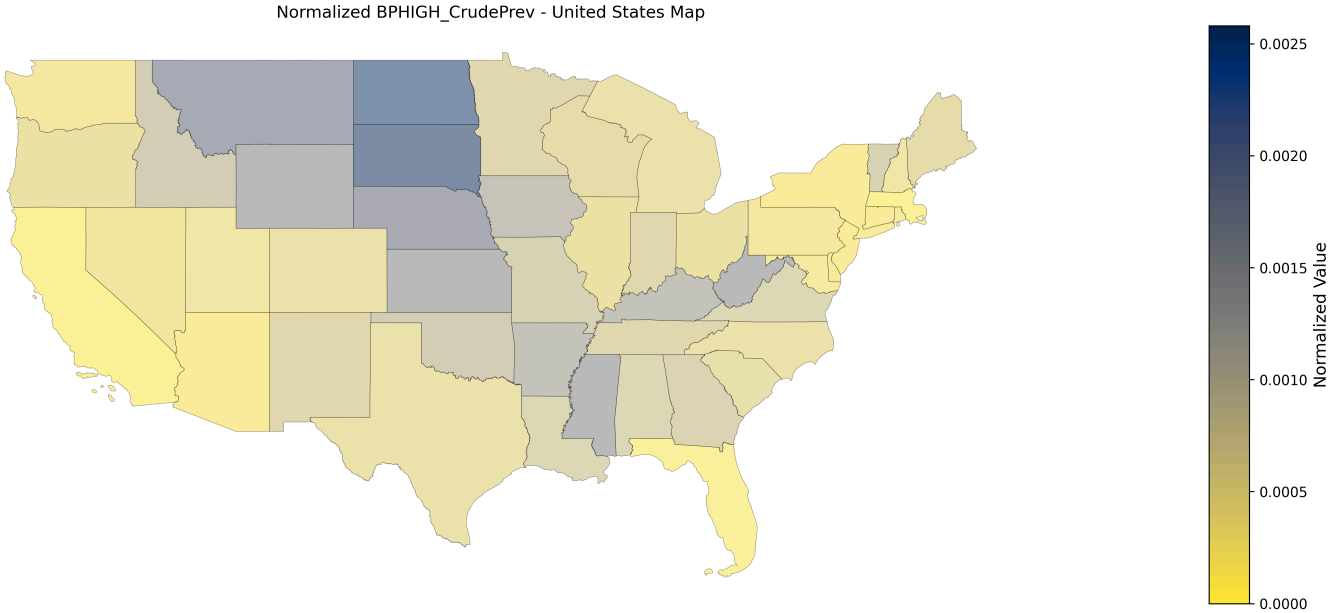
Normalized ARTHRITIS_CrudePrev - United States Map



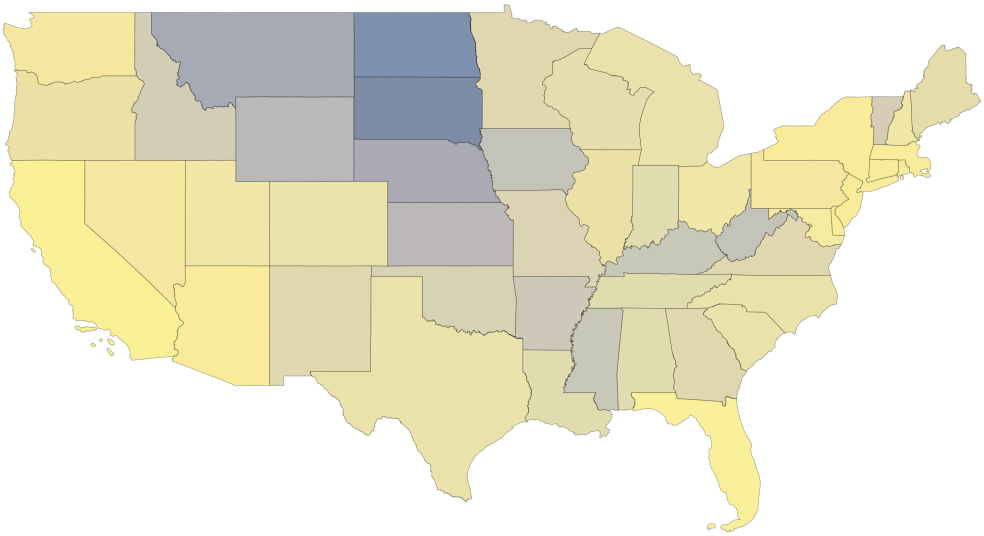
Normalized BINGE_CrudePrev - United States Map

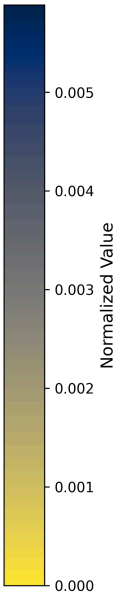


Normalized BPHIGH_CrudePrev - United States Map

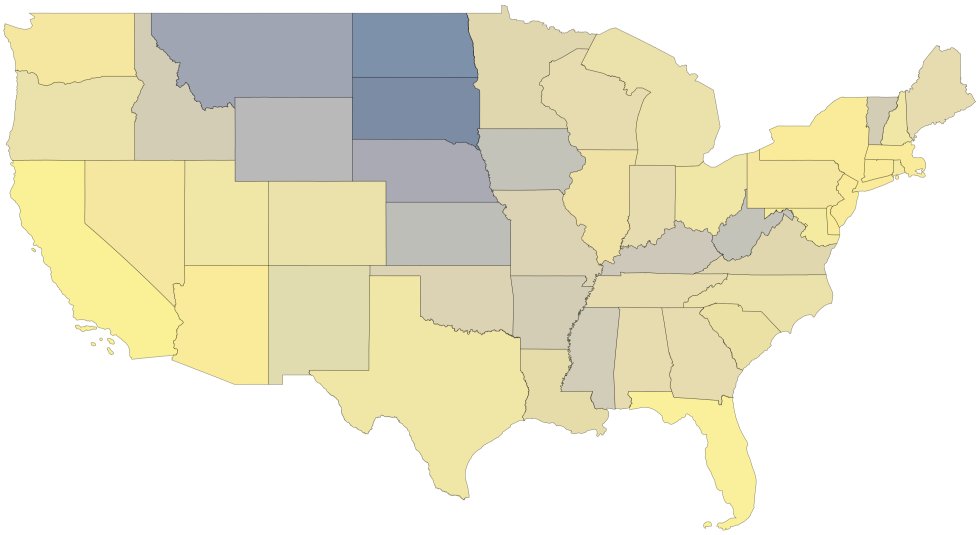


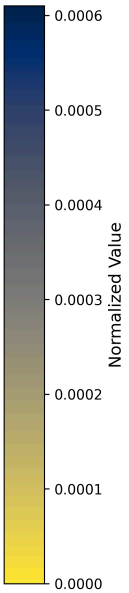
Normalized BPMED_CrudePrev - United States Map



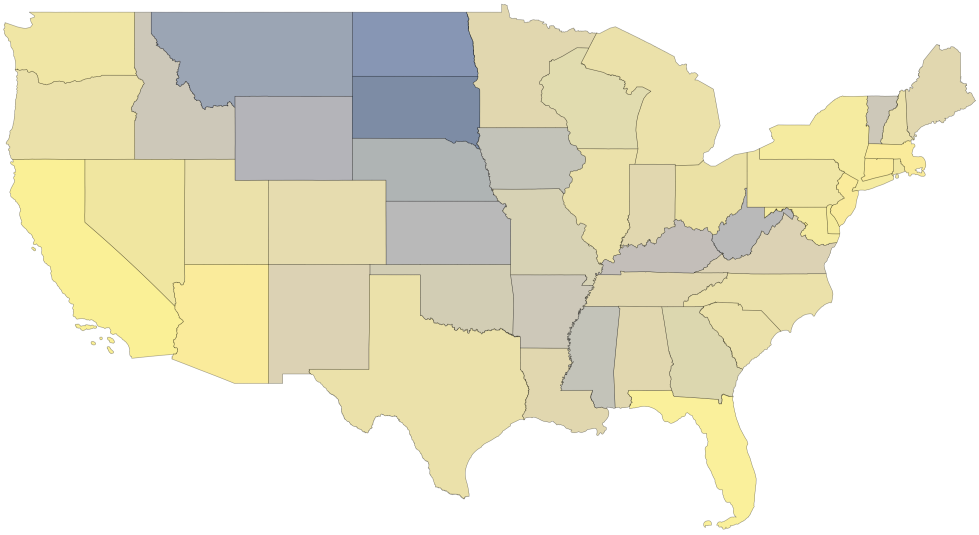


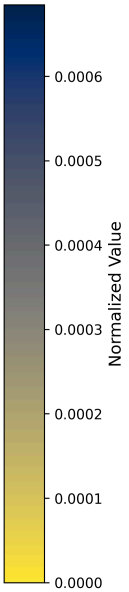
Normalized CANCER_CrudePrev - United States Map



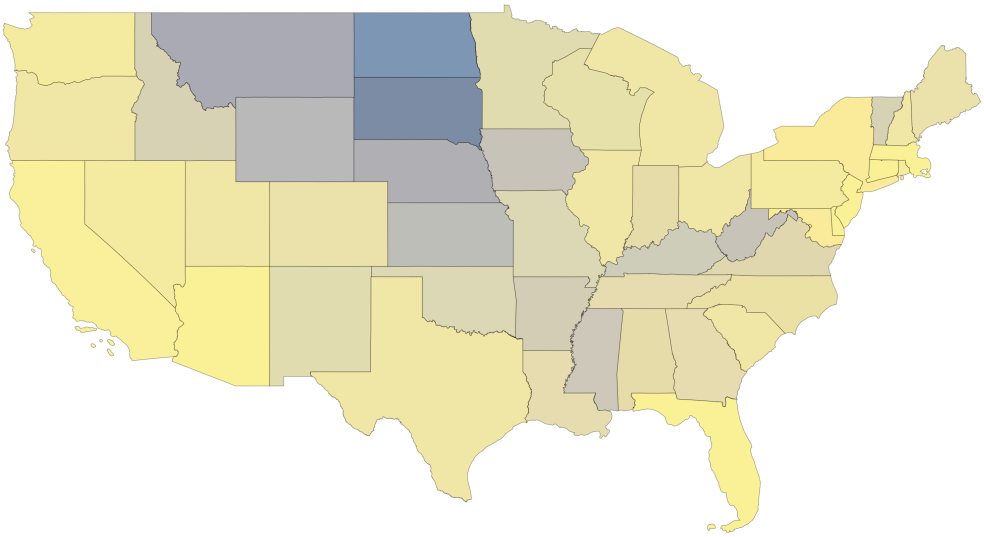


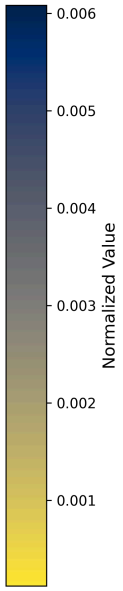
Normalized CASTHMA_CrudePrev - United States Map



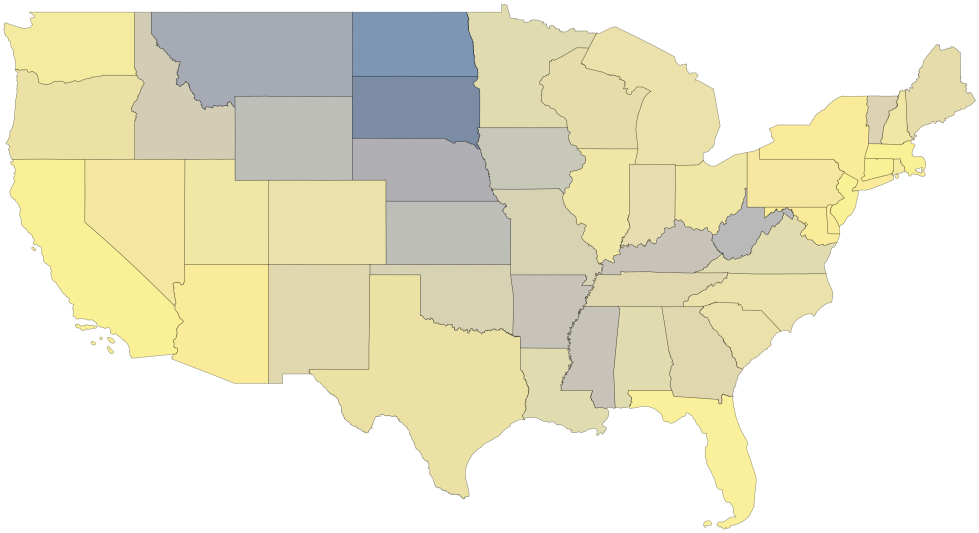


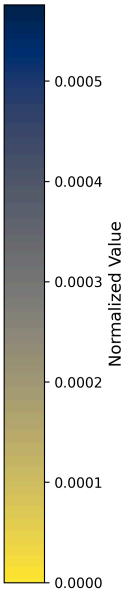
Normalized CERVICAL_CrudePrev - United States Map



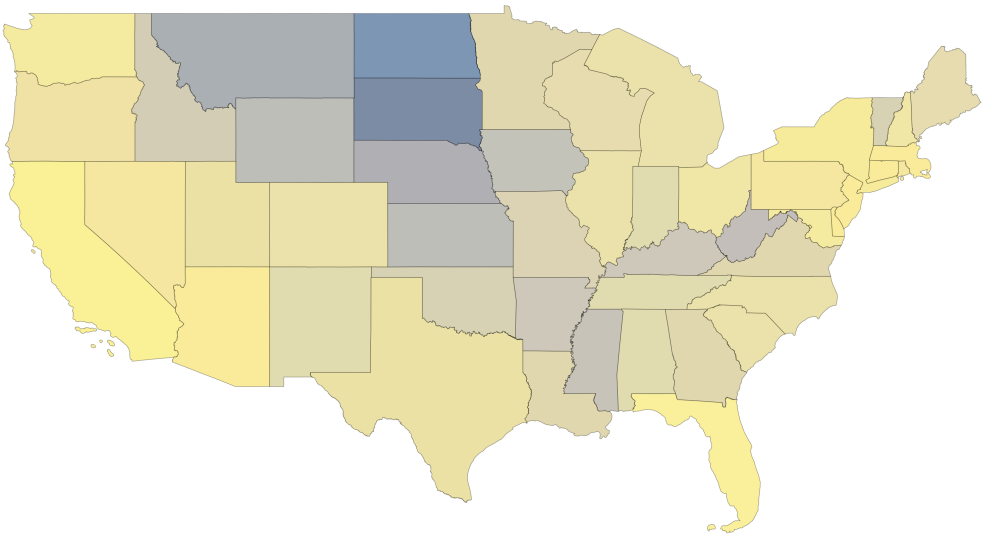


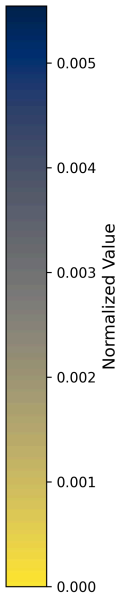
Normalized CHD_CrudePrev - United States Map



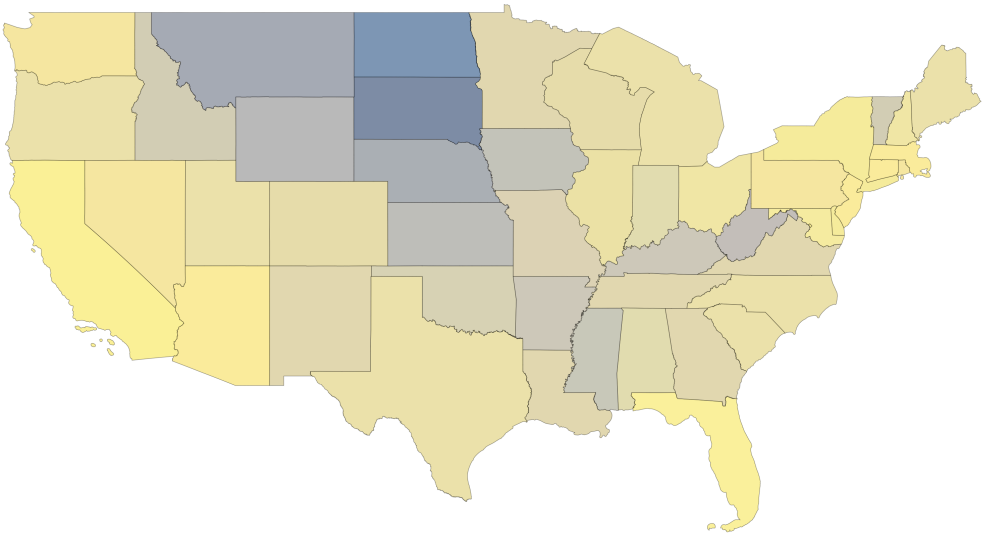


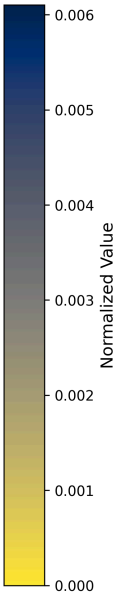
Normalized CHECKUP_CrudePrev - United States Map



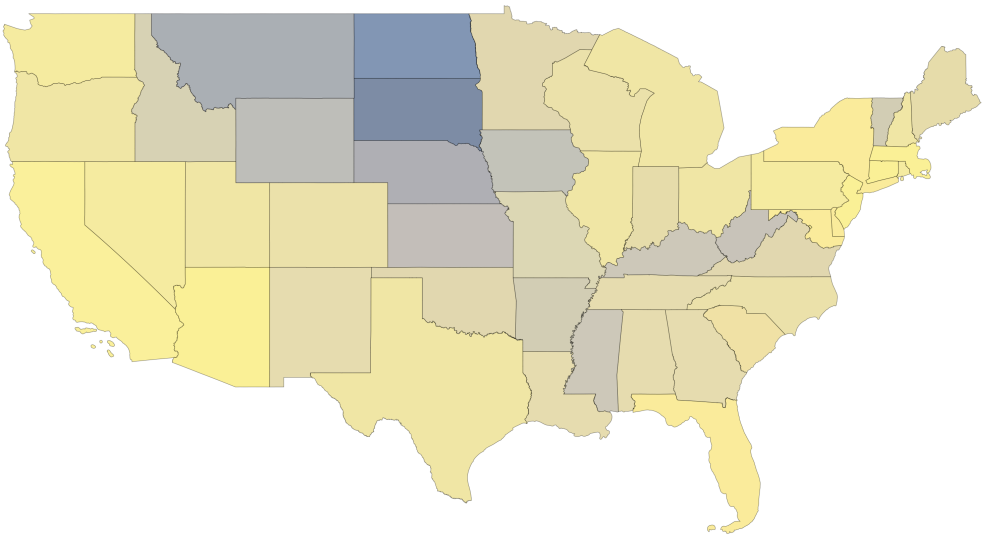


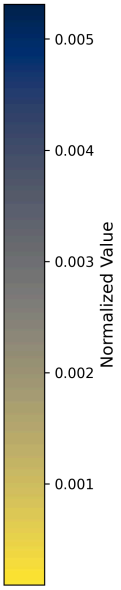
Normalized CHOLSCREEN_CrudePrev - United States Map



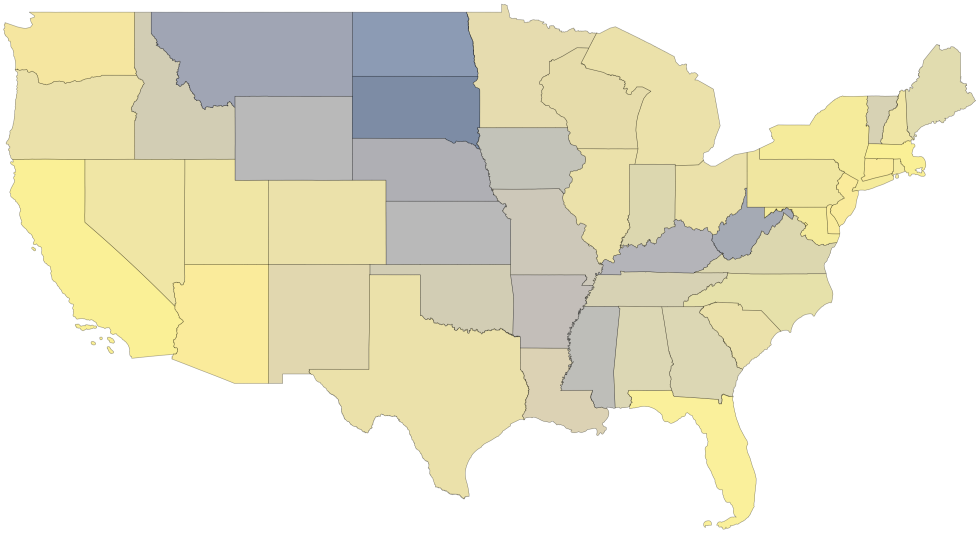


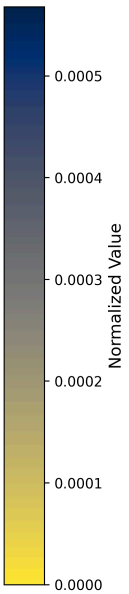
Normalized COLON_SCREEN_CrudePrev - United States Map



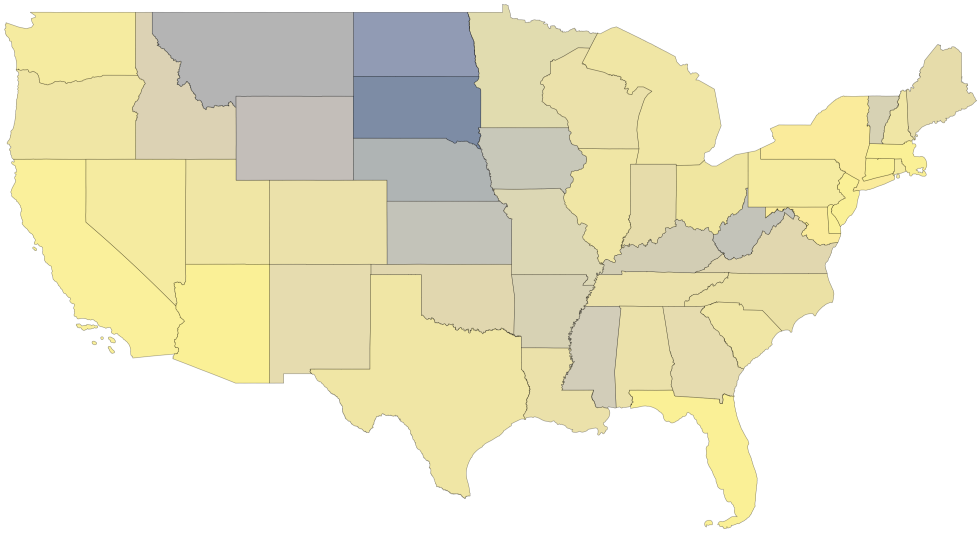


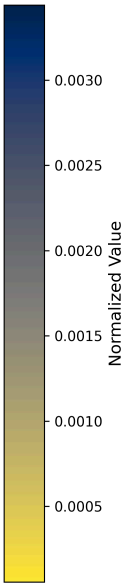
Normalized COPD_CrudePrev - United States Map



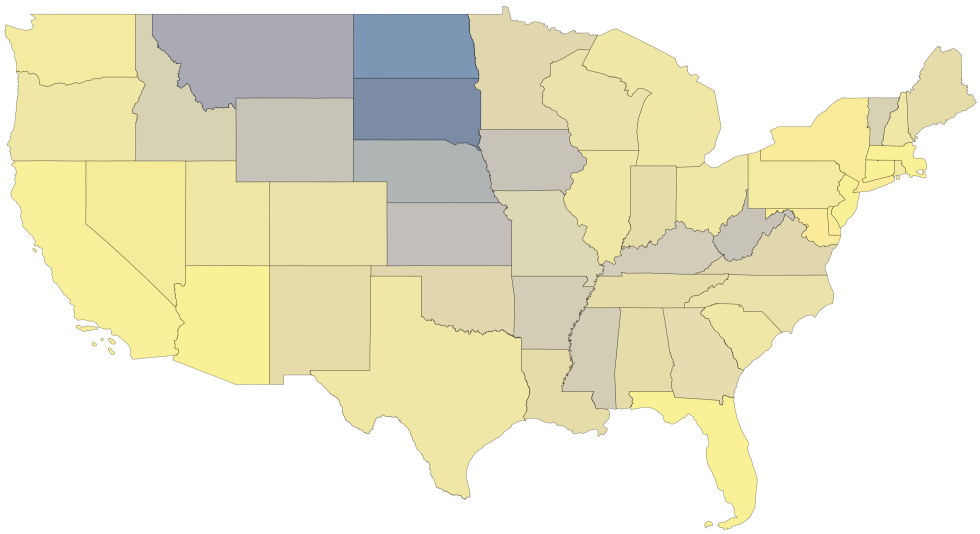


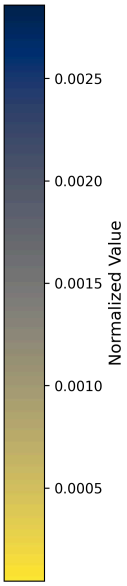
Normalized COREM_CrudePrev - United States Map



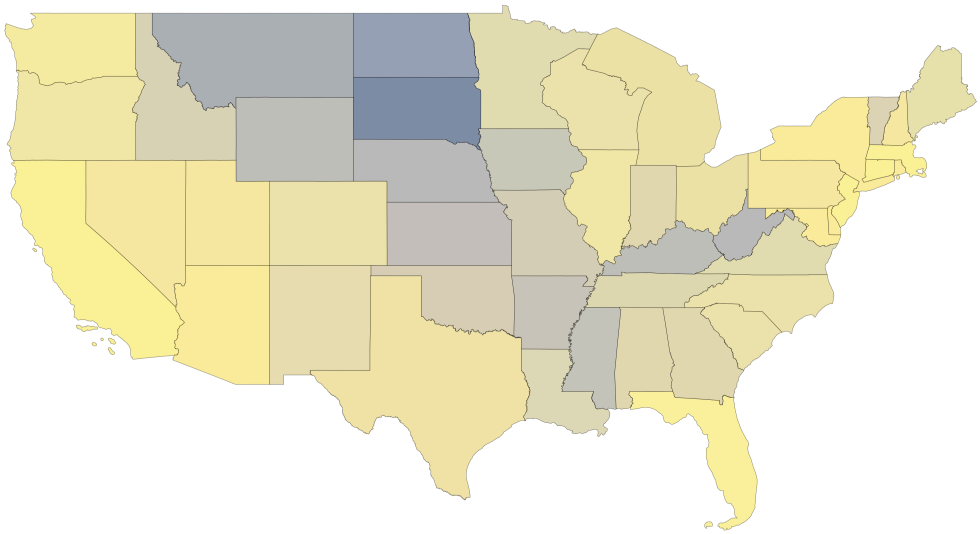


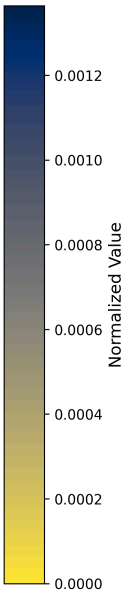
Normalized COREW_CrudePrev - United States Map



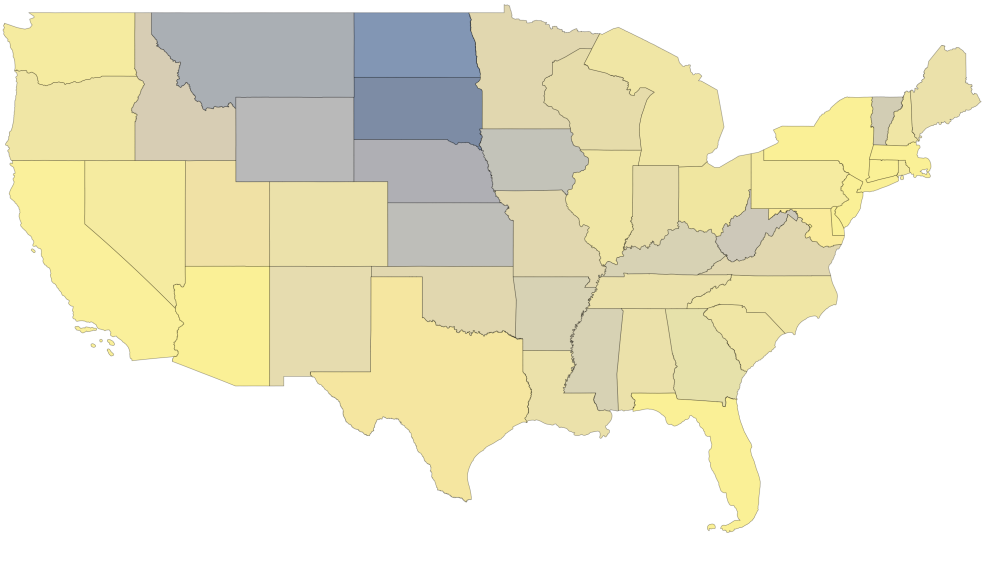


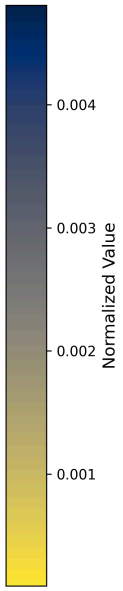
Normalized CSMOKING_CrudePrev - United States Map



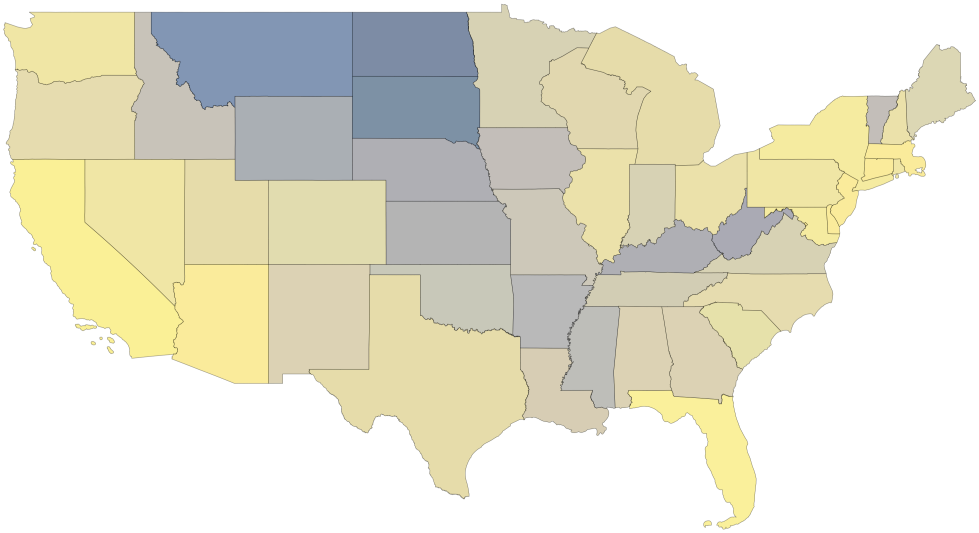


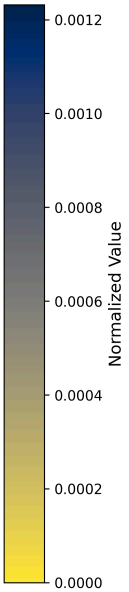
Normalized DENTAL_CrudePrev - United States Map



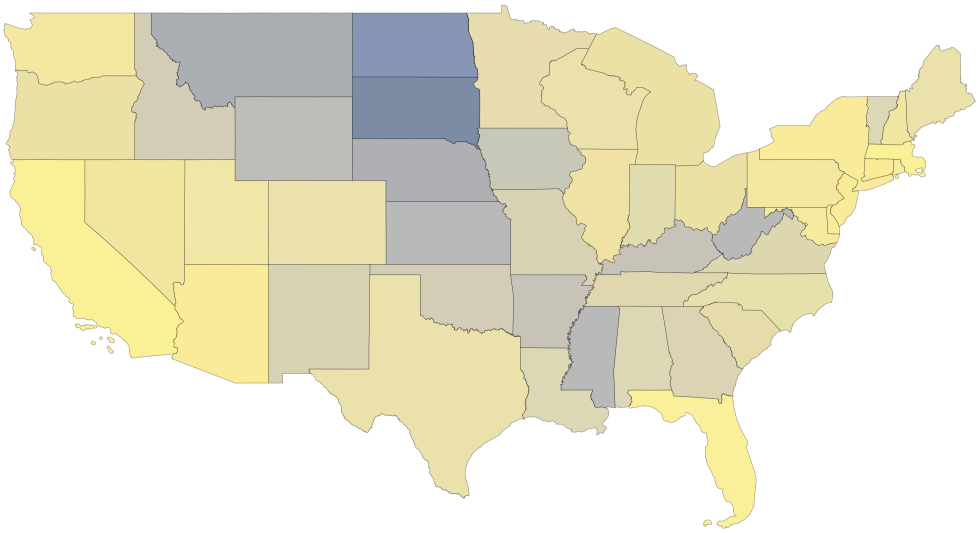


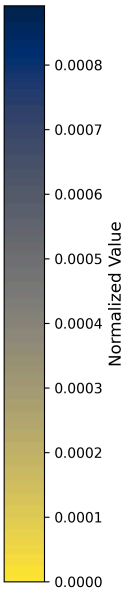
Normalized DEPRESSION_CrudePrev - United States Map



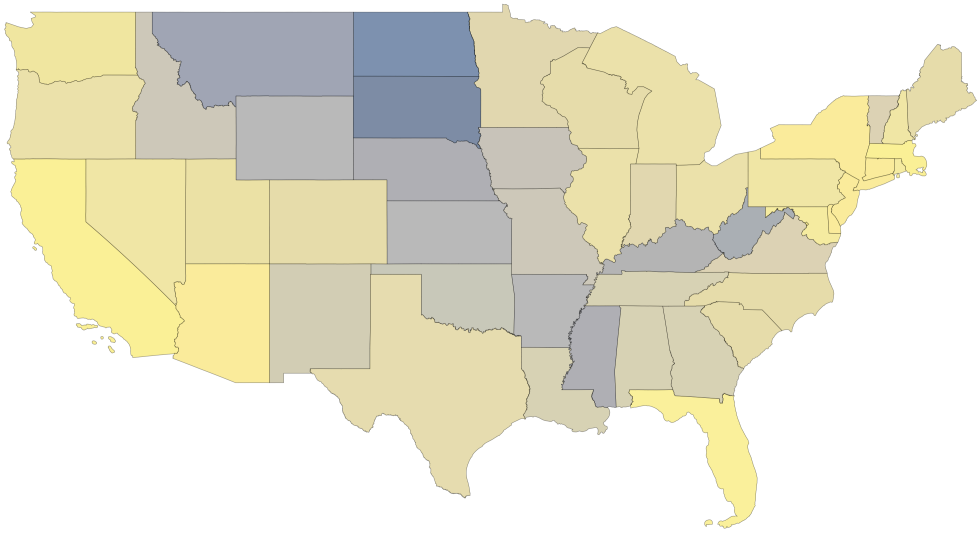


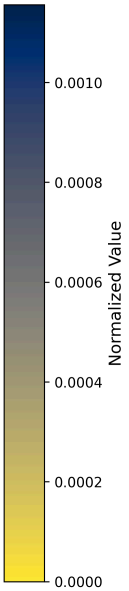
Normalized DIABETES_CrudePrev - United States Map



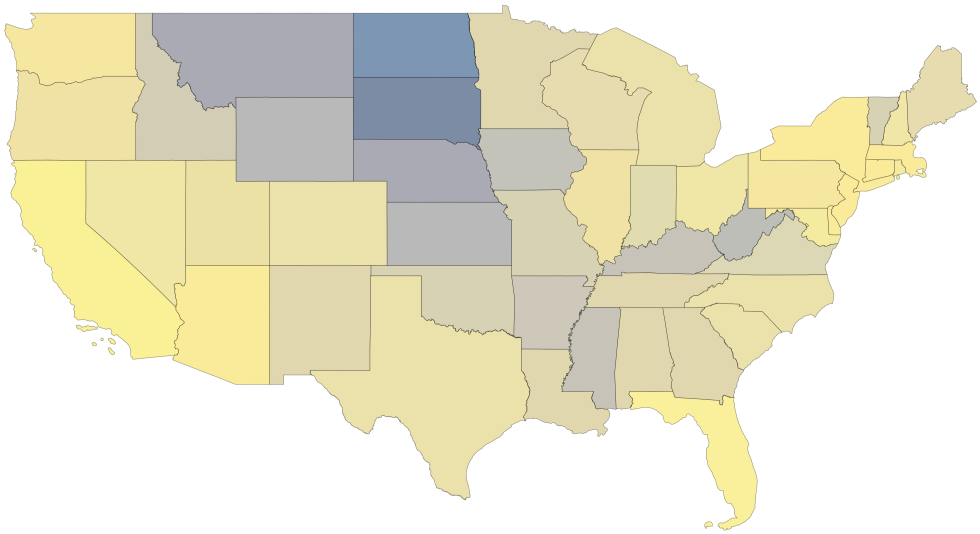


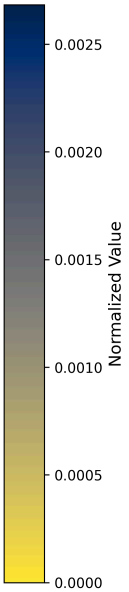
Normalized GHLTH_CrudePrev - United States Map



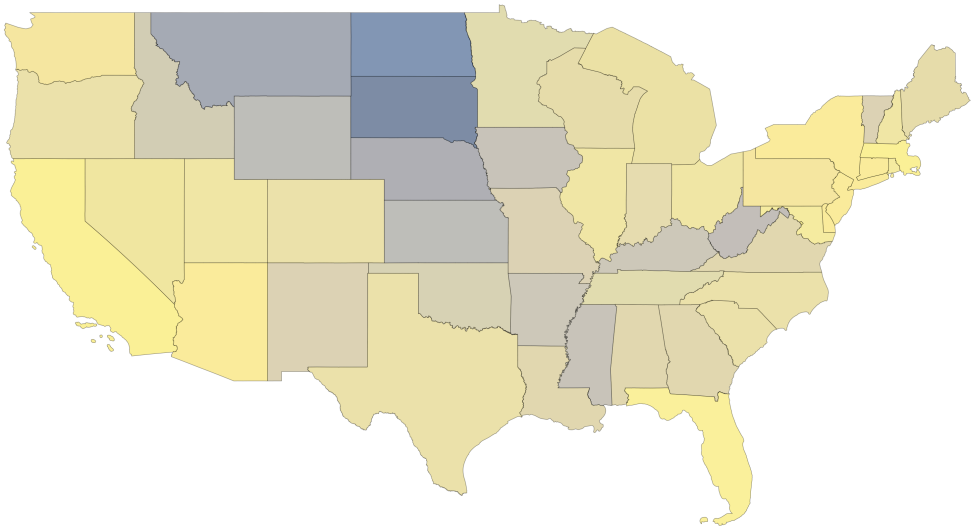


Normalized HIGHCHOL_CrudePrev - United States Map

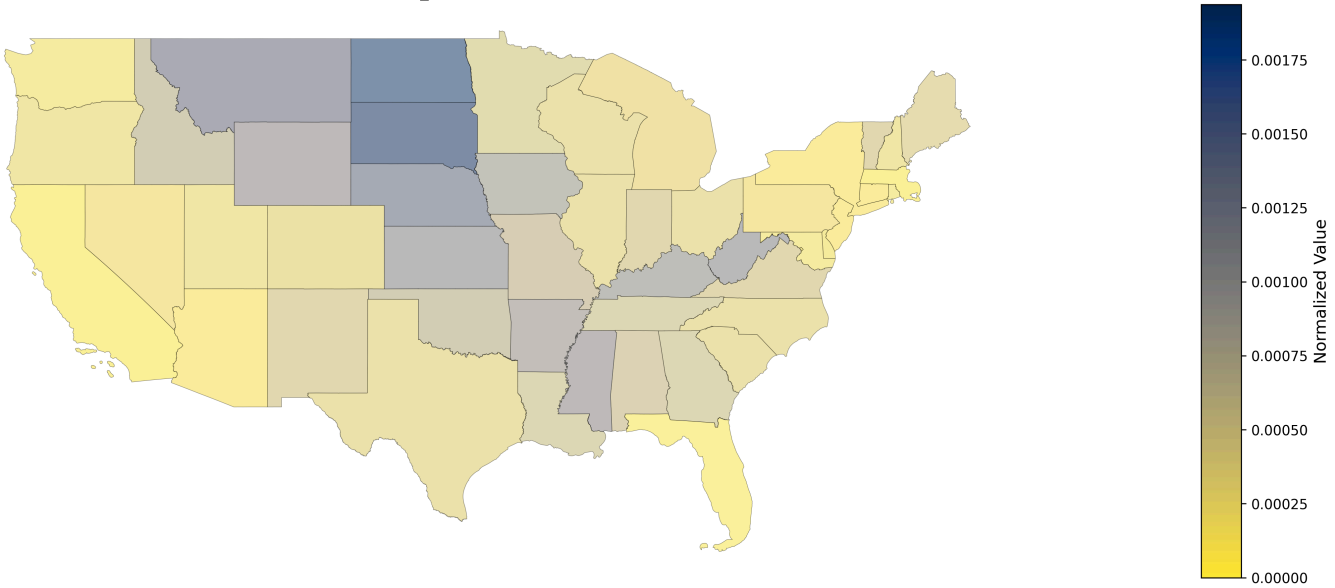




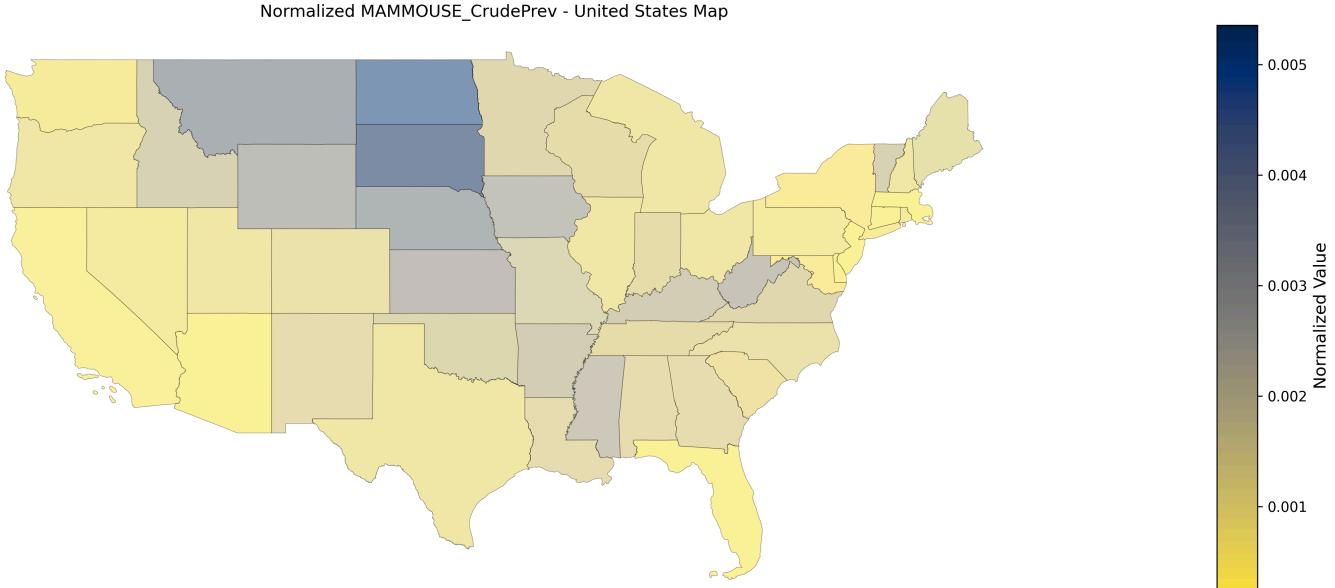
Normalized KIDNEY_CrudePrev - United States Map



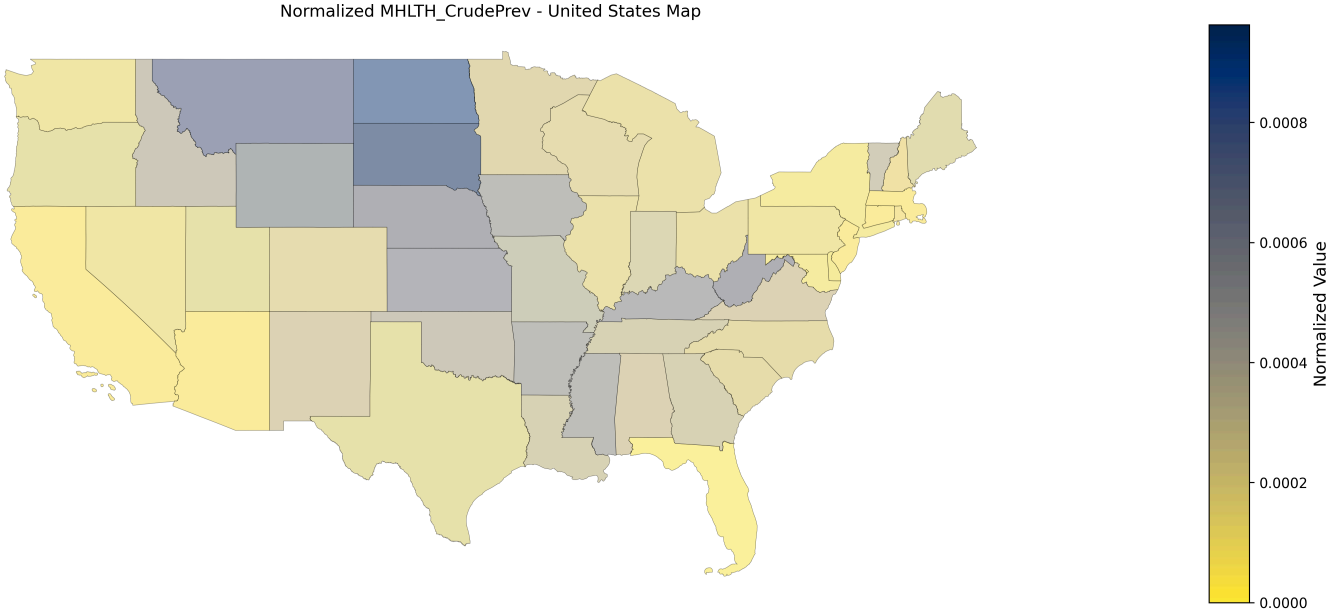
Normalized LPA_CrudePrev - United States Map



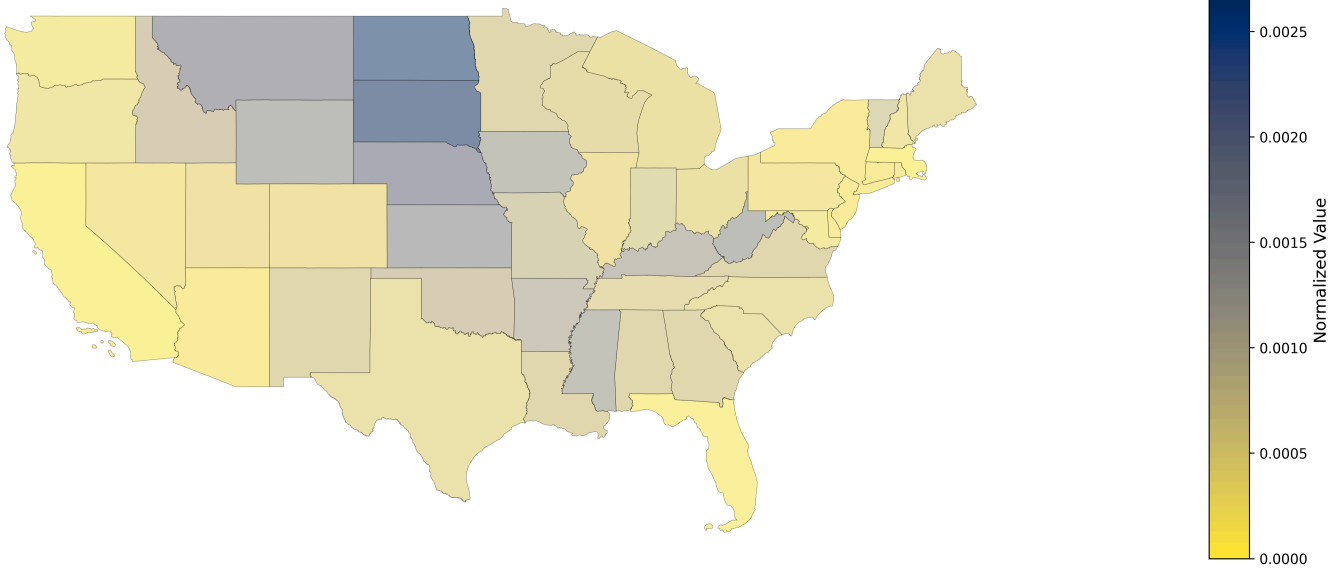
Normalized MAMMOUSE_CrudePrev - United States Map



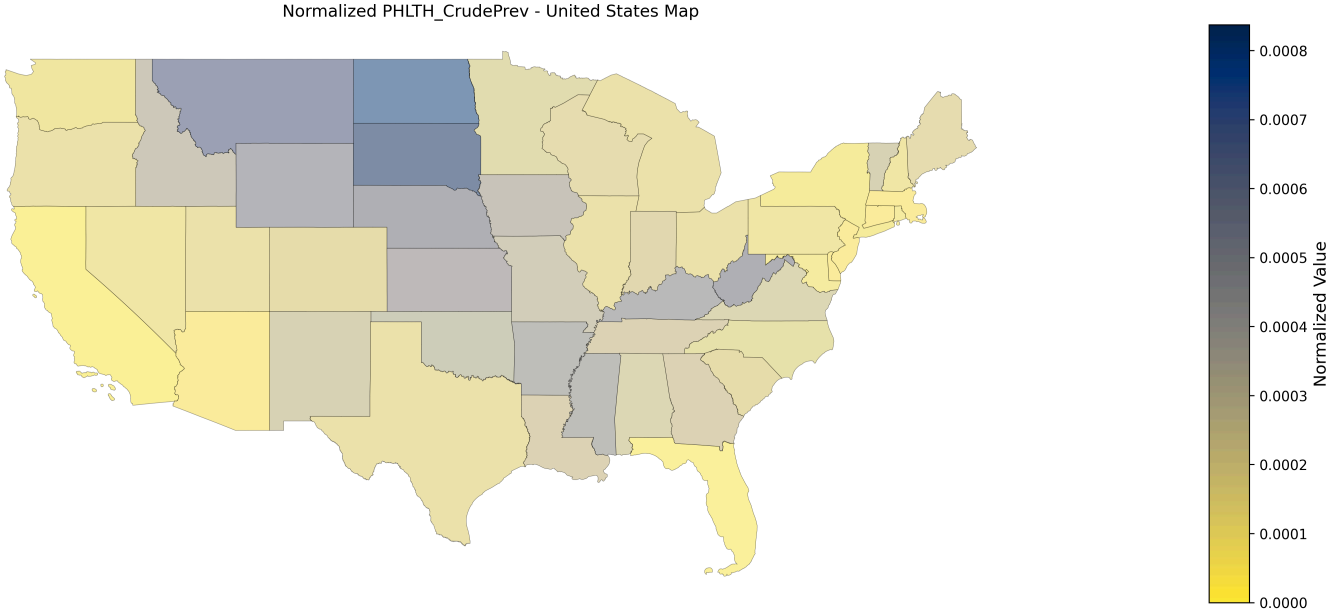
Normalized MHLTH_CrudePrev - United States Map



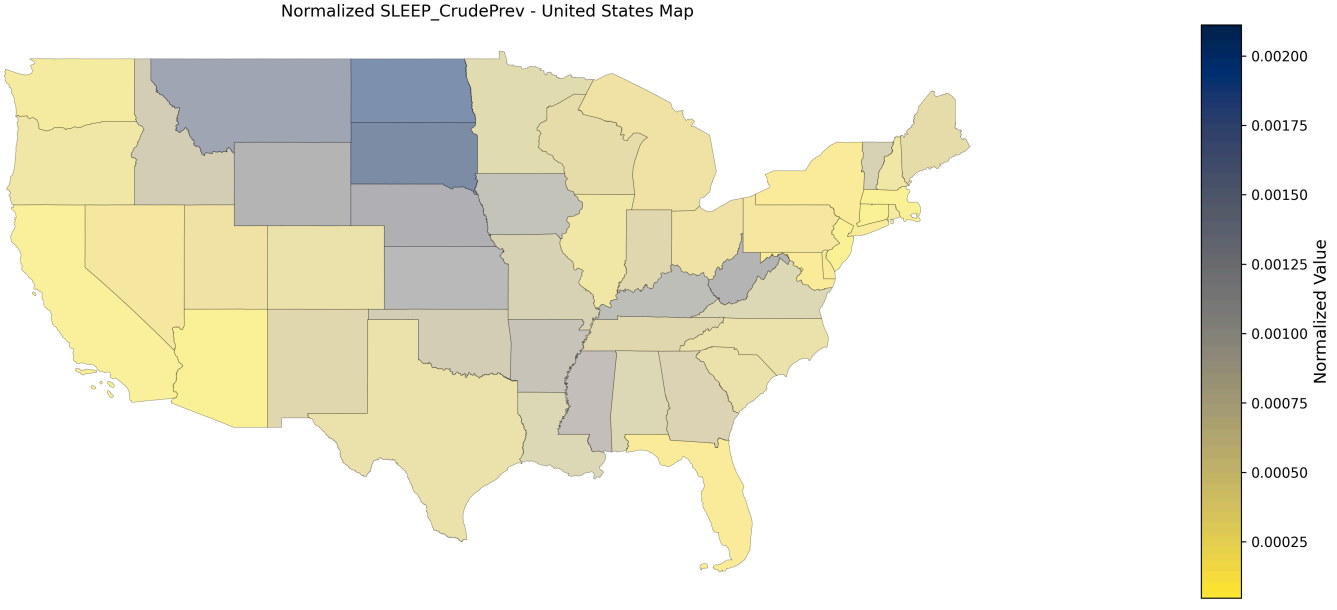
Normalized OBESITY_CrudePrev - United States Map



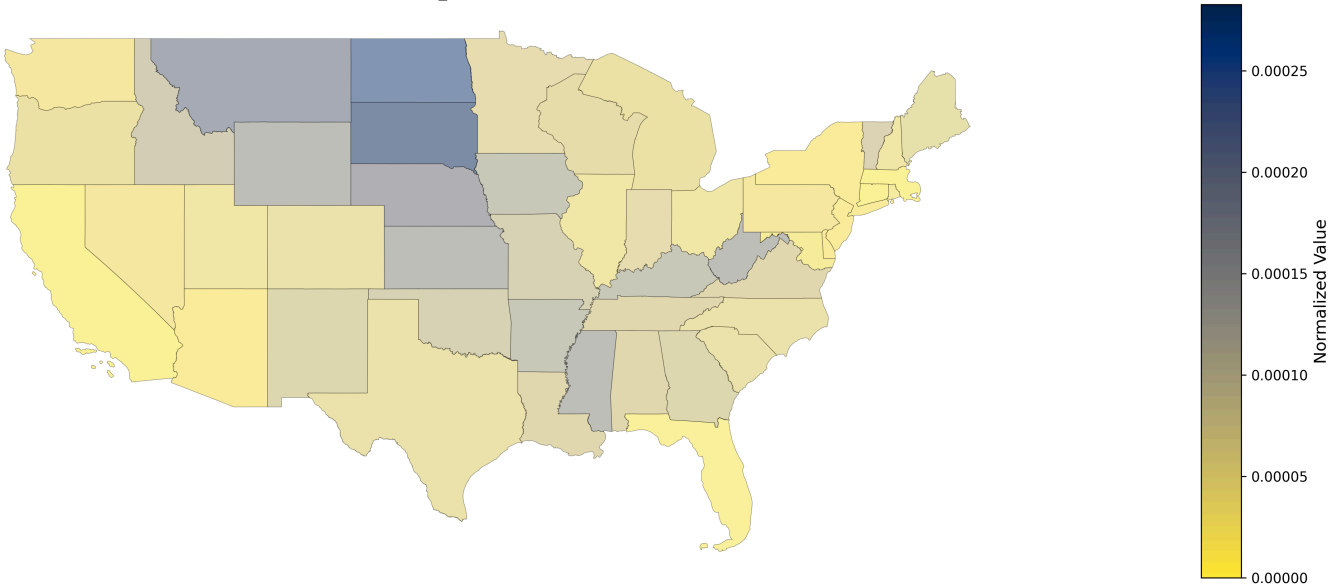
Normalized PHLTH_CrudePrev - United States Map



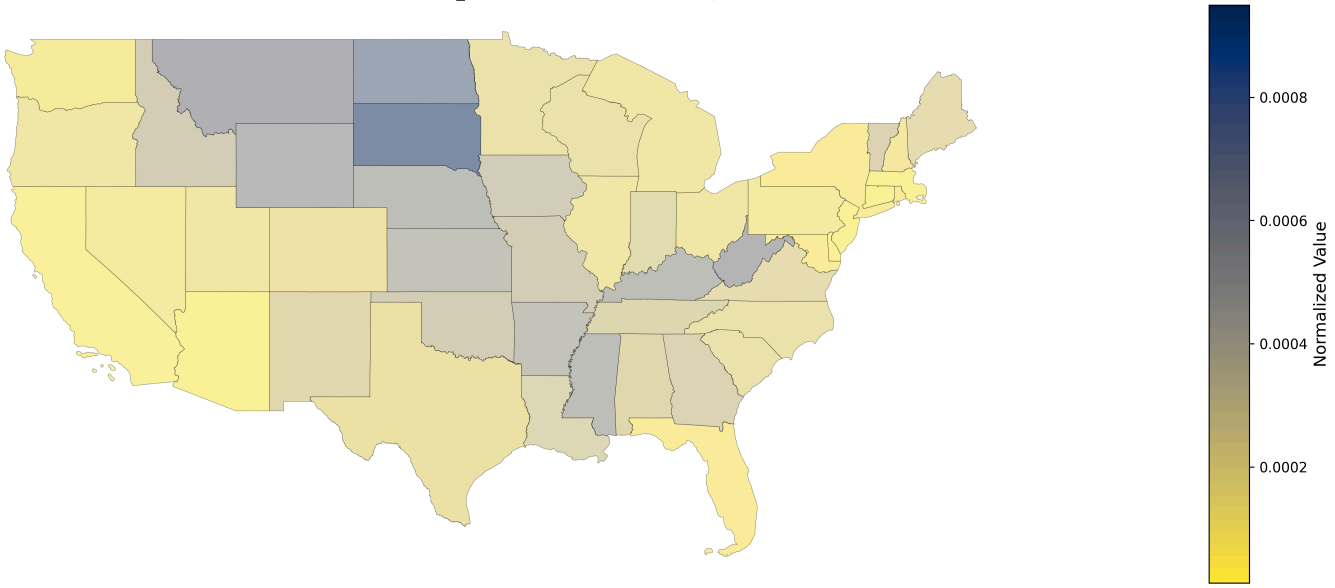
Normalized SLEEP_CrudePrev - United States Map



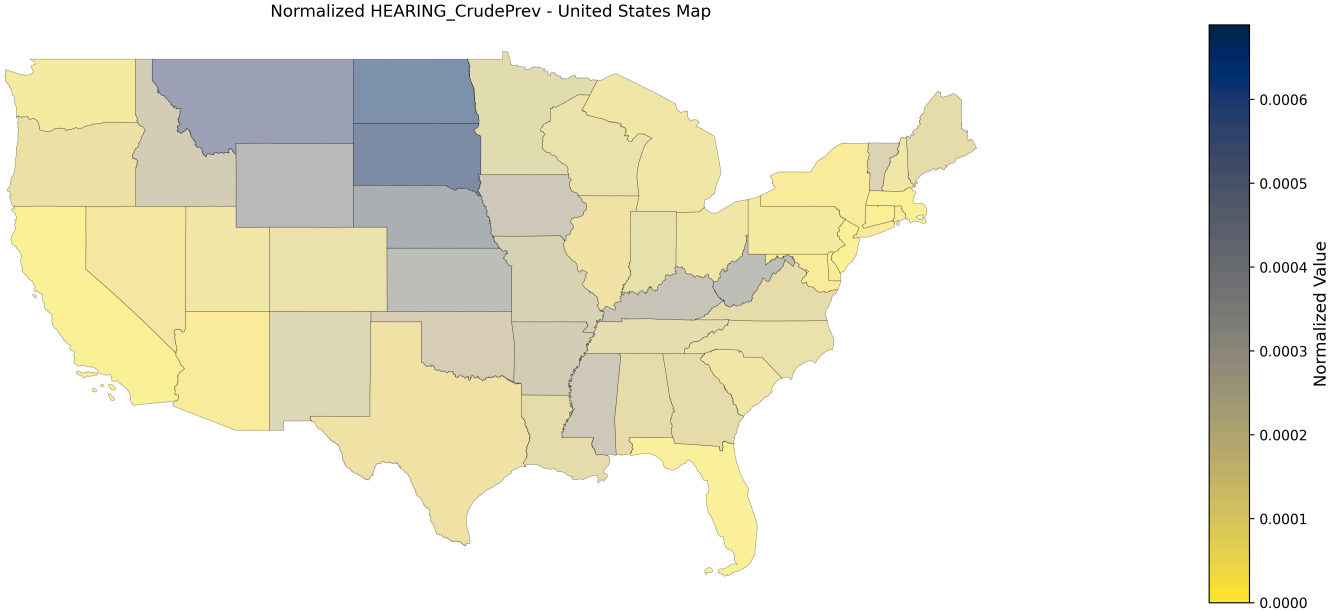
Normalized STROKE_CrudePrev - United States Map



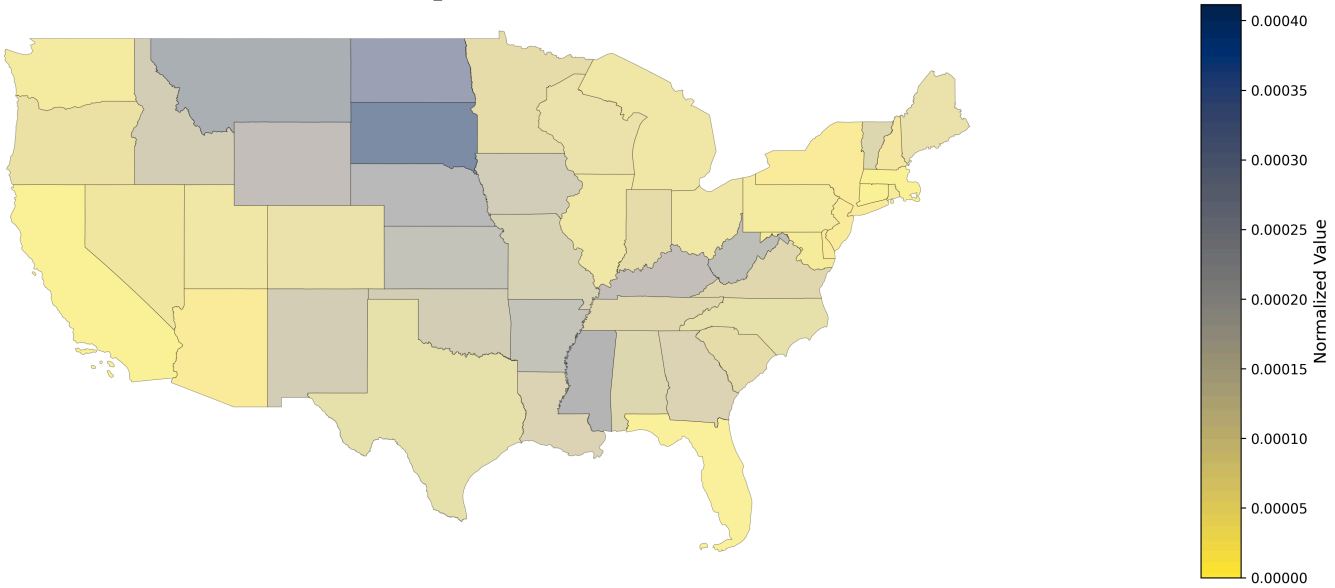
Normalized TEETHLOST_CrudePrev - United States Map



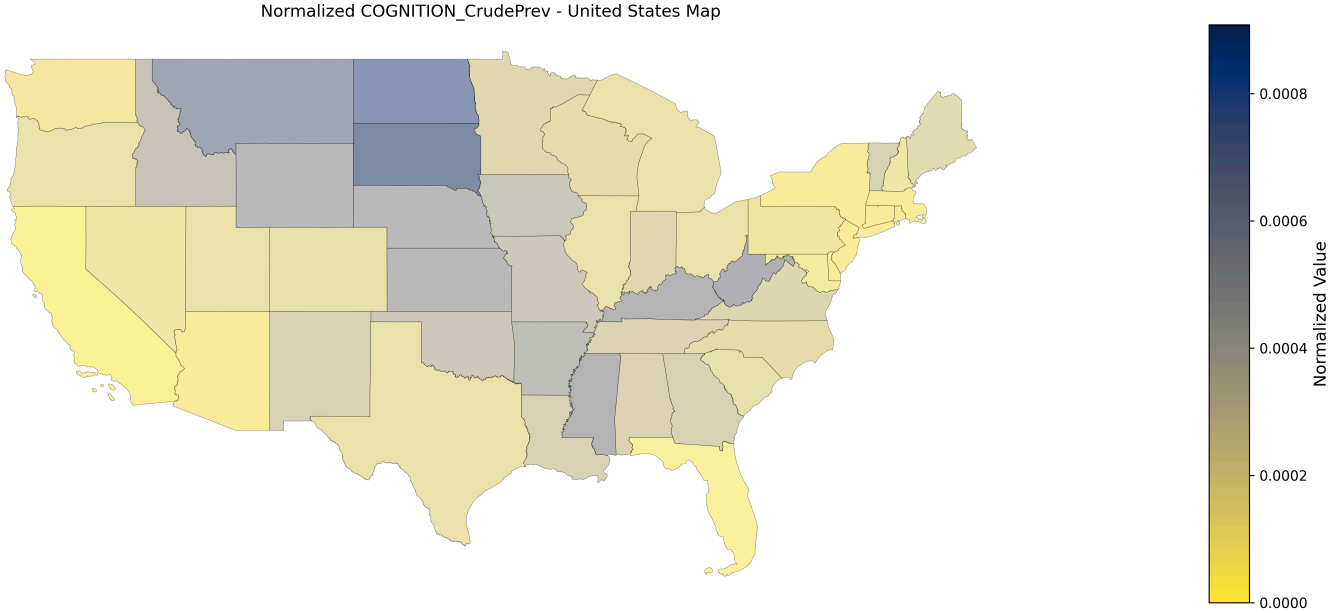
Normalized HEARING_CrudePrev - United States Map



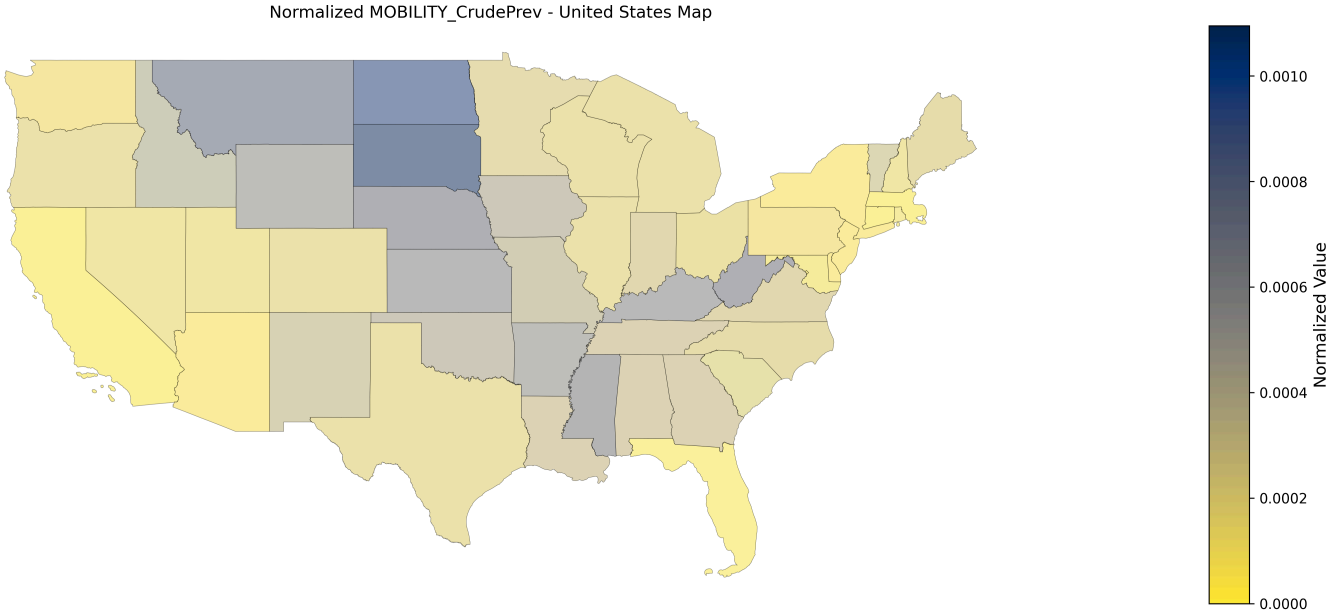
Normalized VISION_CrudePrev - United States Map

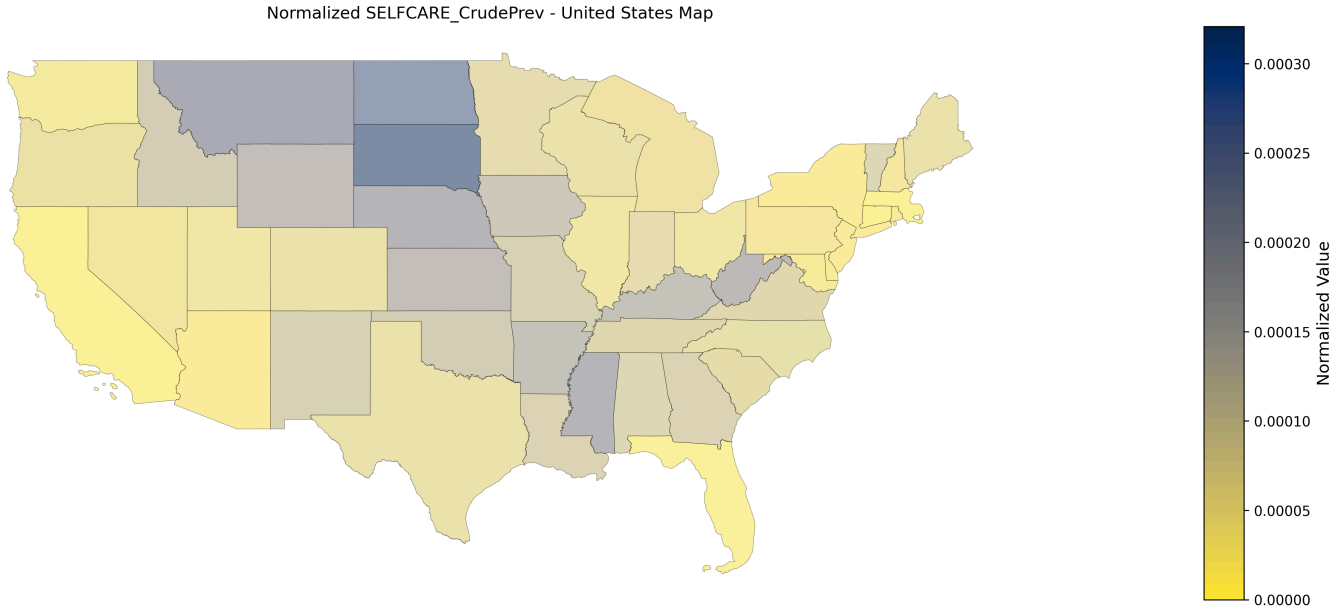


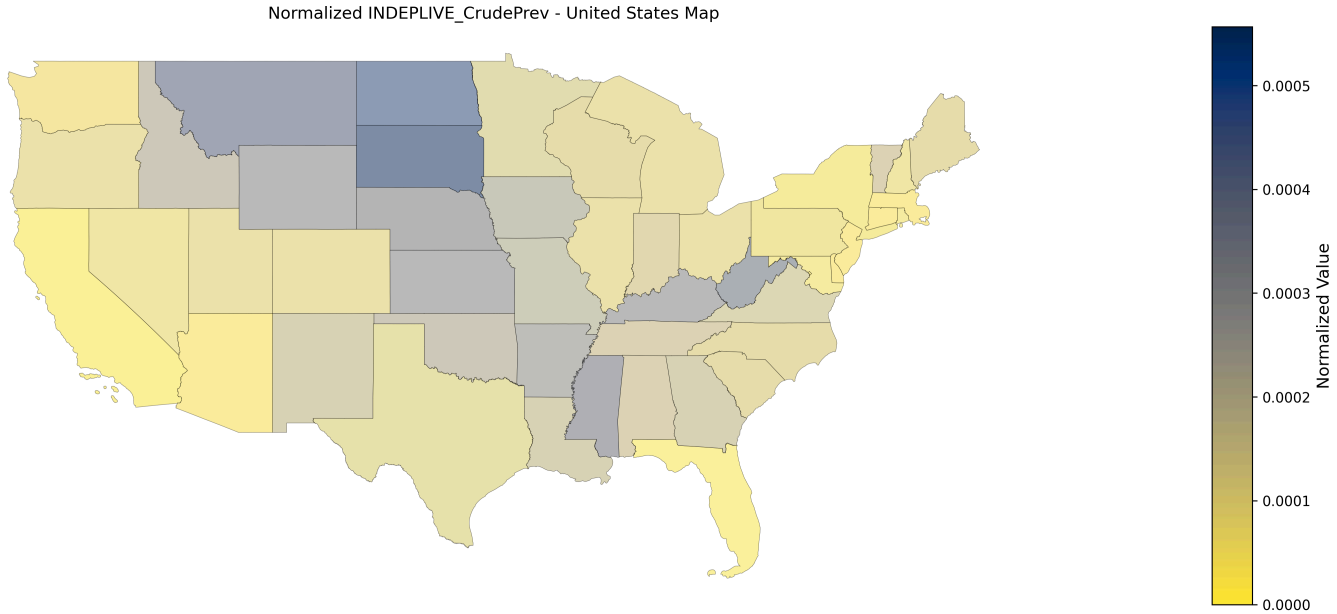
Normalized COGNITION_CrudePrev - United States Map

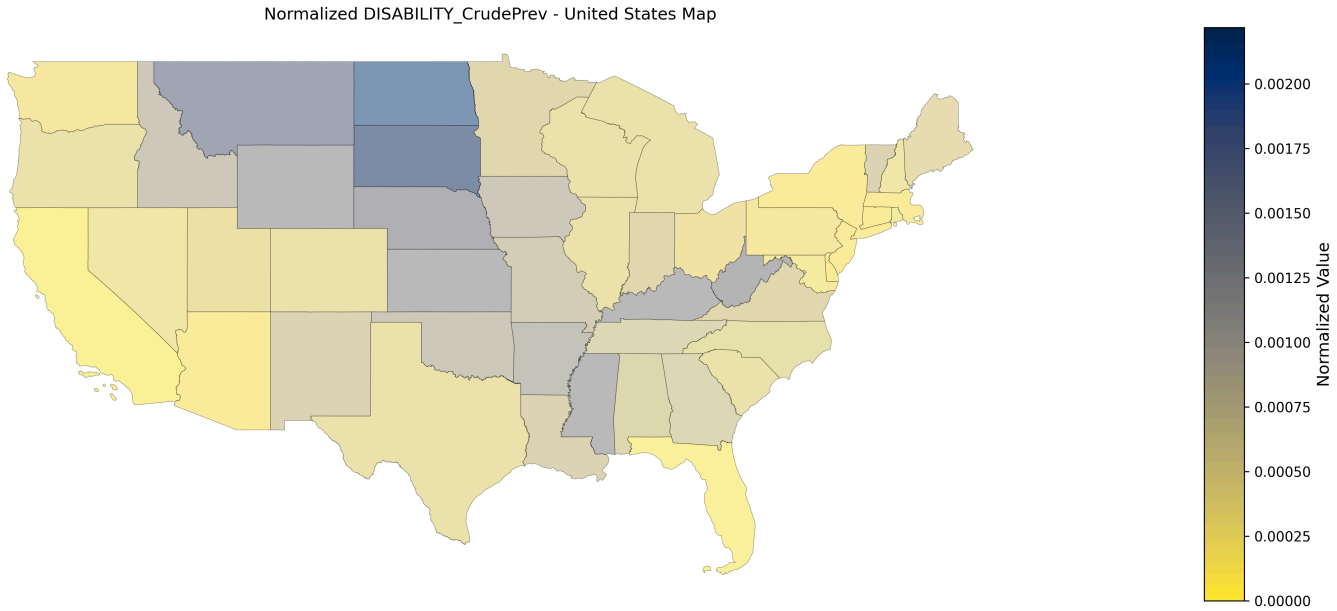


Normalized MOBILITY_CrudePrev - United States Map









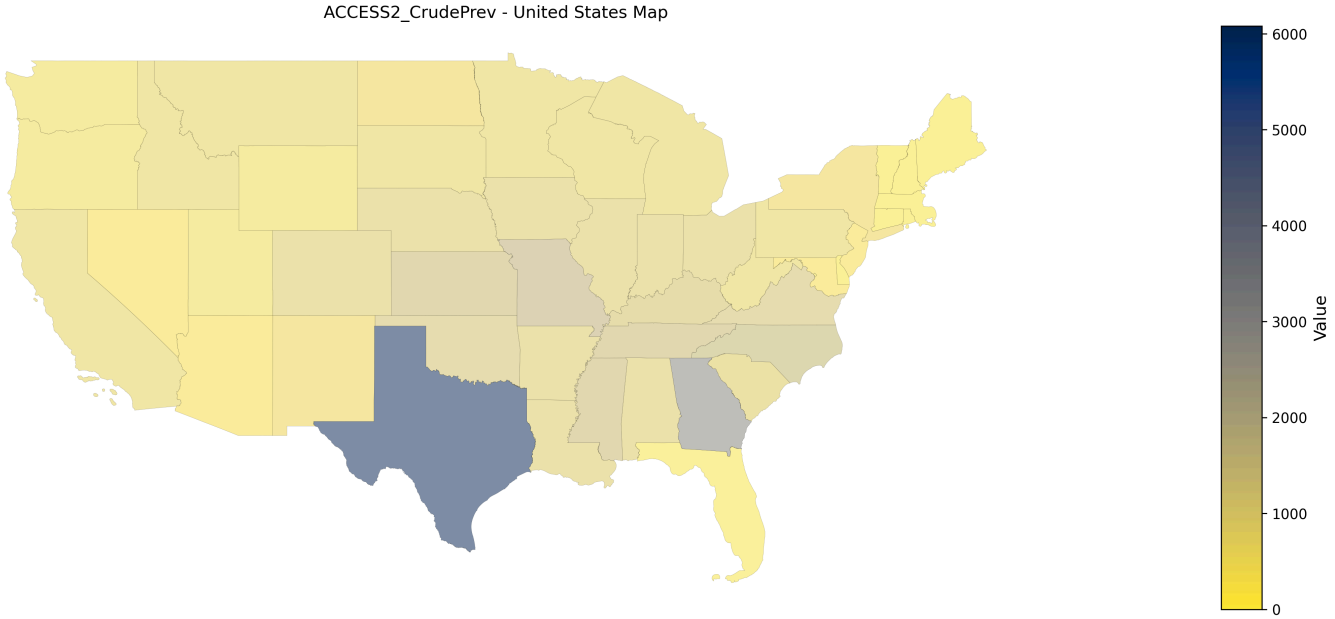
Function to Plot Multiple Not-Normalized Columns

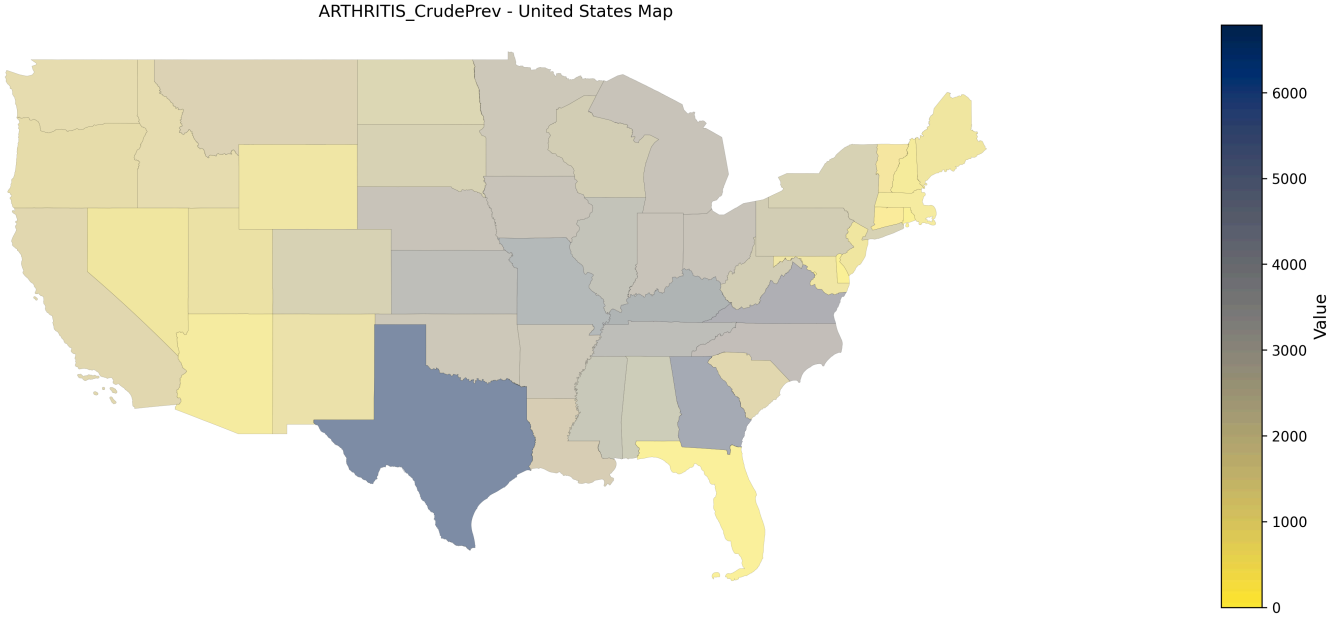
In [ ]:
```python
def plot_data_multiple_columns(geo_df, data_columns, colormap):
    for column in data_columns:
        # Plotting
        f, ax = plt.subplots(1, 1, figsize=(15, 10), sharex=True, sharey=True, dpi=300)
        f.tight_layout()
        plt.title(f'{column} - United States Map')
```

```
        ax.set_axis_off()
        divider = make_axes_locatable(ax)
        cax = divider.append_axes("right", size="3%", pad=0.5, alpha=0.5)
        geo_df.plot(column, ax=ax, alpha=0.5, cmap=colormap, edgecolor='k', legend=True, cax=cax, linewidth=0.1)
        plt.ylabel('Value', fontsize=12)
        plt.show()
```

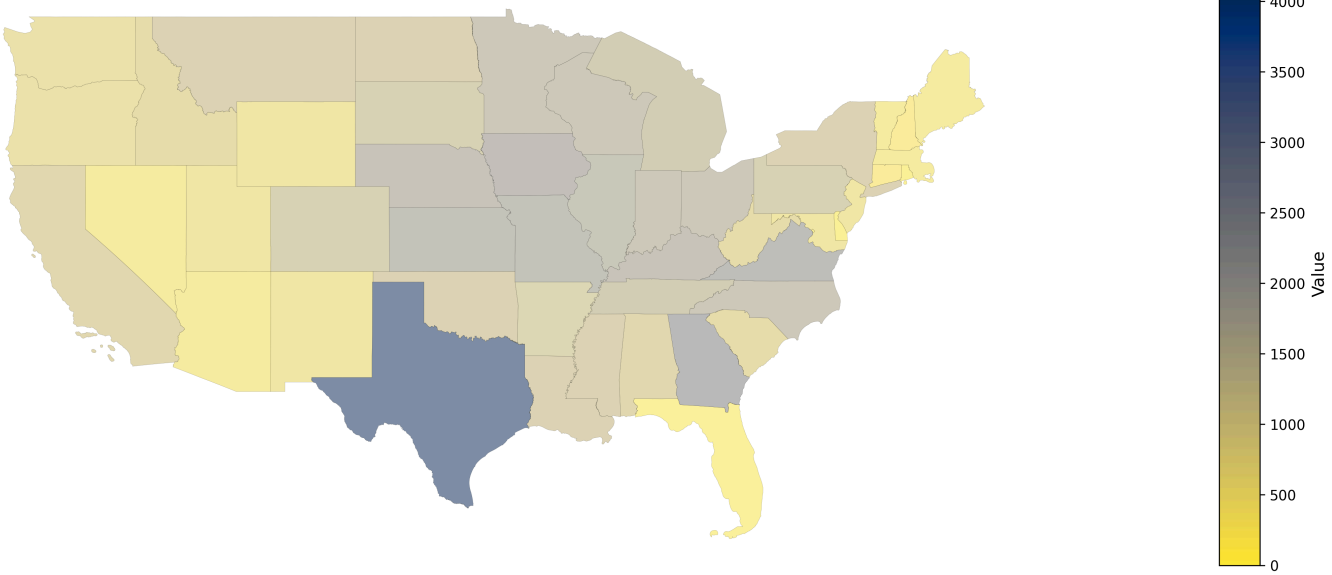
In [ ]:

```
# Get columns ending with '_CrudePrev'
columns_to_plot = [col for col in df.columns if col.endswith('_CrudePrev')]

plot_data_multiple_columns(merged_gdf, columns_to_plot, 'cividis_r')
```

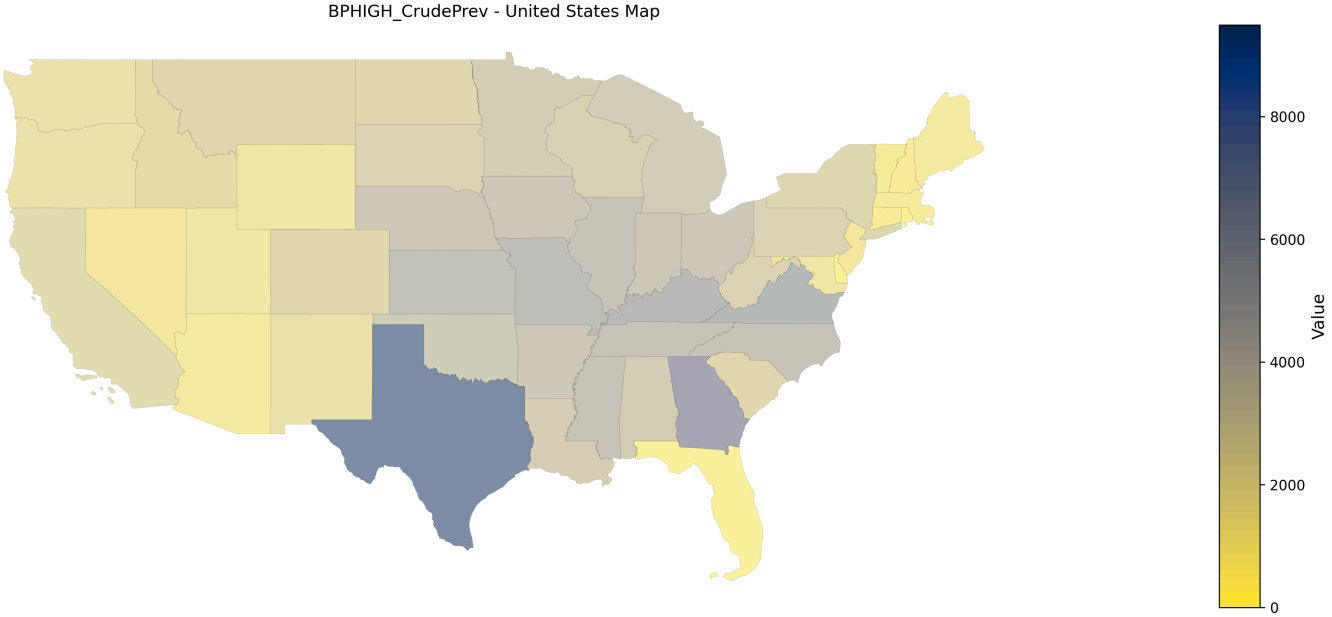




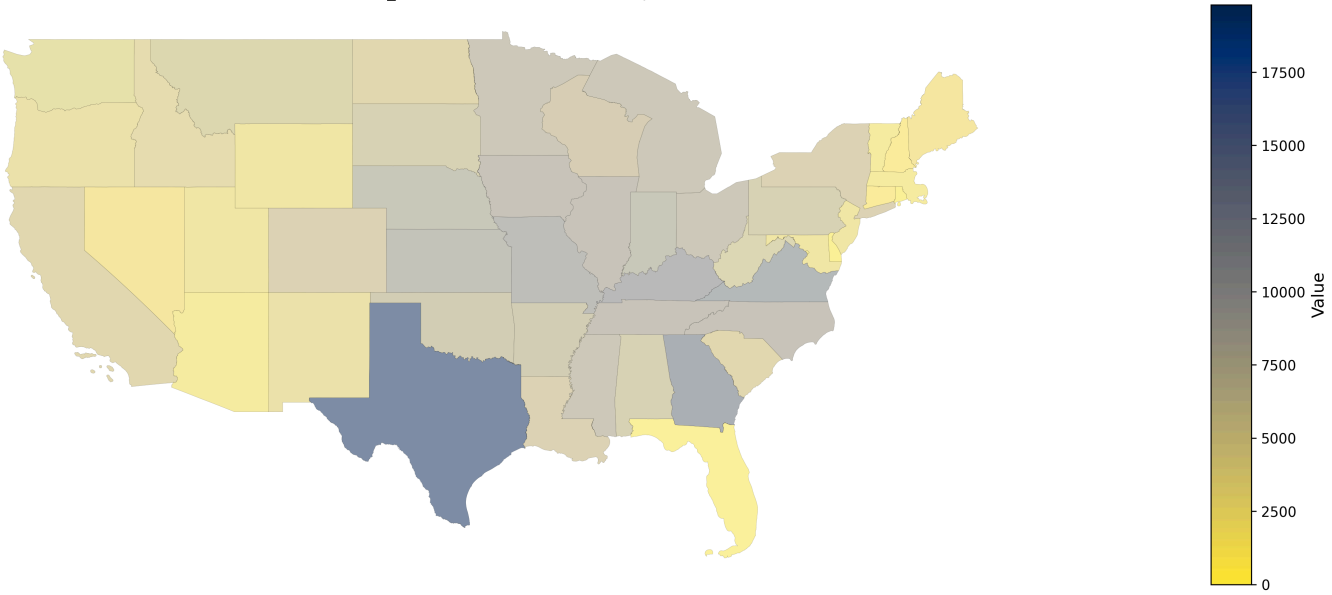
BINGE_CrudePrev - United States Map



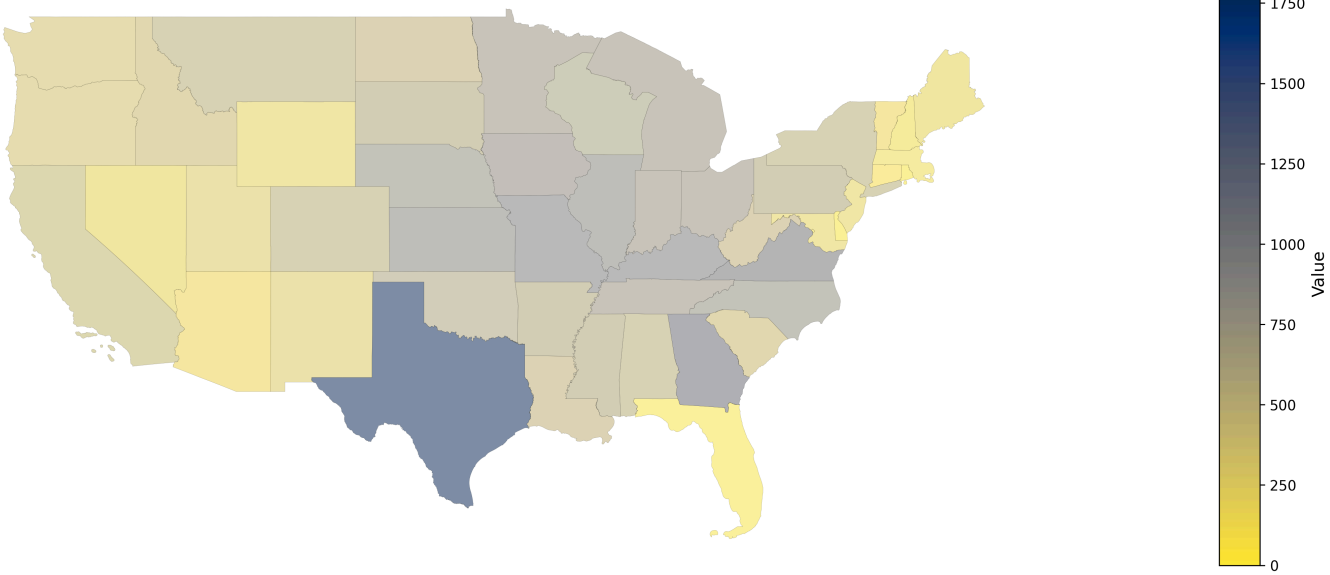
BPHIGH_CrudePrev - United States Map



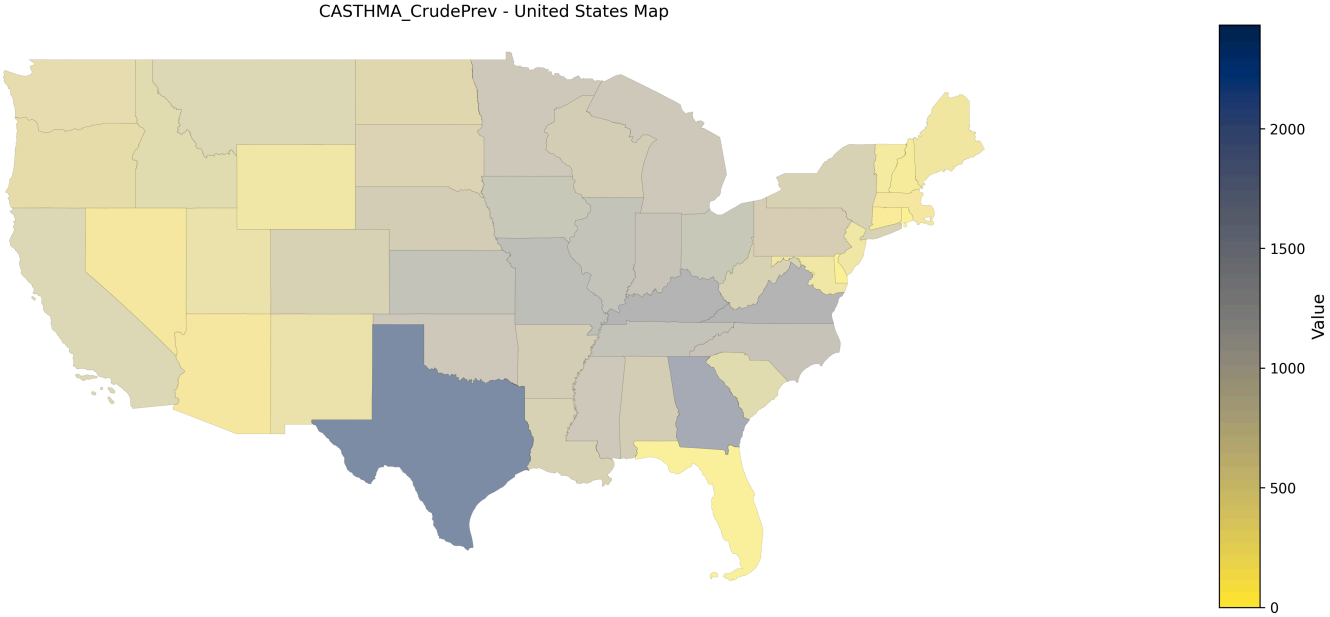
BPMED_CrudePrev - United States Map



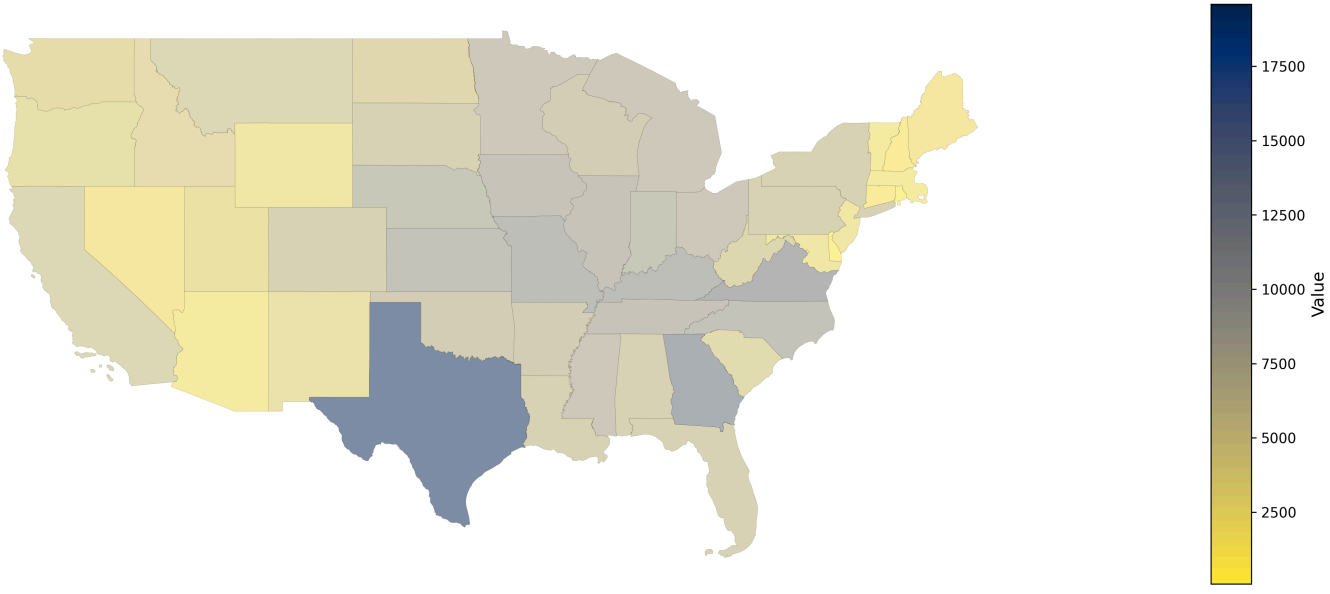
CANCER_CrudePrev - United States Map



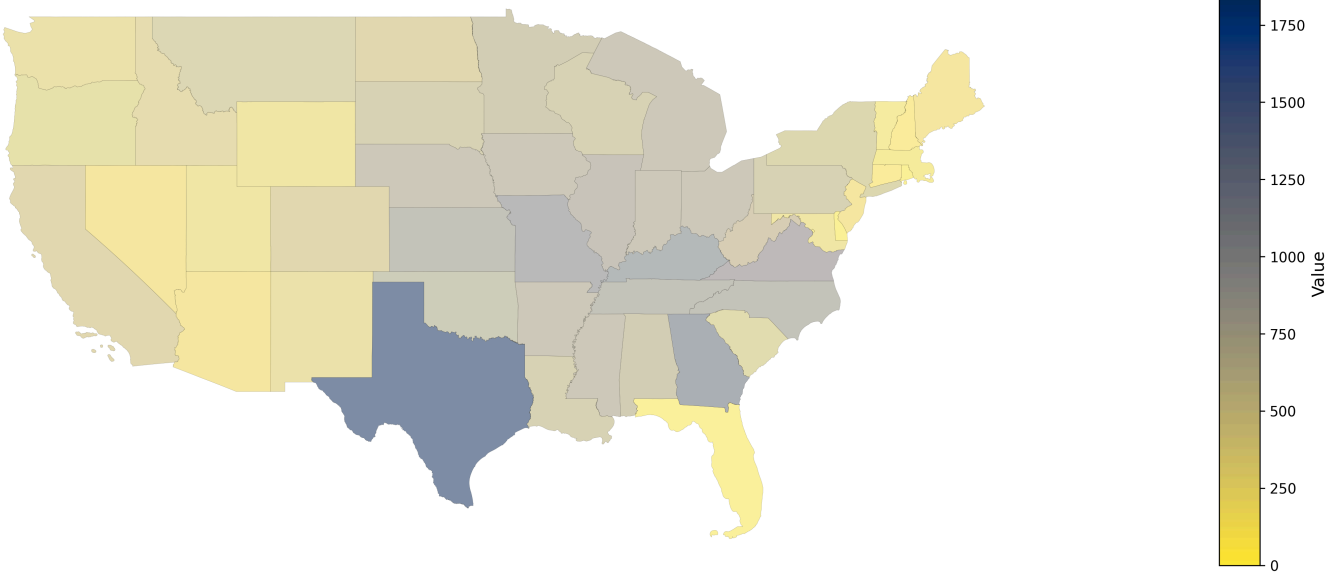
CASTHMA_CrudePrev - United States Map



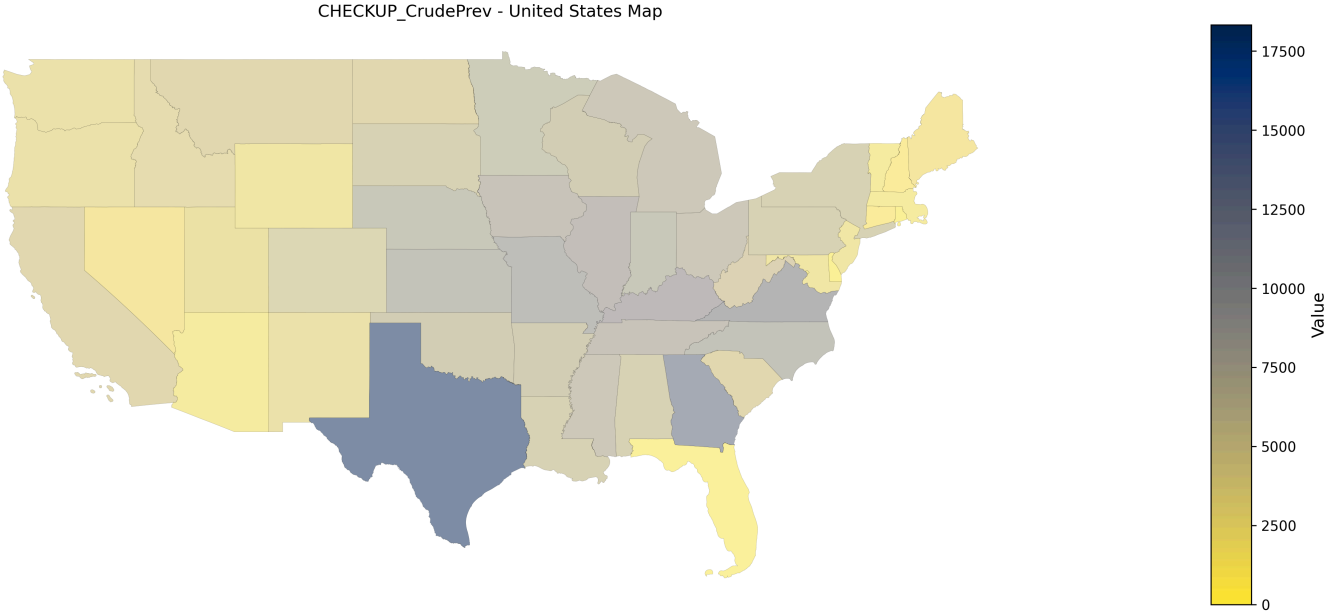
CERVICAL_CrudePrev - United States Map



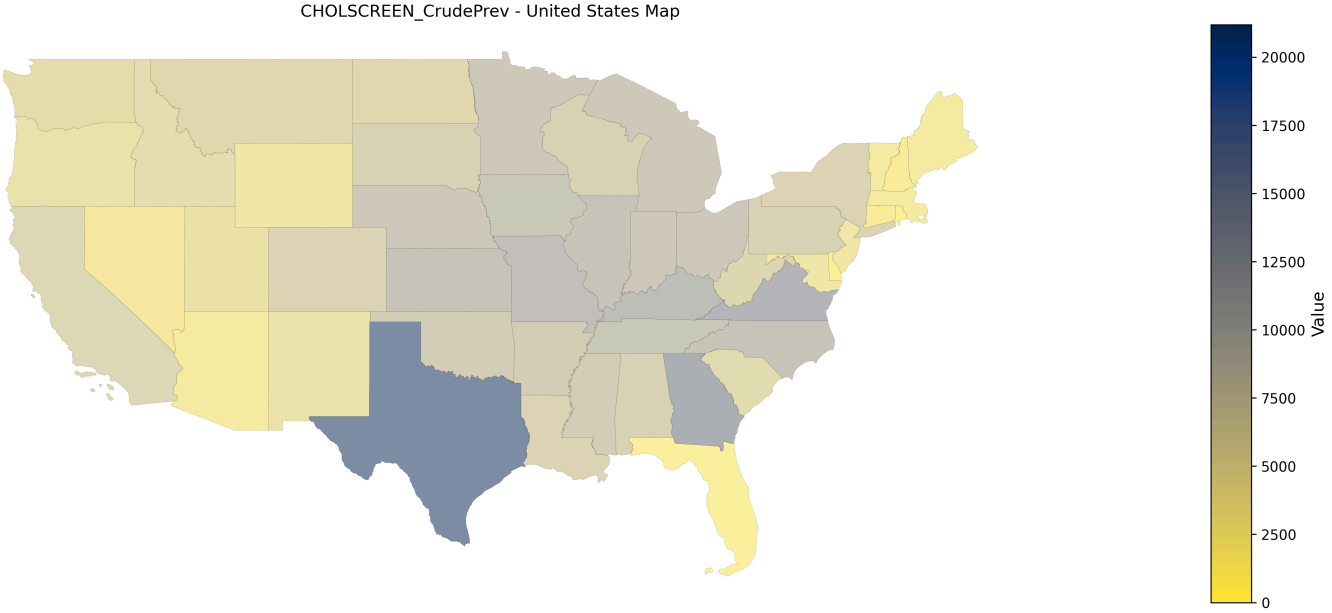
CHD_CrudePrev - United States Map



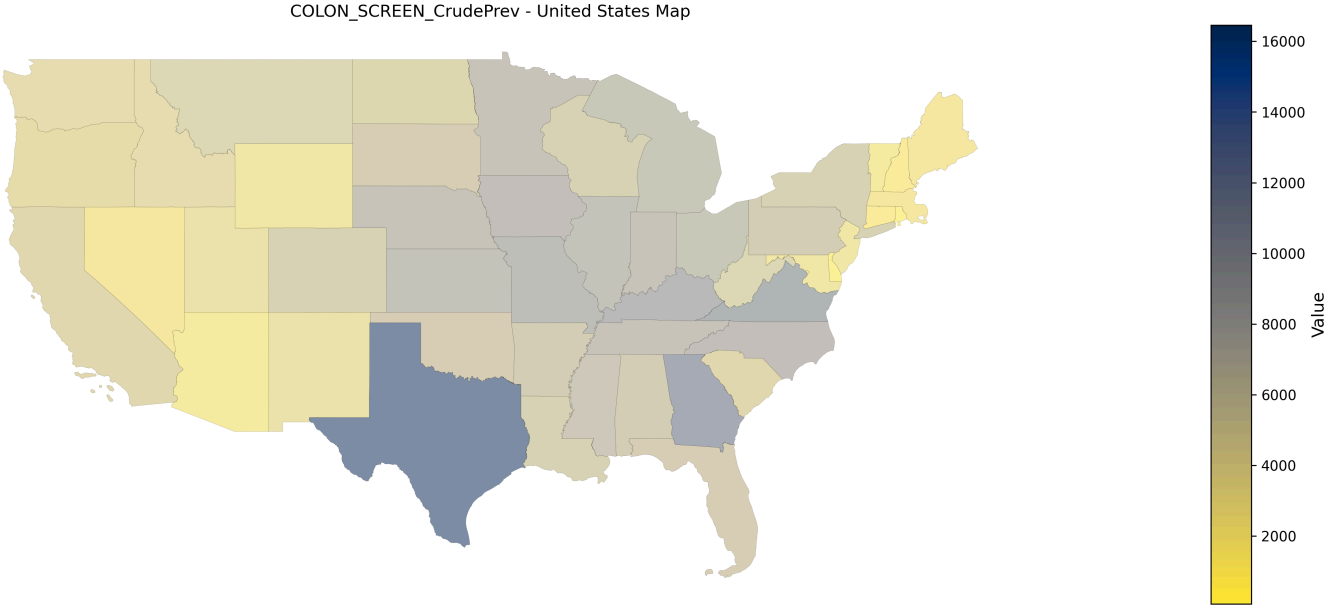
CHECKUP_CrudePrev - United States Map



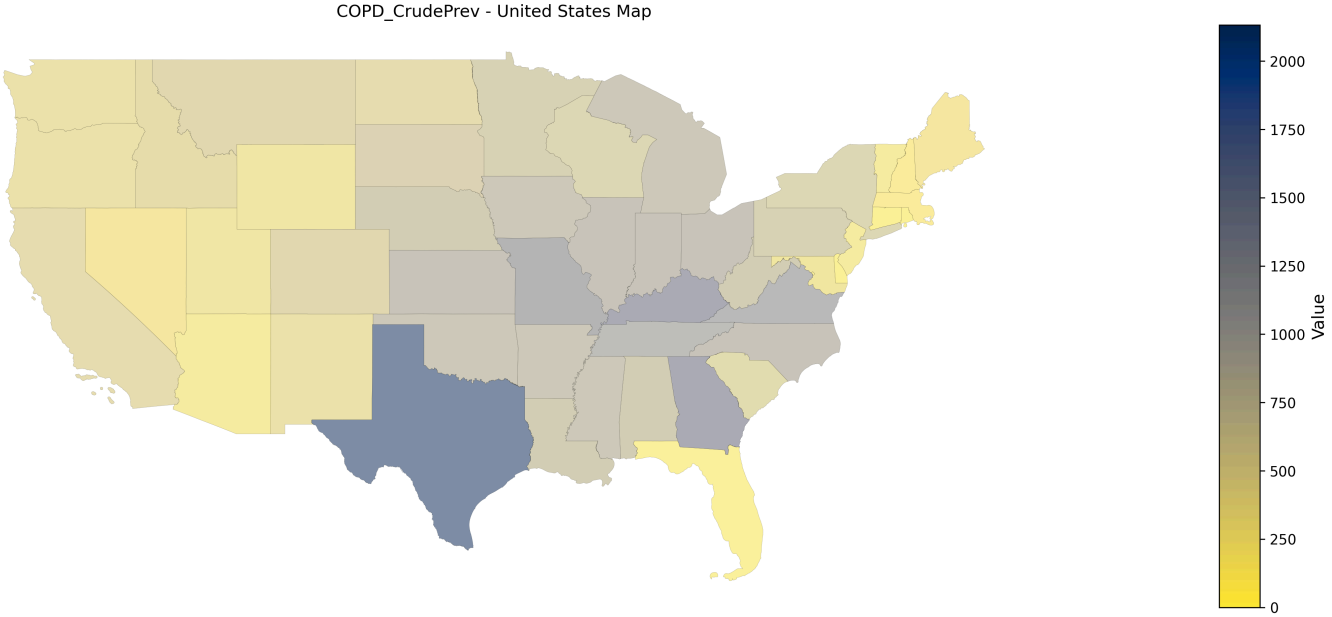
CHOLSCREEN_CrudePrev - United States Map



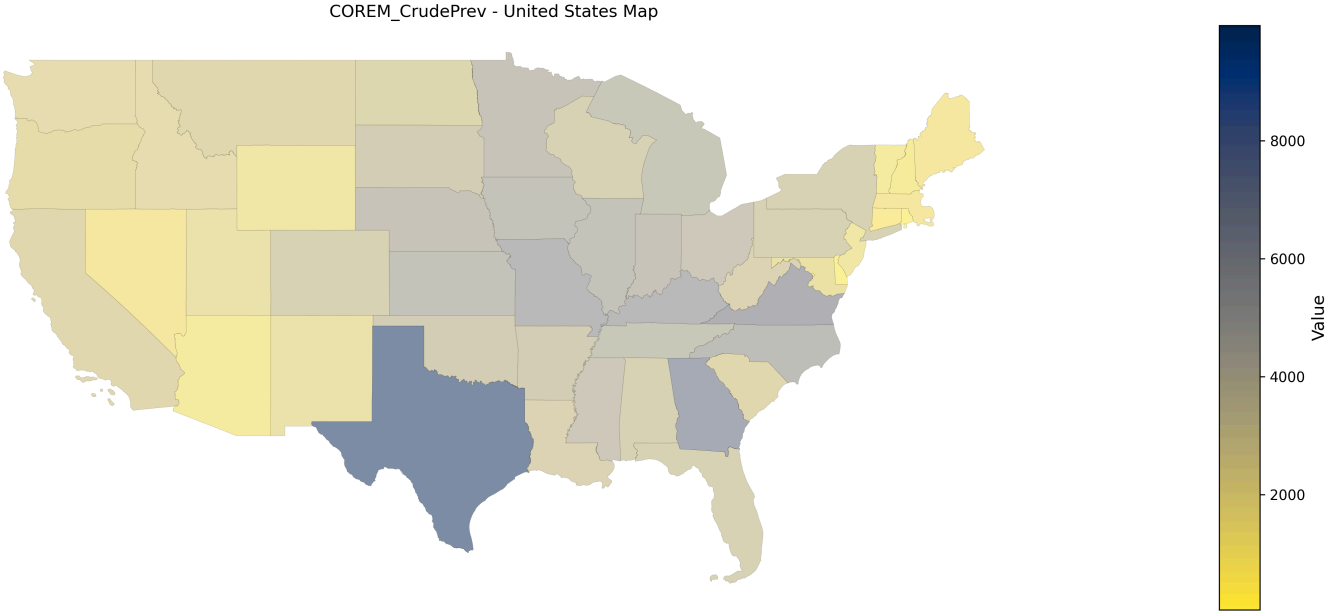
COLON_SCREEN_CrudePrev - United States Map



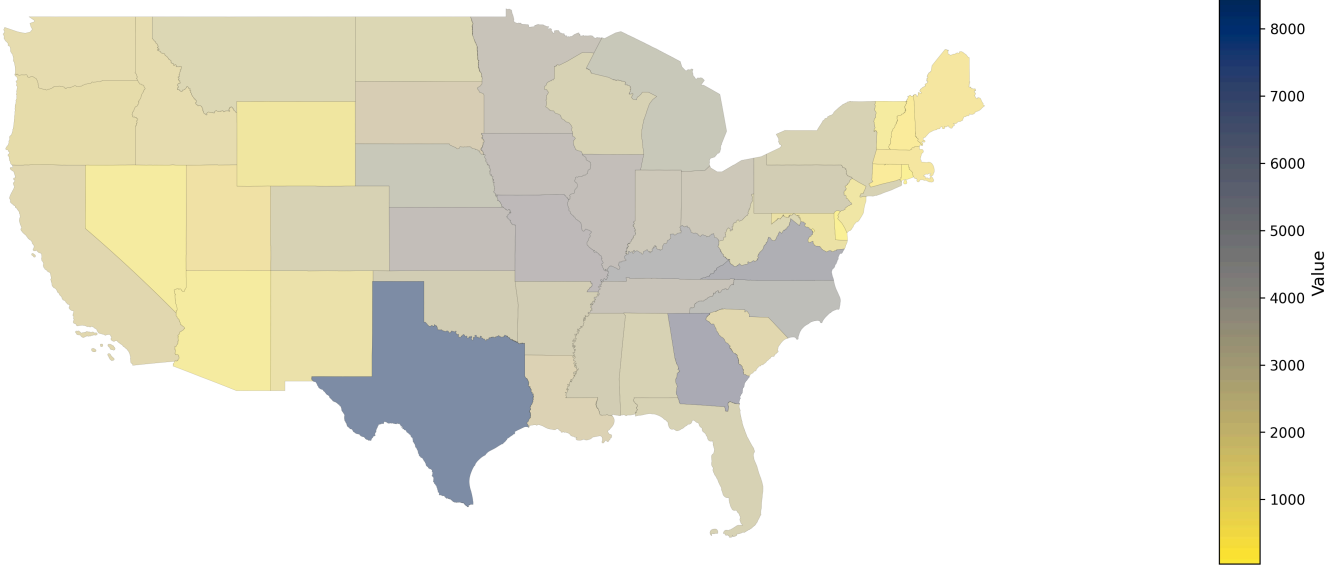
COPD_CrudePrev - United States Map



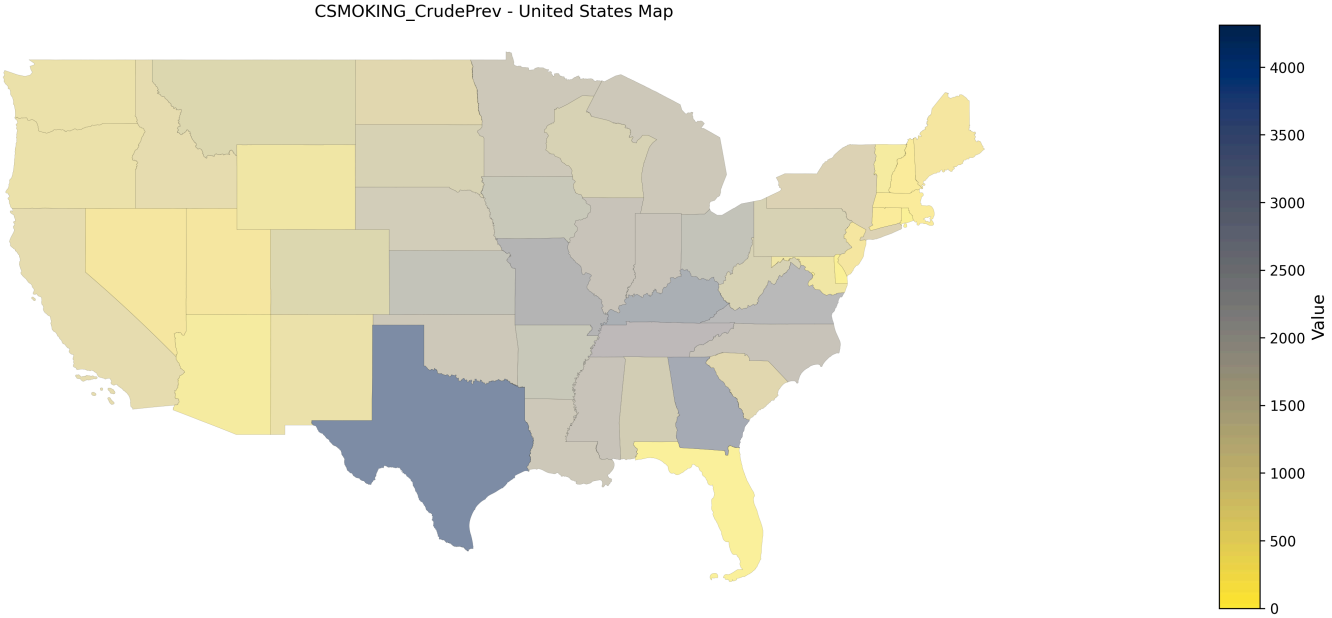
COREM_CrudePrev - United States Map



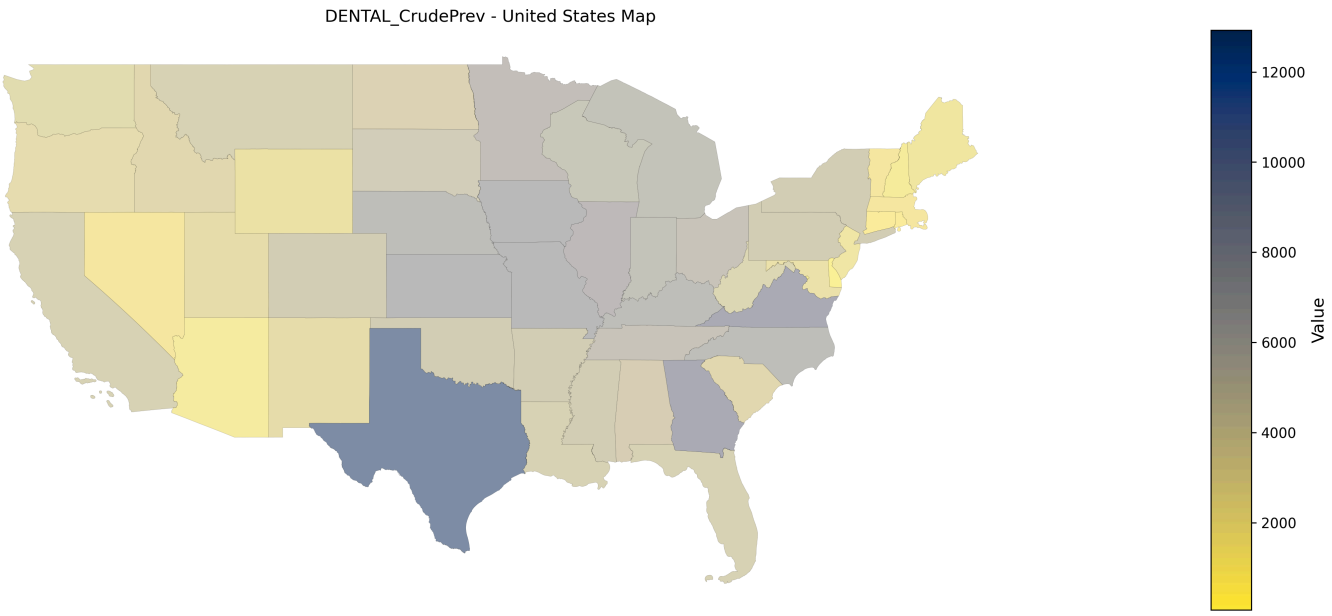
COREW_CrudePrev - United States Map



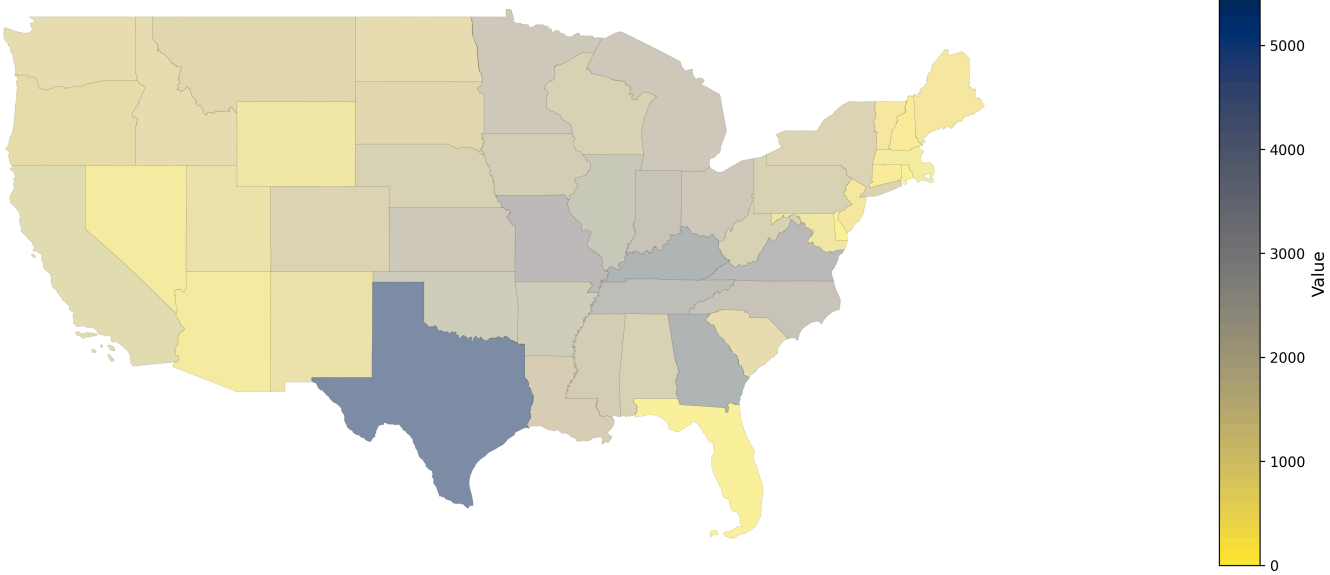
CSMOKING_CrudePrev - United States Map



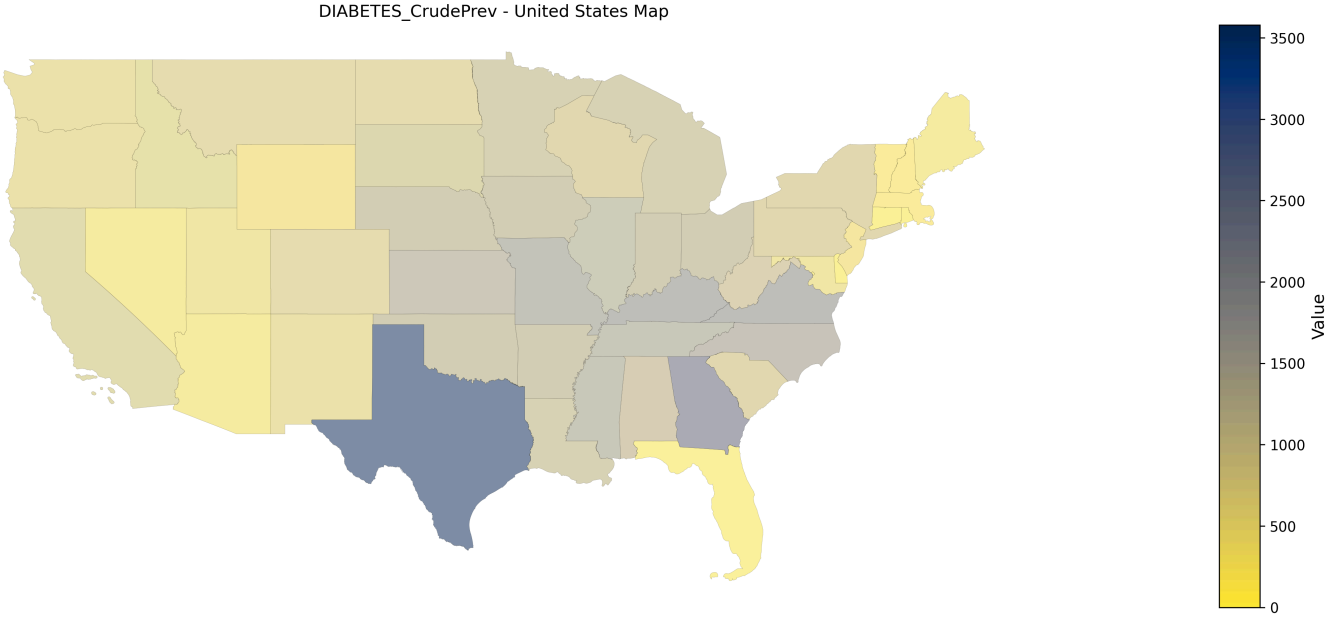
DENTAL_CrudePrev - United States Map



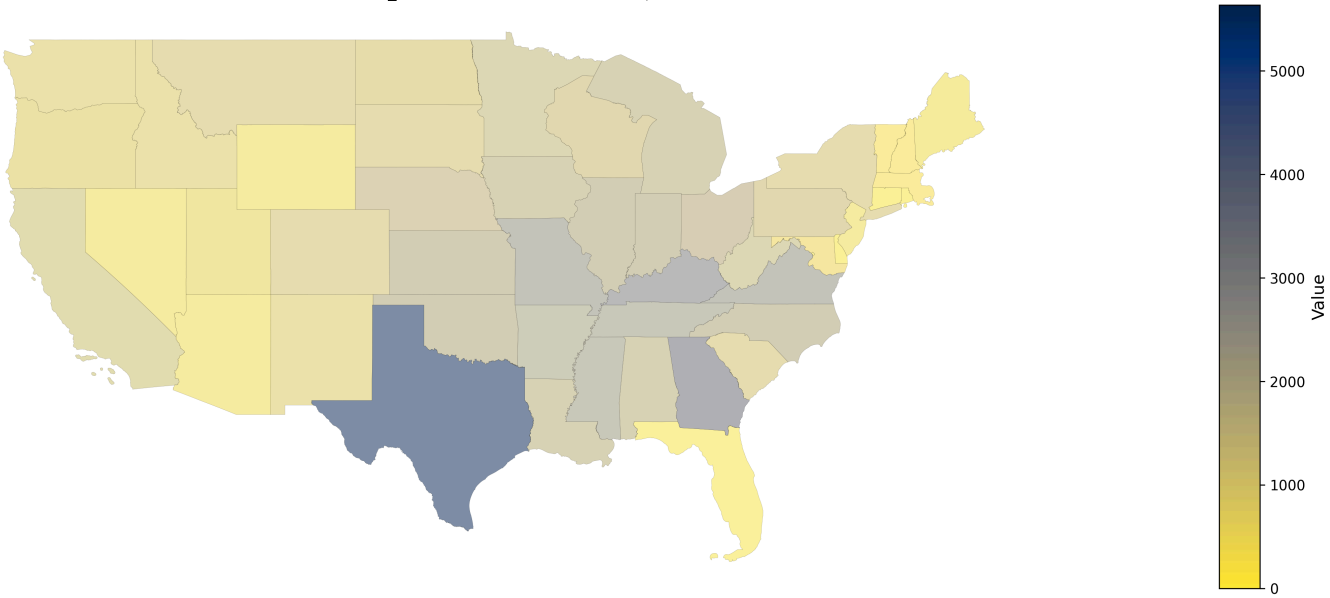
DEPRESSION_CrudePrev - United States Map



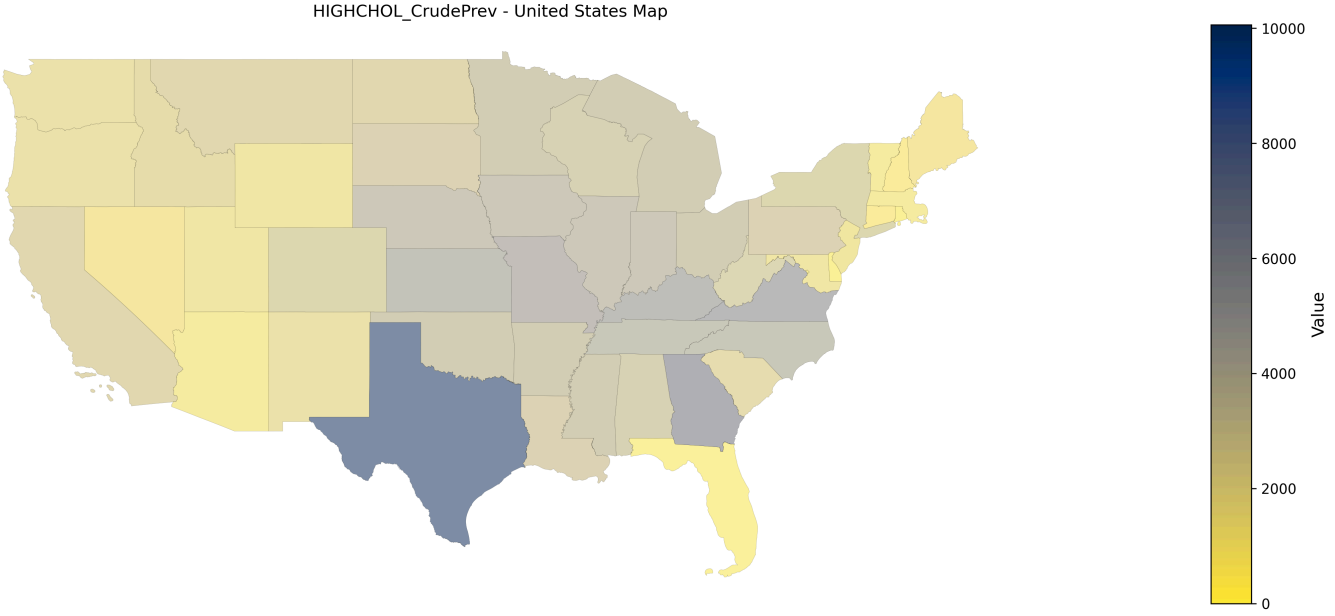
DIABETES_CrudePrev - United States Map



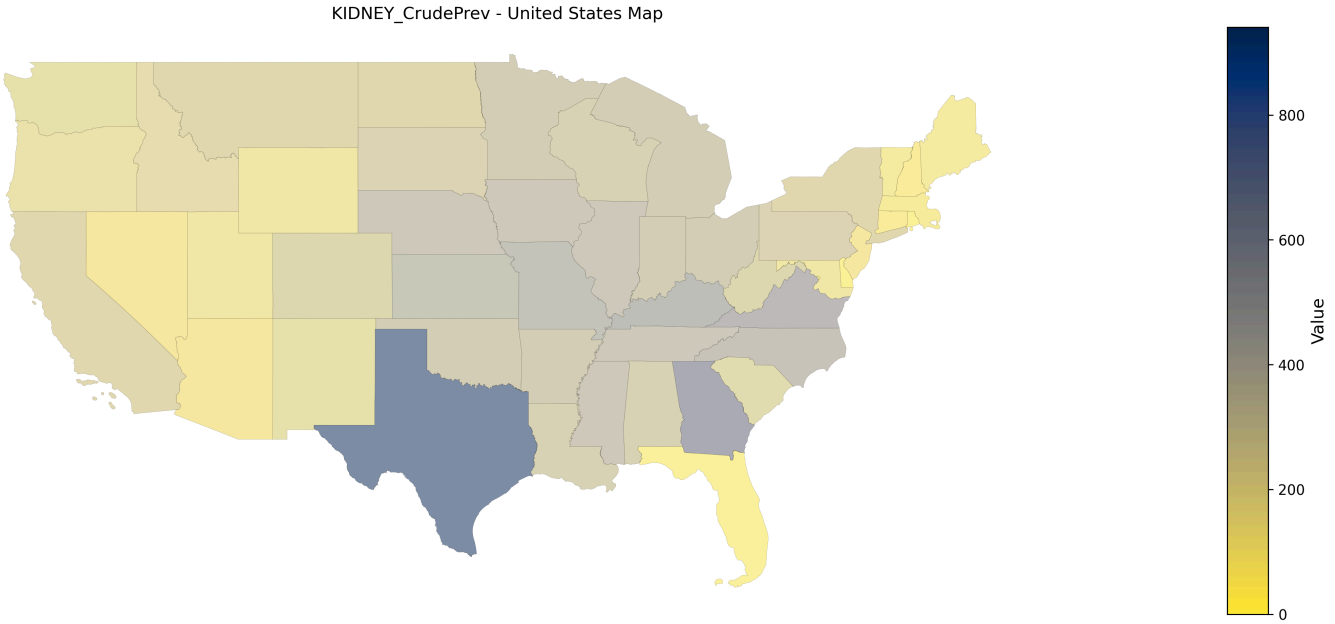
GHLTH_CrudePrev - United States Map



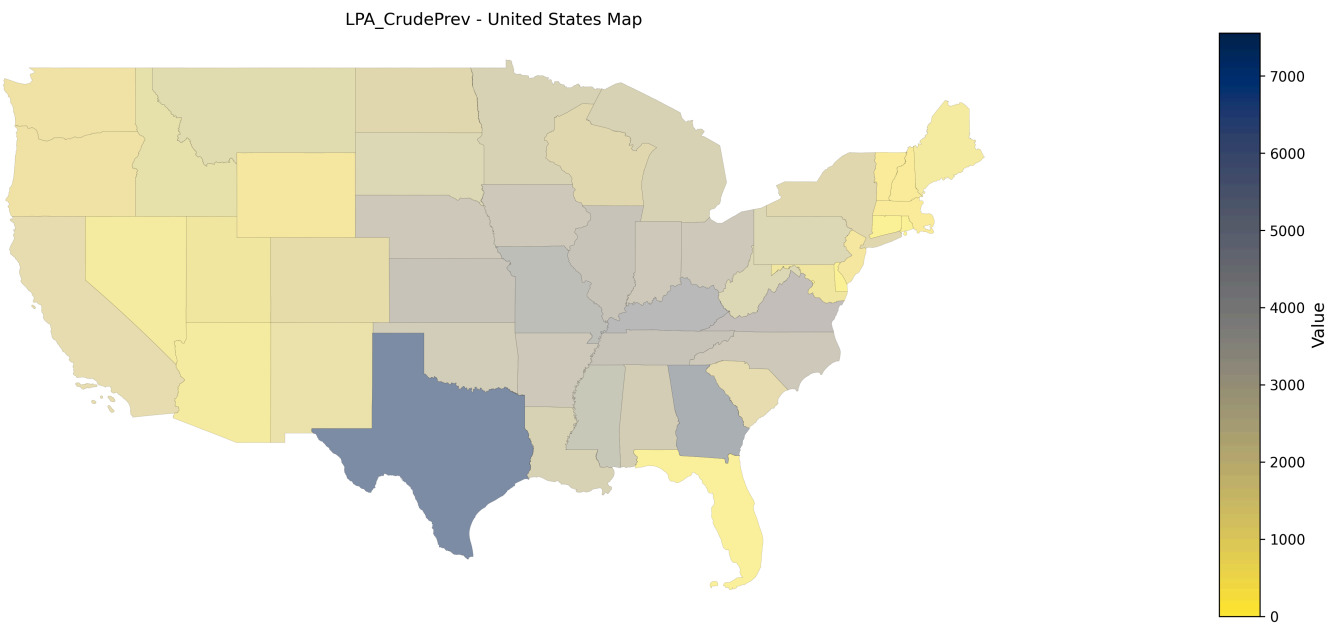
HIGHCHOL_CrudePrev - United States Map



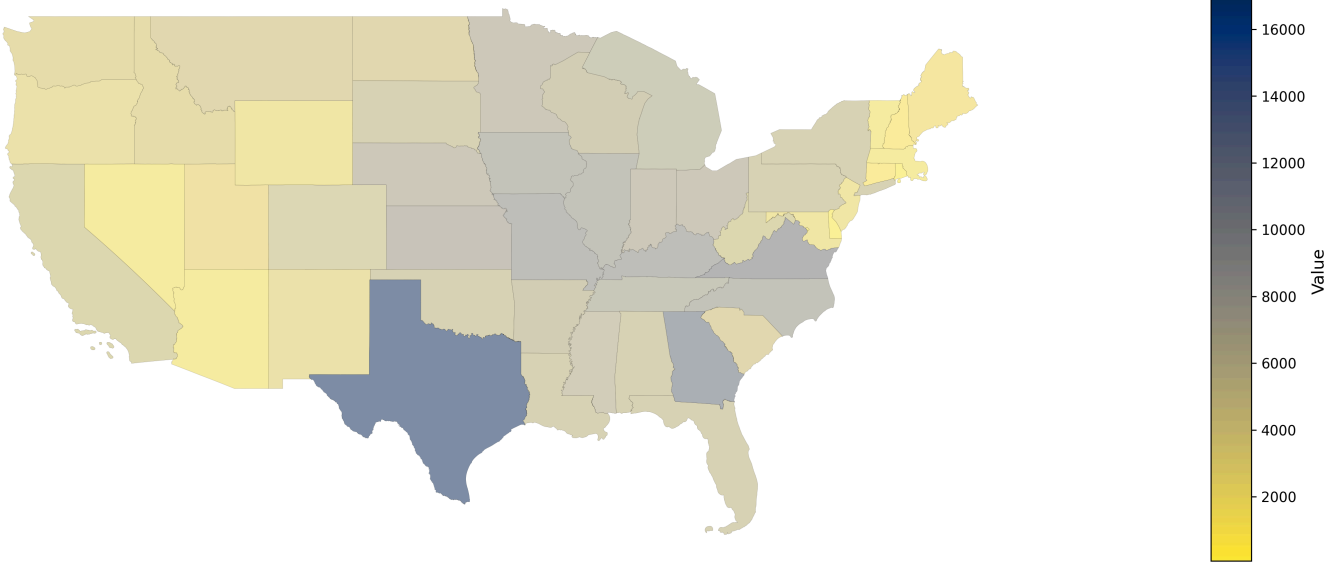
KIDNEY_CrudePrev - United States Map



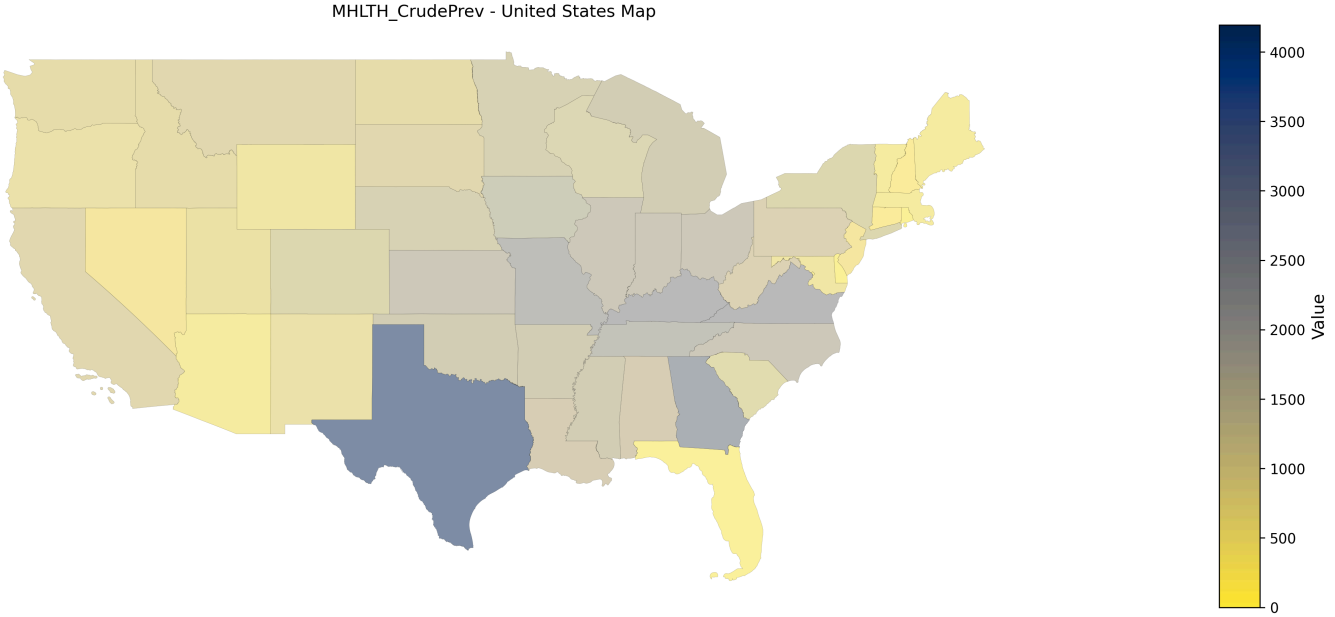
LPA_CrudePrev - United States Map



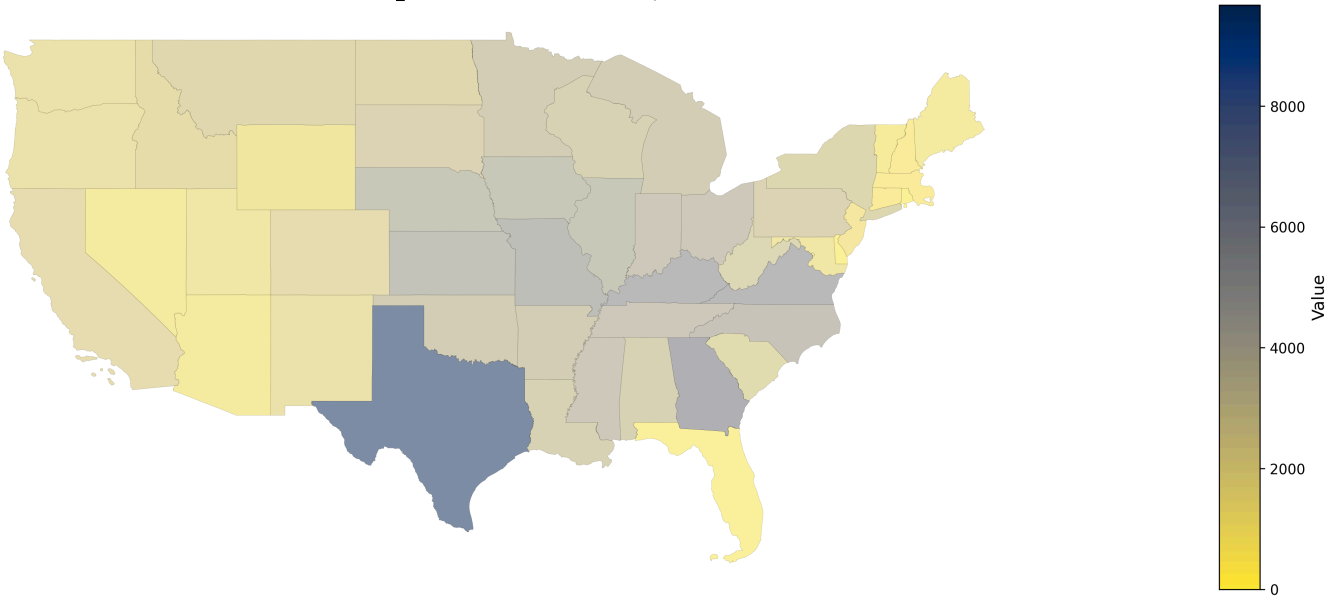
MAMMOUSE_CrudePrev - United States Map



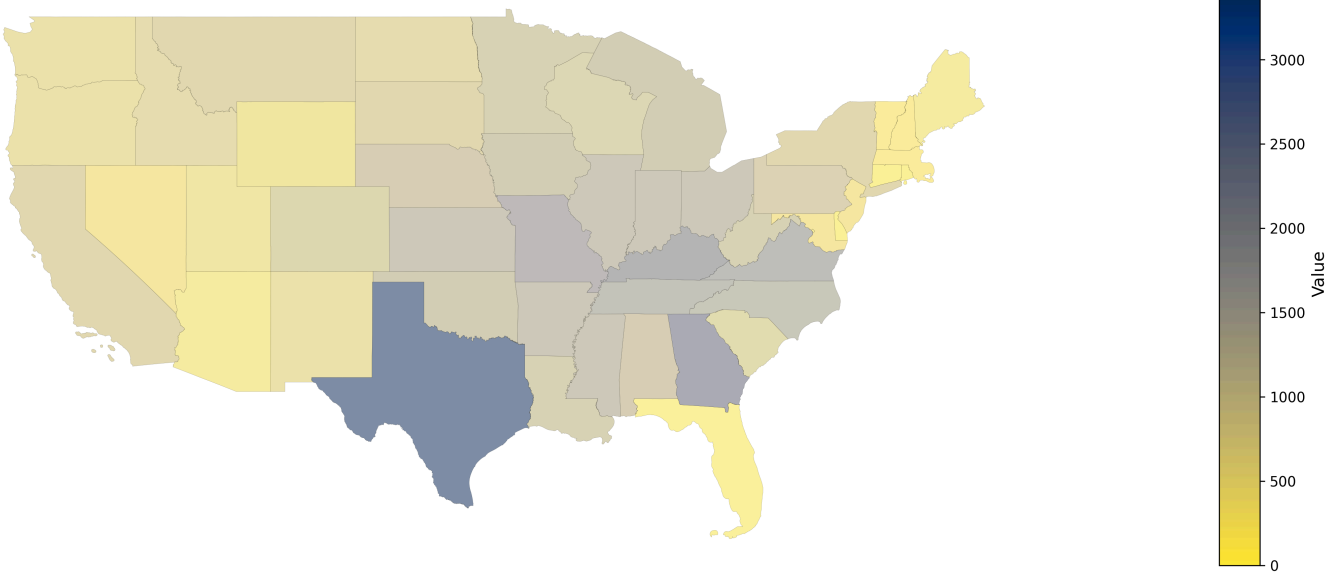
MHLTH_CrudePrev - United States Map



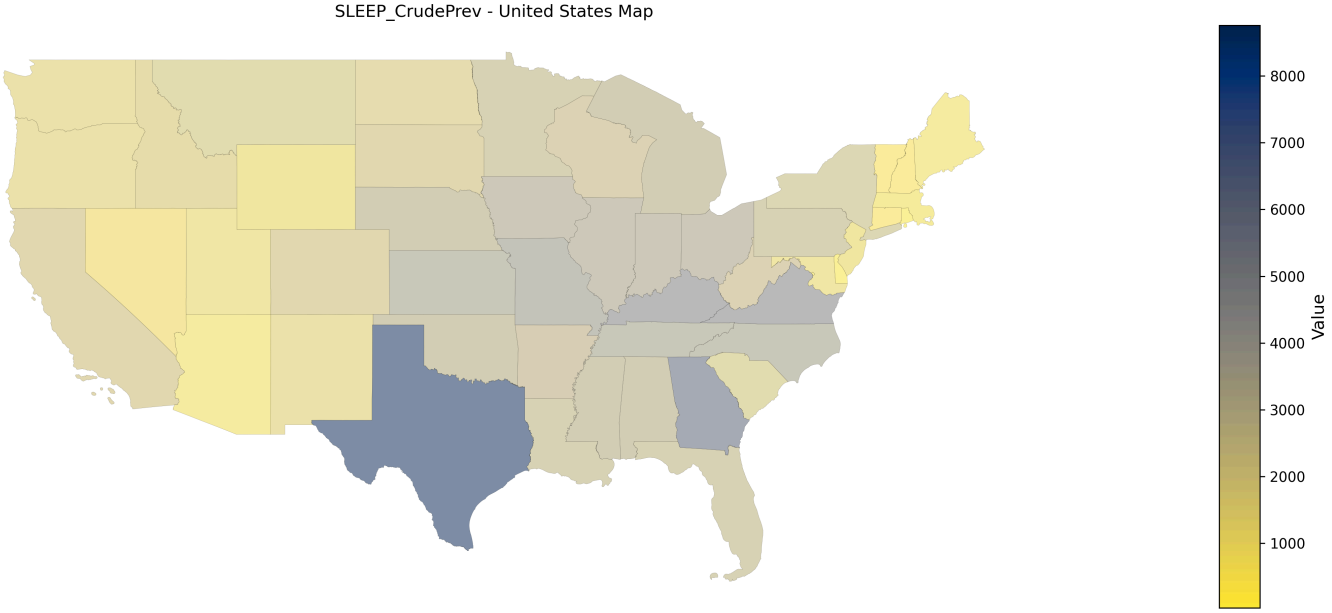
OBESITY_CrudePrev - United States Map



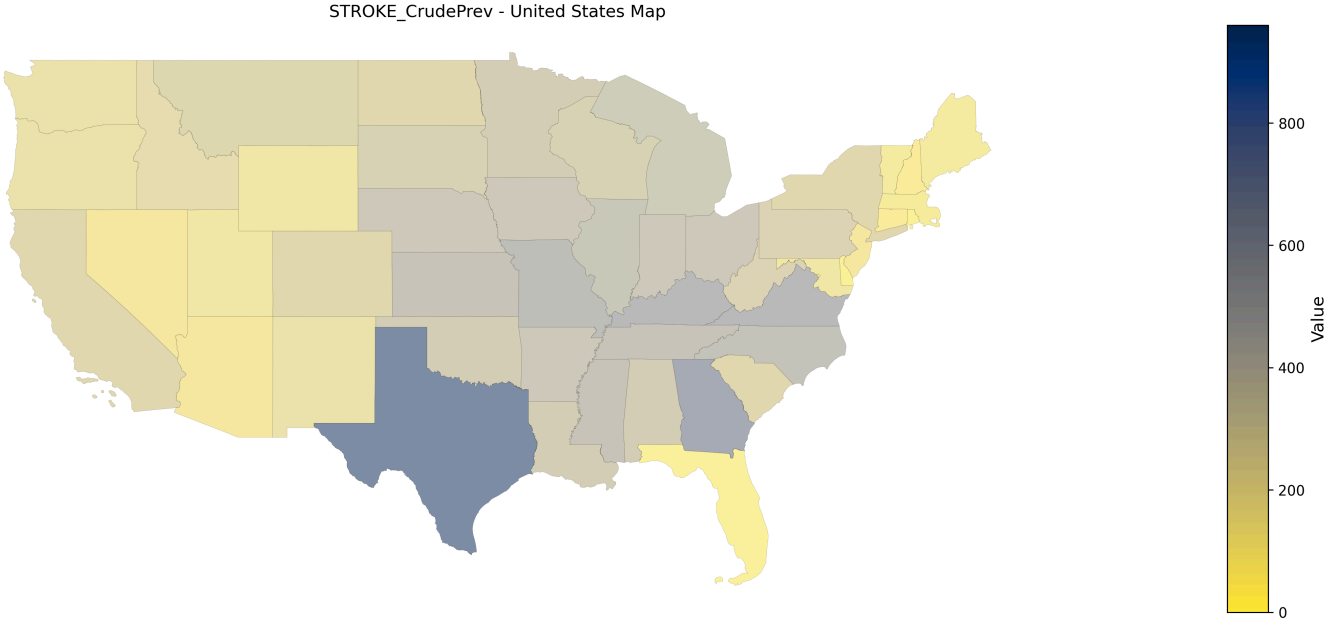
PHLTH_CrudePrev - United States Map



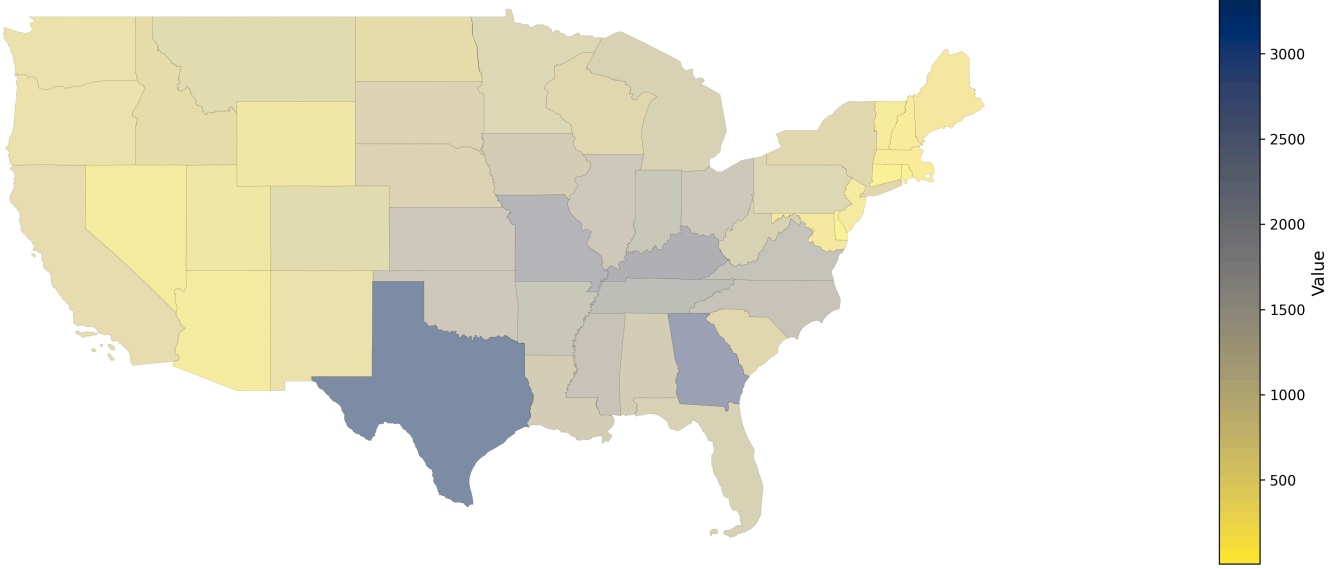
SLEEP_CrudePrev - United States Map



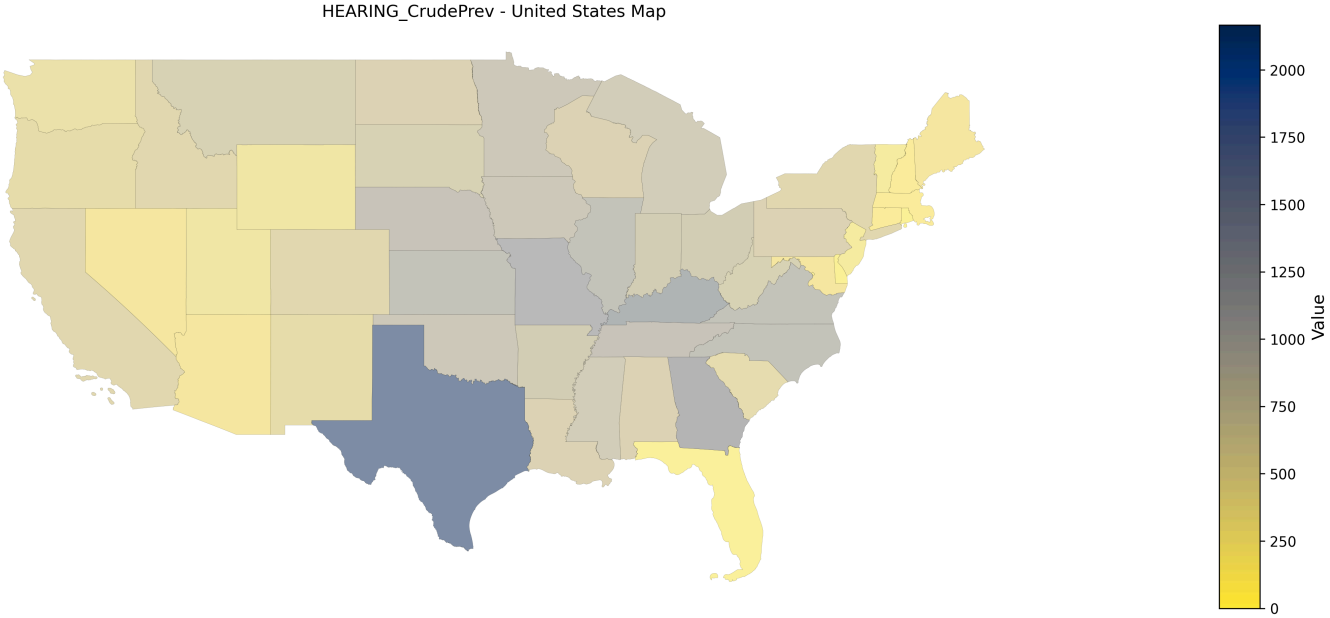
STROKE_CrudePrev - United States Map



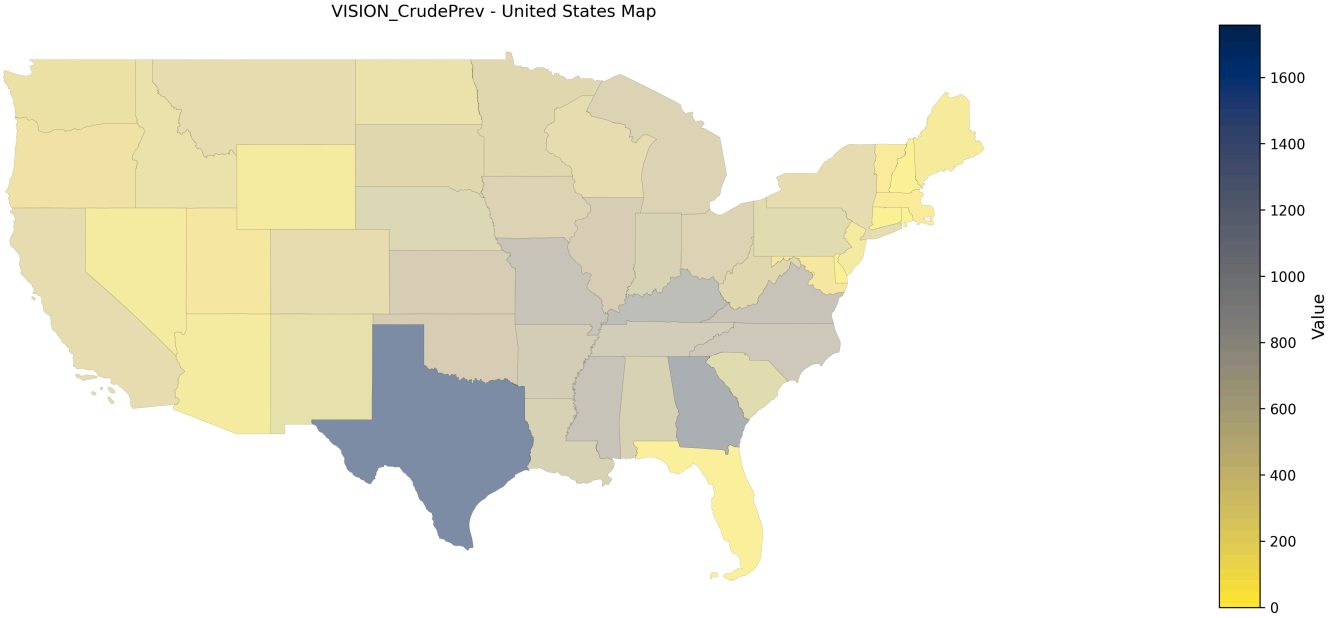
TEETHLOST_CrudePrev - United States Map



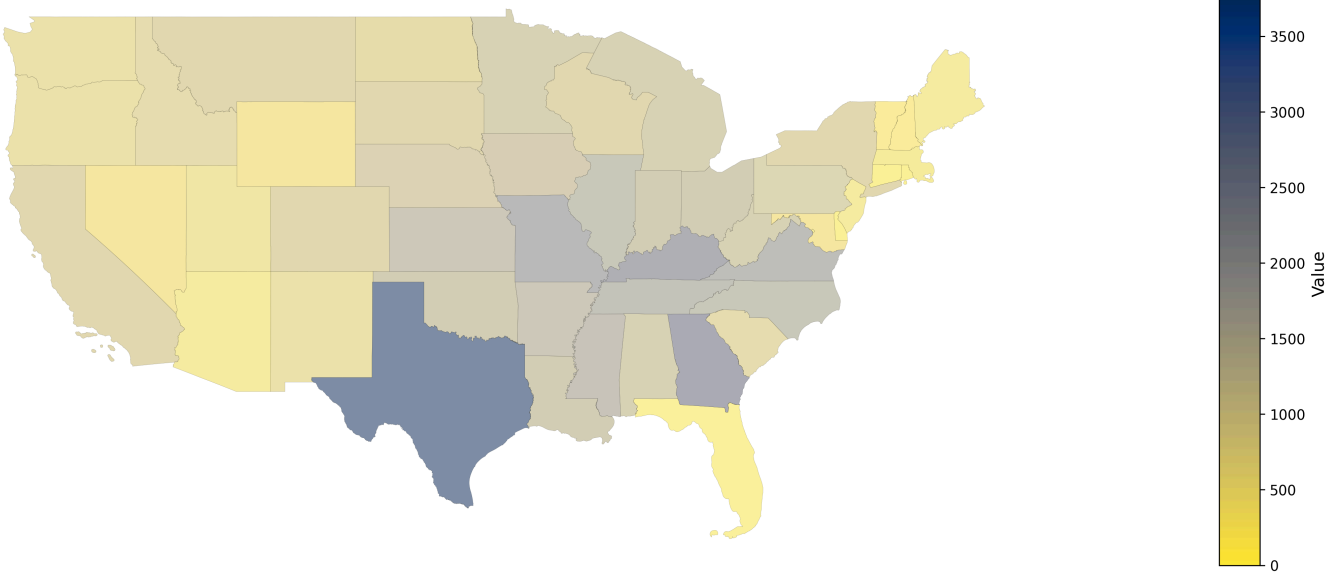
HEARING_CrudePrev - United States Map



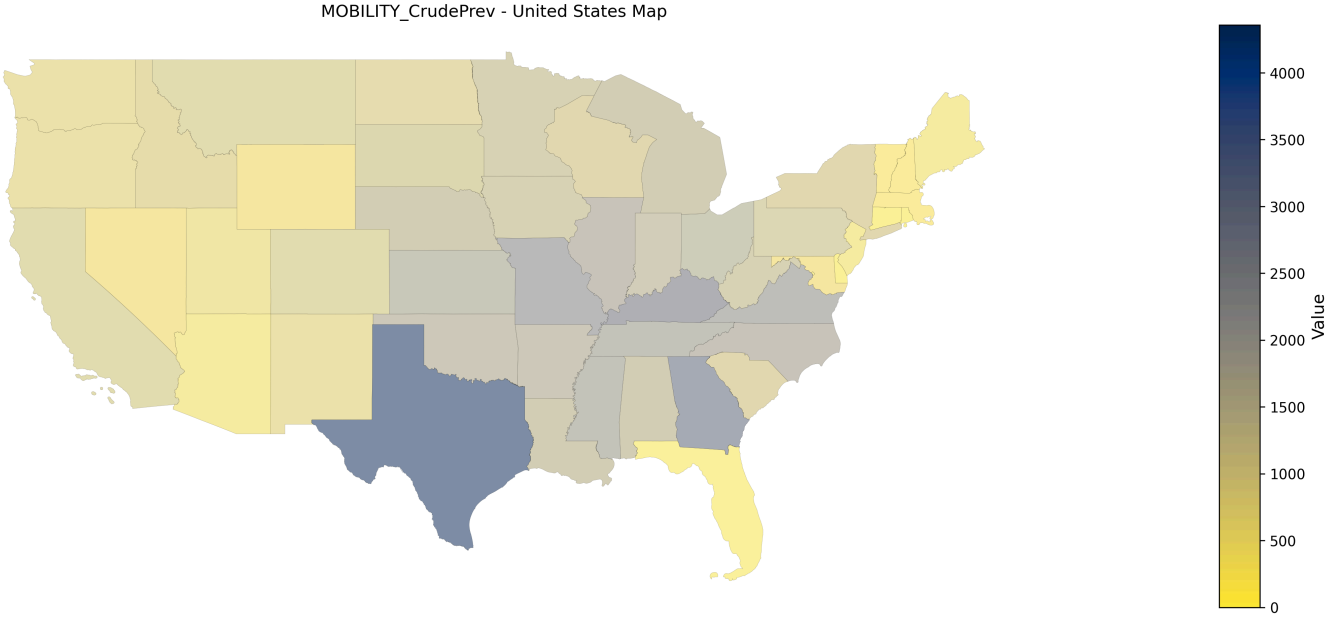
VISION_CrudePrev - United States Map



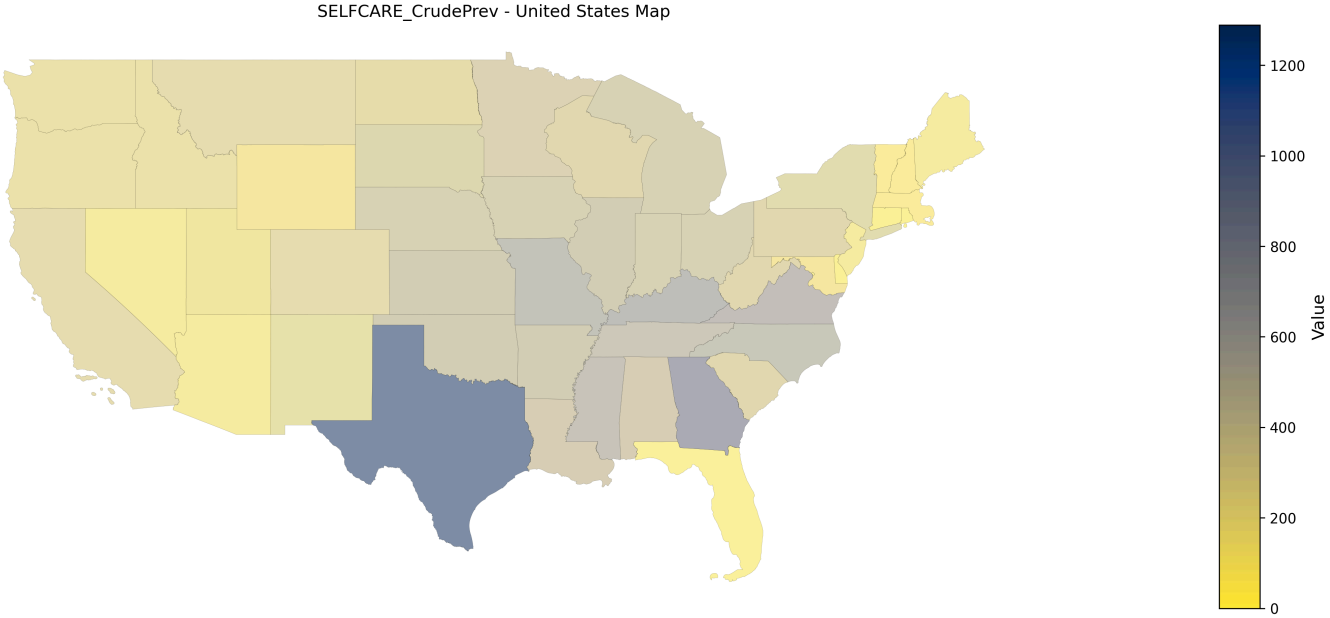
COGNITION_CrudePrev - United States Map

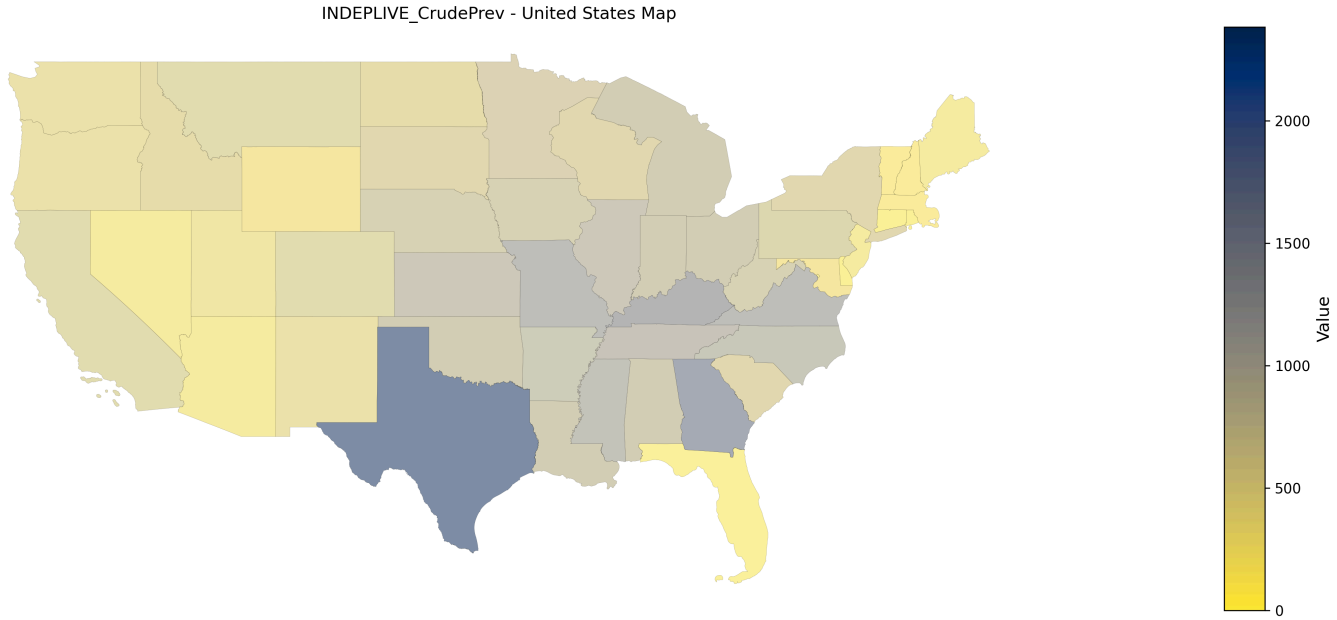


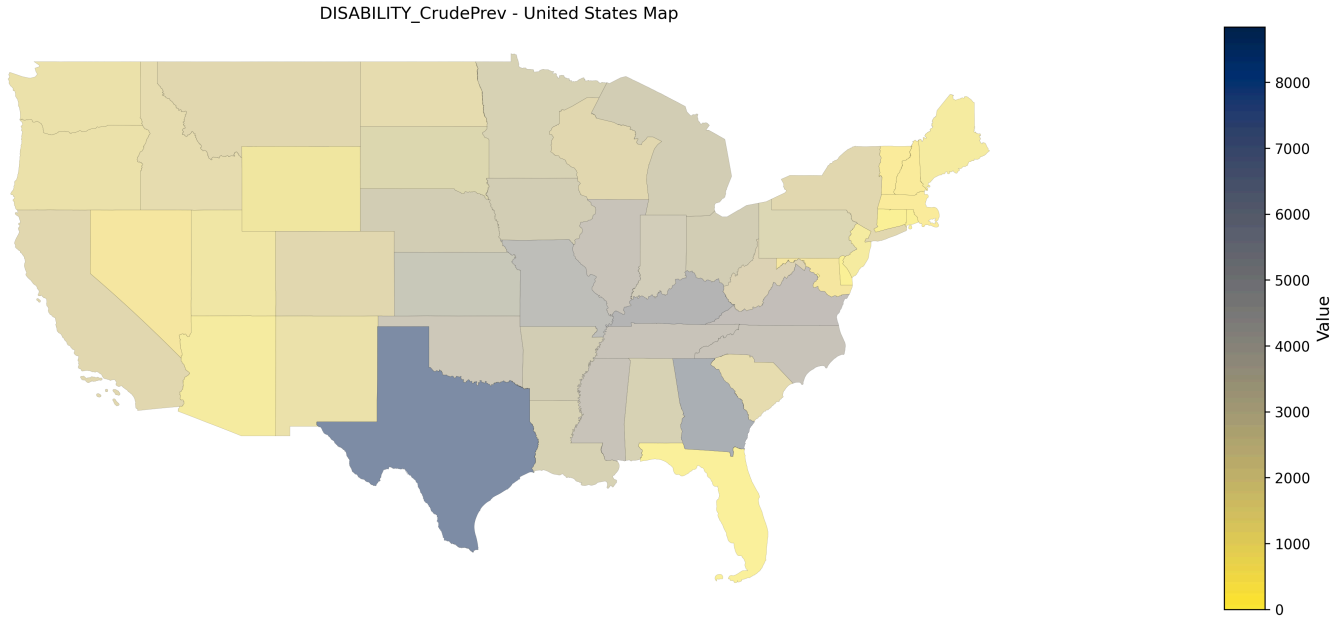
MOBILITY_CrudePrev - United States Map



SELFCARE_CrudePrev - United States Map







A Focus on Illinois State

Filter by the State of Illinois

In [ ]:
```
df_il = df[df['StateAbbr']=='IL']
```

Import Illinois State Shapefile

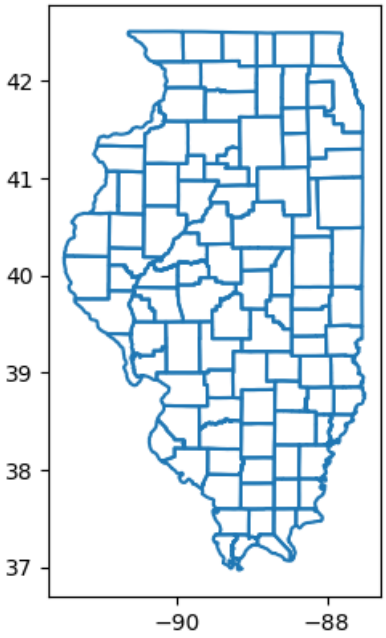
In [ ]:
```
path = "C:/Users/user/Desktop/GIS/Final Project/IL_BNDY_County/IL_BNDY_County_Py.shp"
county = gpd.read_file(path)
county = county.to_crs("EPSG:4326")
county
```

Out[ ]:

	COUNTY_NAM	CO_FIPS	geometry
0	MCHENRY	111	POLYGON ((-88.70742 42.49351, -88.68809 42.493...
1	BOONE	7	POLYGON ((-88.70742 42.49351, -88.70740 42.493...
2	OGLE	141	POLYGON ((-89.68808 42.19949, -89.66846 42.200...
3	WILL	197	POLYGON ((-88.26146 41.72439, -88.26146 41.724...
4	LASALLE	99	POLYGON ((-88.93885 41.62836, -88.93871 41.628...
...	...	...	...
97	JEFFERSON	81	POLYGON ((-89.14445 38.47386, -89.12638 38.473...
98	LAWRENCE	101	POLYGON ((-87.90805 38.85012, -87.88944 38.849...
99	MARION	121	POLYGON ((-89.13843 38.73632, -89.13827 38.750...
100	UNION	181	POLYGON ((-89.04143 37.59649, -89.04143 37.596...
101	POPE	151	POLYGON ((-88.70860 37.59925, -88.69093 37.599...

102 rows × 3 columns

In [ ]:
```
county.boundary.plot()
plt.show()
```



Preprocess and Merge

In [ ]:
```
df_il['CountyName'] = df_il['CountyName'].str.upper()
df_il.head(2)
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_17260\1893931094.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-v
iew-versus-a-copy
  df_il['CountyName'] = df_il['CountyName'].str.upper()
```

Out[ ]:

	StateAbbr	StateDesc	CountyName	CountyFIPS	TotalPopulation	ACCESS2_CrudePrev	ACCESS2_Crude95CI	ACCESS2_AdjPrev	ACCESS2_Adj
62	IL	Illinois	BOND	17005	16596	9.0	(6.9, 11.7)	9.4	(7.3, 1
63	IL	Illinois	CALHOUN	17013	4369	7.0	(5.1, 9.1)	7.7	(5.7, 1

2 rows × 154 columns

In [ ]:
```
# Assuming df2 is your DataFrame
df_il.loc[df_il['CountyName'] == 'DE WITT', 'CountyName'] = 'DEWITT'
```

In [ ]:
```
# Merge aggregated DataFrame with the GeoDataFrame
merged_gdf = county.merge(df_il, left_on='COUNTY_NAM', right_on='CountyName', how='left')
merged_gdf.head(2)
```

Out[ ]:

	COUNTY_NAM	CO_FIPS	geometry	StateAbbr	StateDesc	CountyName	CountyFIPS	TotalPopulation	ACCESS2_CrudePrev	ACCESS2_Crude9
0	MCHENRY	111	POLYGON ((-88.70742 42.49351, -88.68809 42.493...	IL	Illinois	MCHENRY	17111	311122	7.4	(5.7,
1	BOONE	7	POLYGON ((-88.70742 42.49351, -88.70740 42.493...	IL	Illinois	BOONE	17007	53159	10.6	(8.1, 1

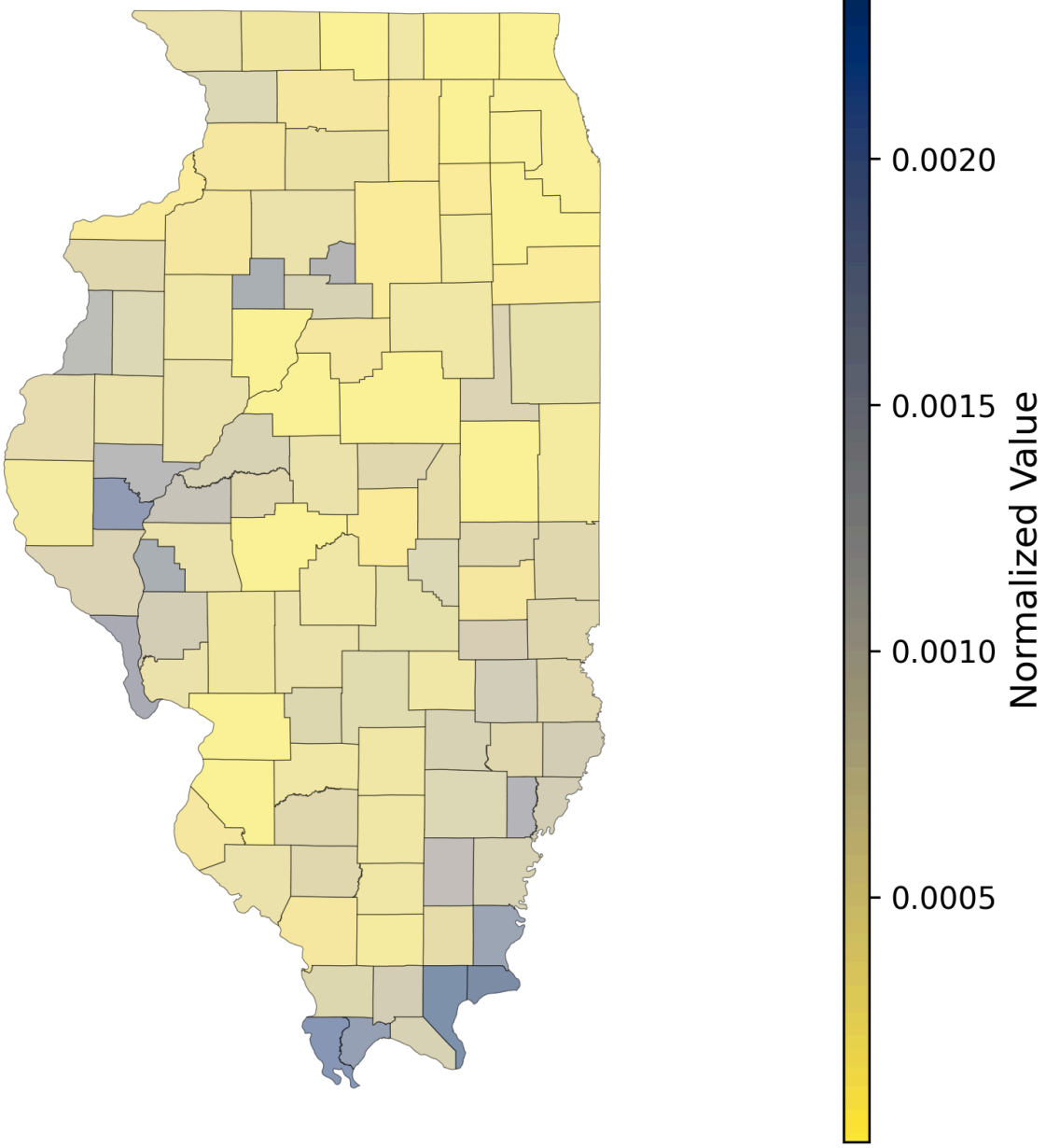
2 rows × 157 columns

Plotting

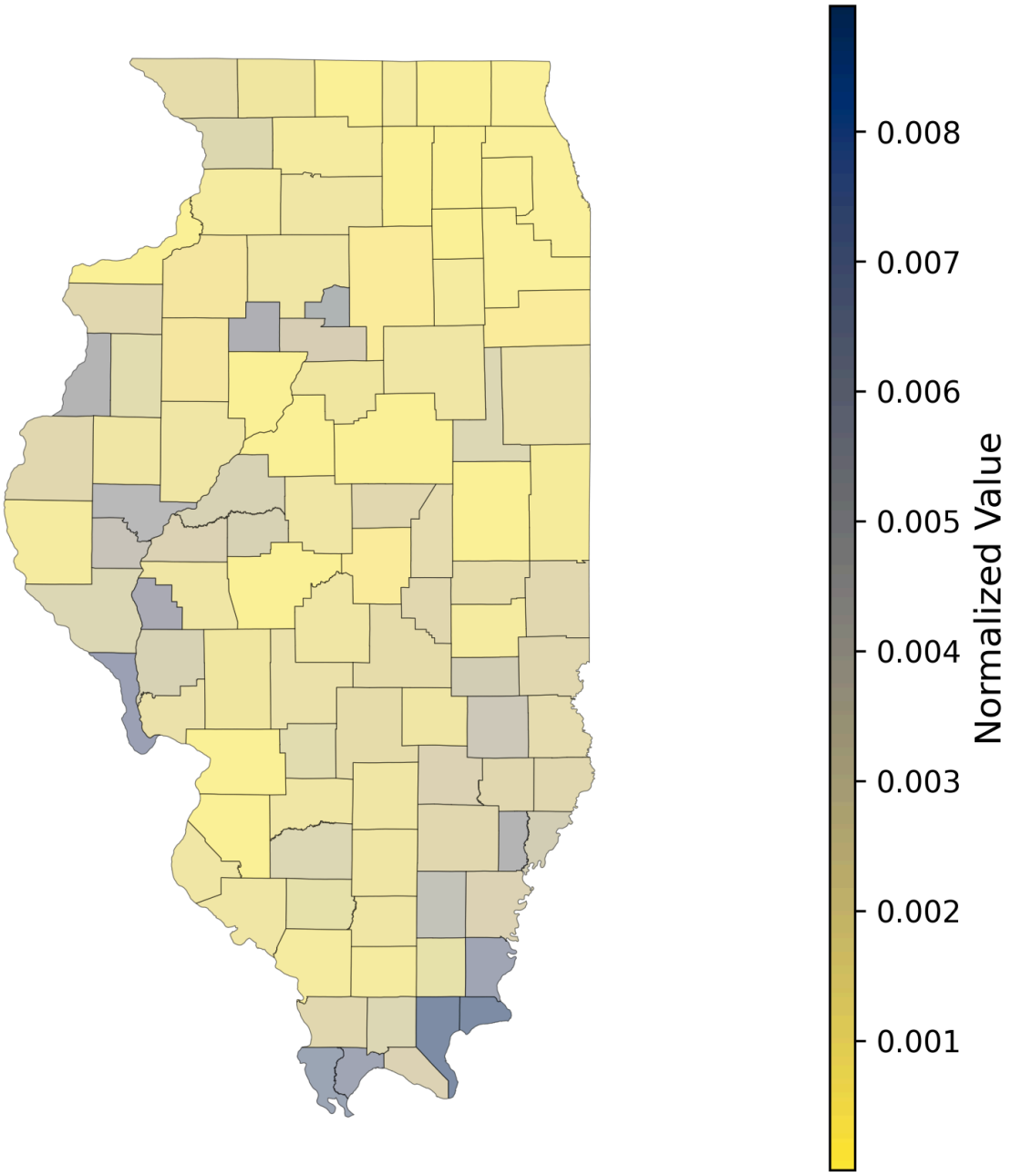
In [ ]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable

def plot_normalized_county_data(geo_df, data_columns, population_column, colormap):
    for column in data_columns:
        # Calculate normalized data by dividing by population
        normalized_column = f'Normalized_{column}'
        geo_df[normalized_column] = geo_df[column] / geo_df[population_column]

        # Plotting
        f, ax = plt.subplots(1, 1, figsize=(10, 6), sharex=True, sharey=True, dpi=300)
        f.tight_layout()
        plt.title(f'Normalized {column} - Illinois Map by County')
        ax.set_axis_off()
        divider = make_axes_locatable(ax)
        cax = divider.append_axes("right", size="3%", pad=0.5, alpha=0.5)
        geo_df.plot(normalized_column, ax=ax, alpha=0.5, cmap=colormap, edgecolor='k', legend=True, cax=cax, linewidth=0.
        plt.ylabel('Normalized Value', fontsize=12)
        plt.show()
```

In [ ]:
```
# Get columns ending with '_CrudePrev'
columns_to_plot = [col for col in df.columns if col.endswith('_CrudePrev')]

plot_normalized_county_data(merged_gdf, columns_to_plot, 'TotalPopulation', 'cividis_r')
```

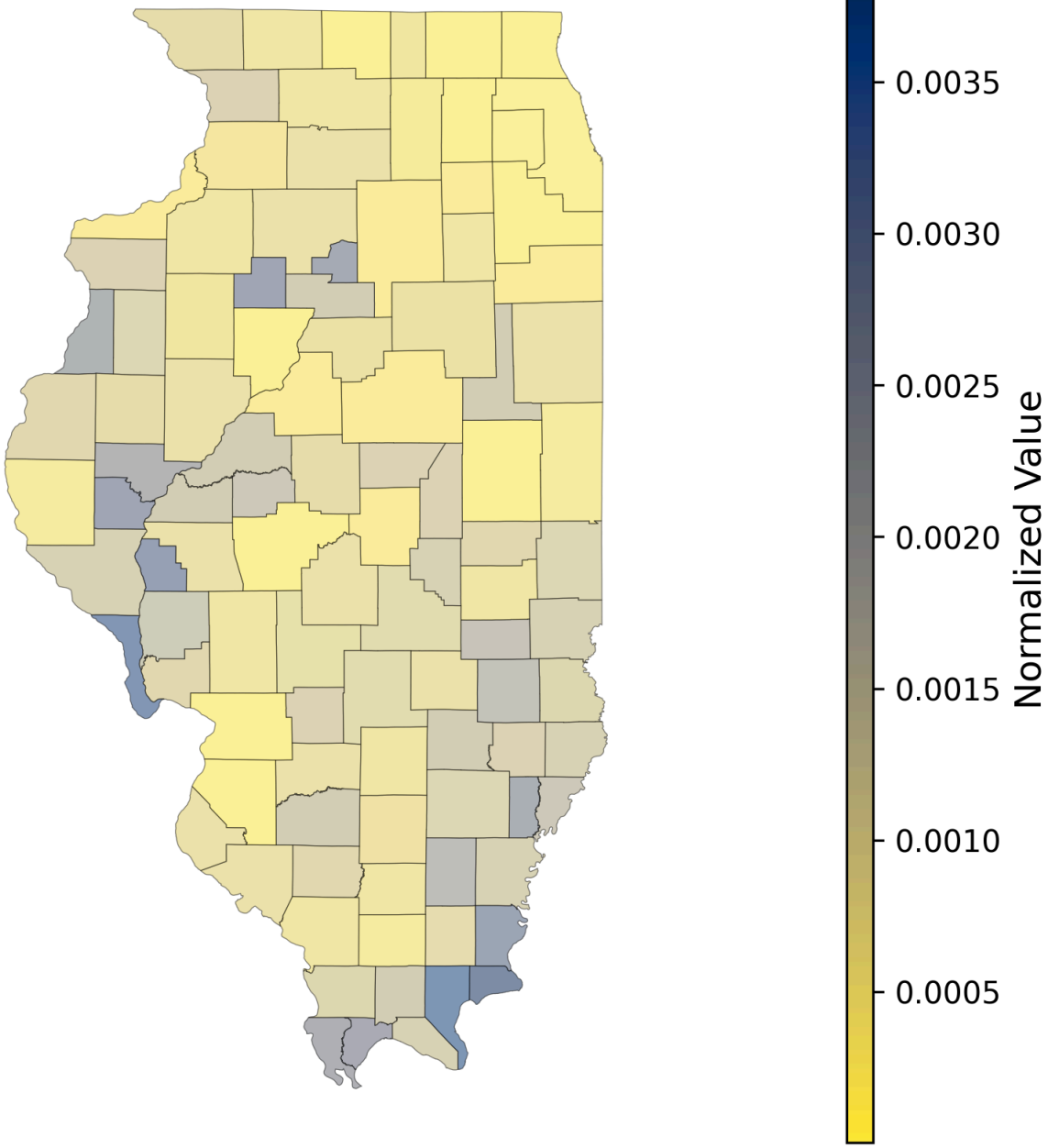
Normalized ACCESS2_CrudePrev - Illinois Map by County



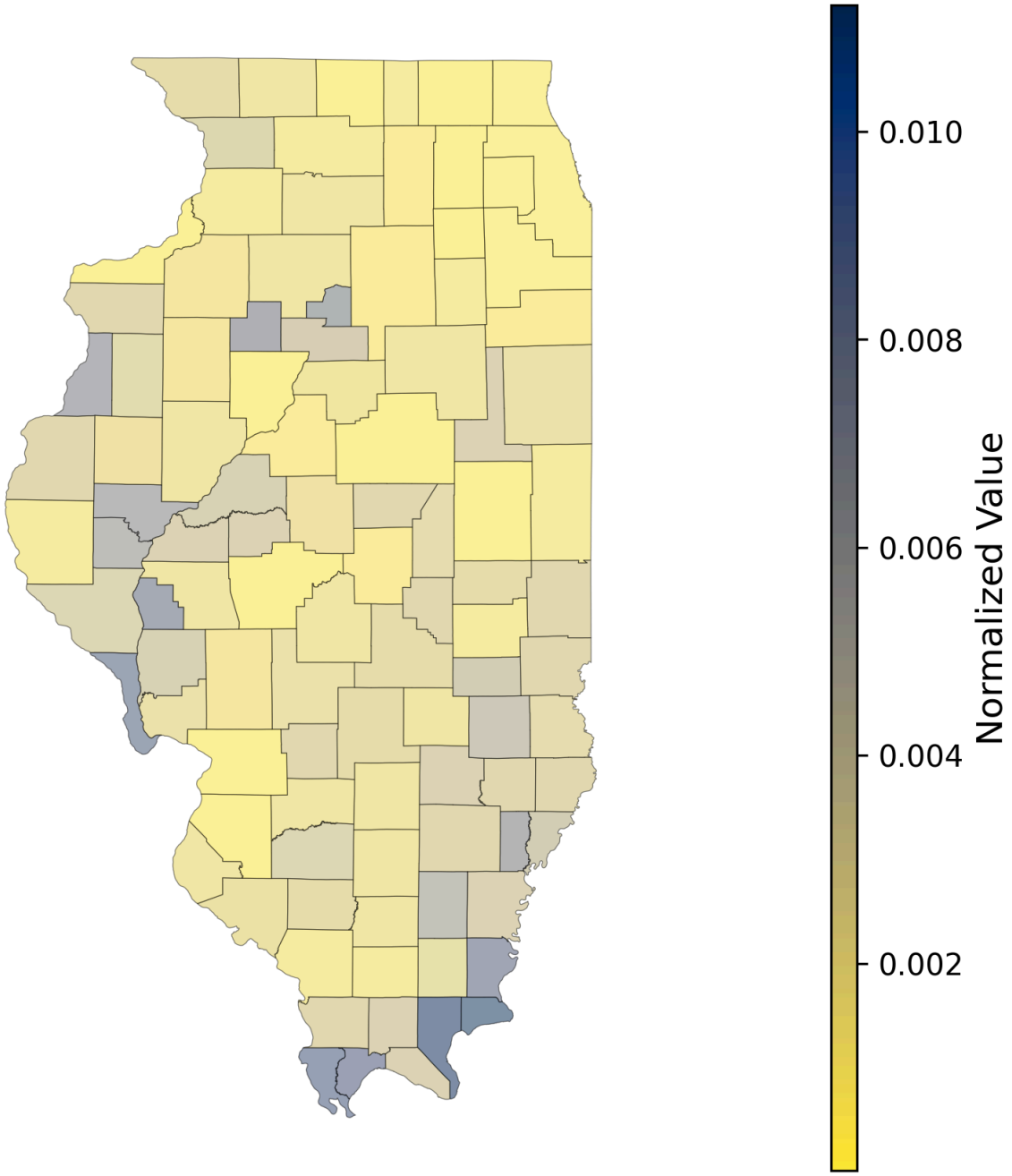
Normalized ARTHRITIS_CrudePrev - Illinois Map by County



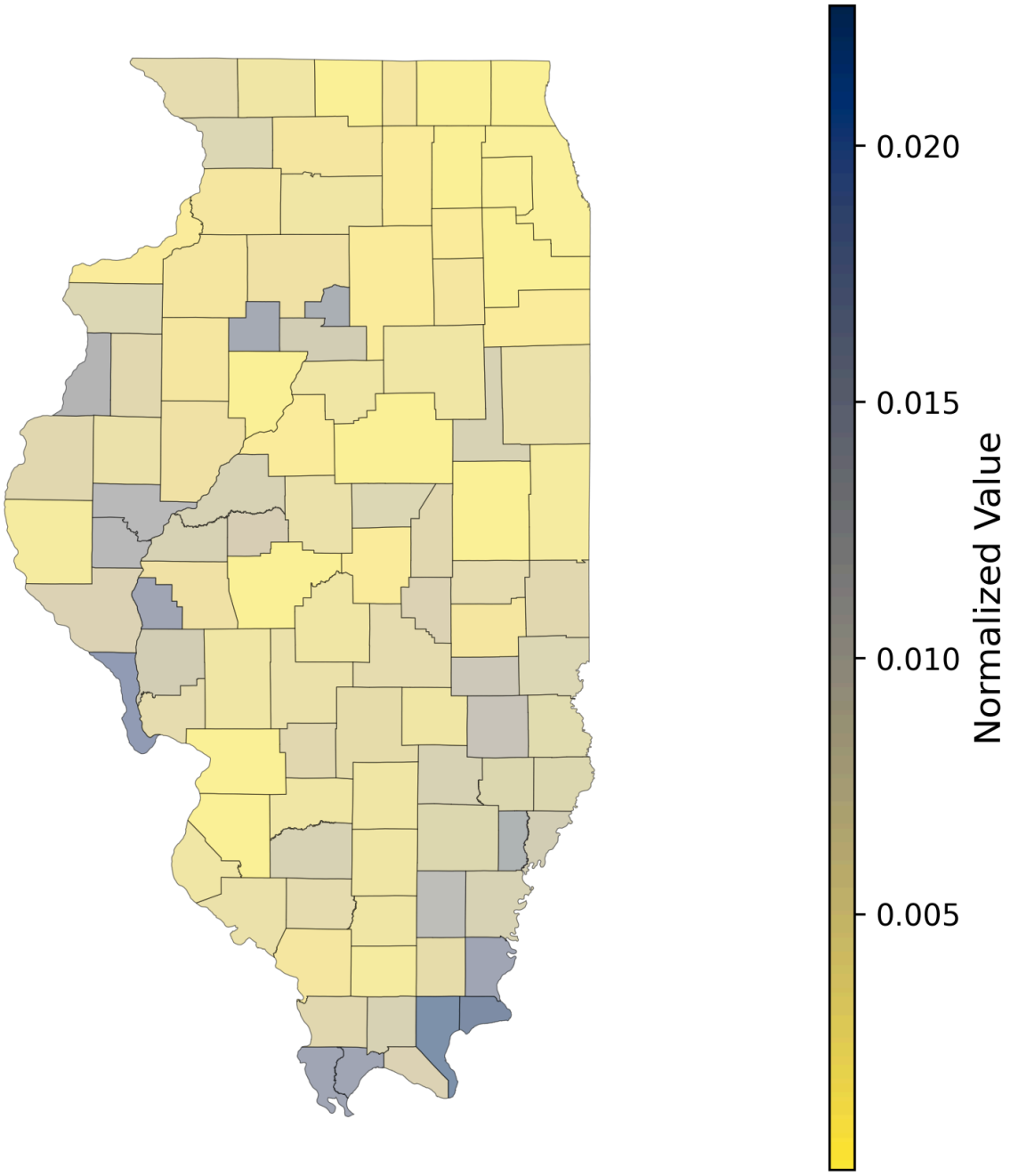
Normalized BINGE_CrudePrev - Illinois Map by County



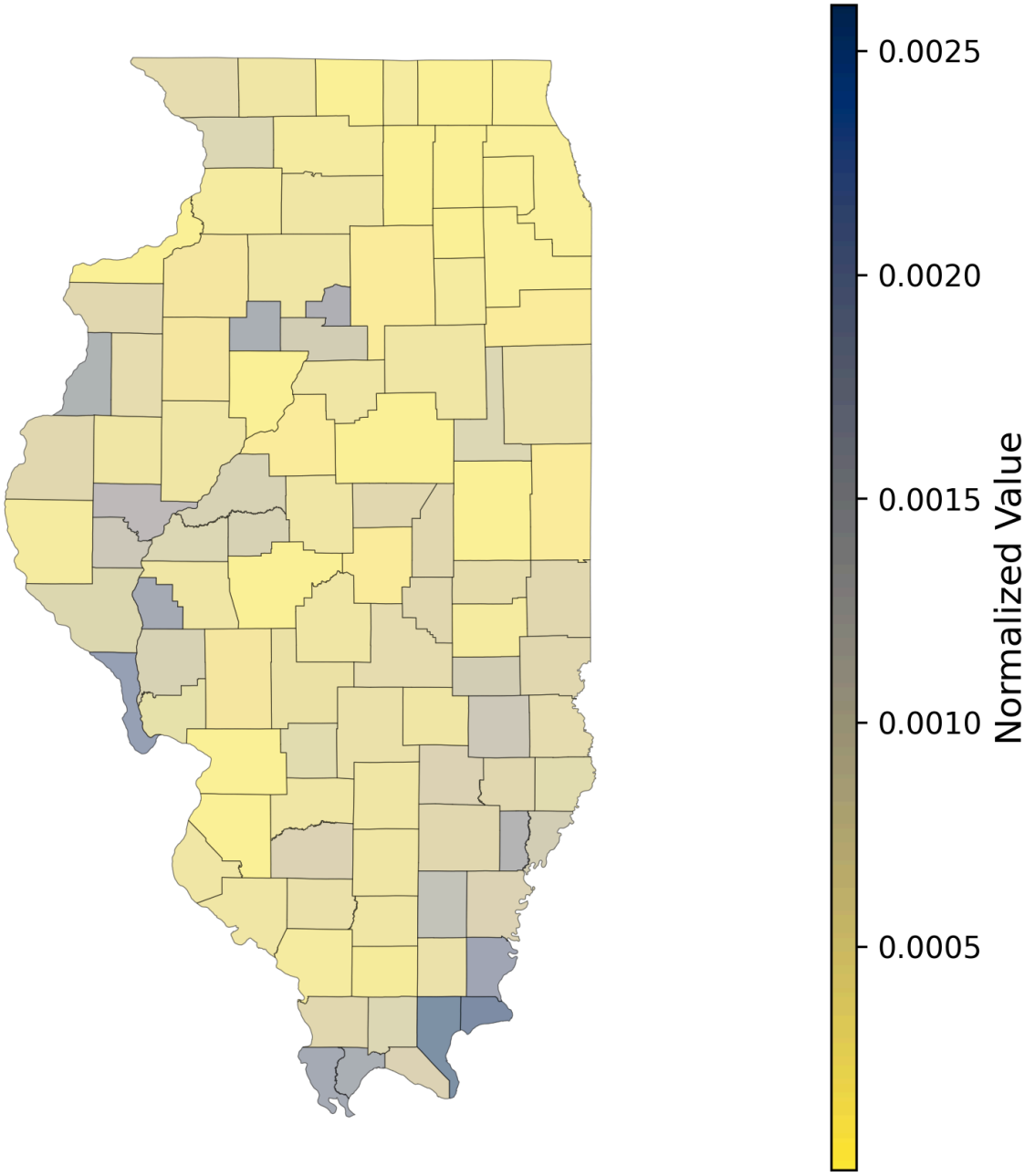
Normalized BPHIGH_CrudePrev - Illinois Map by County



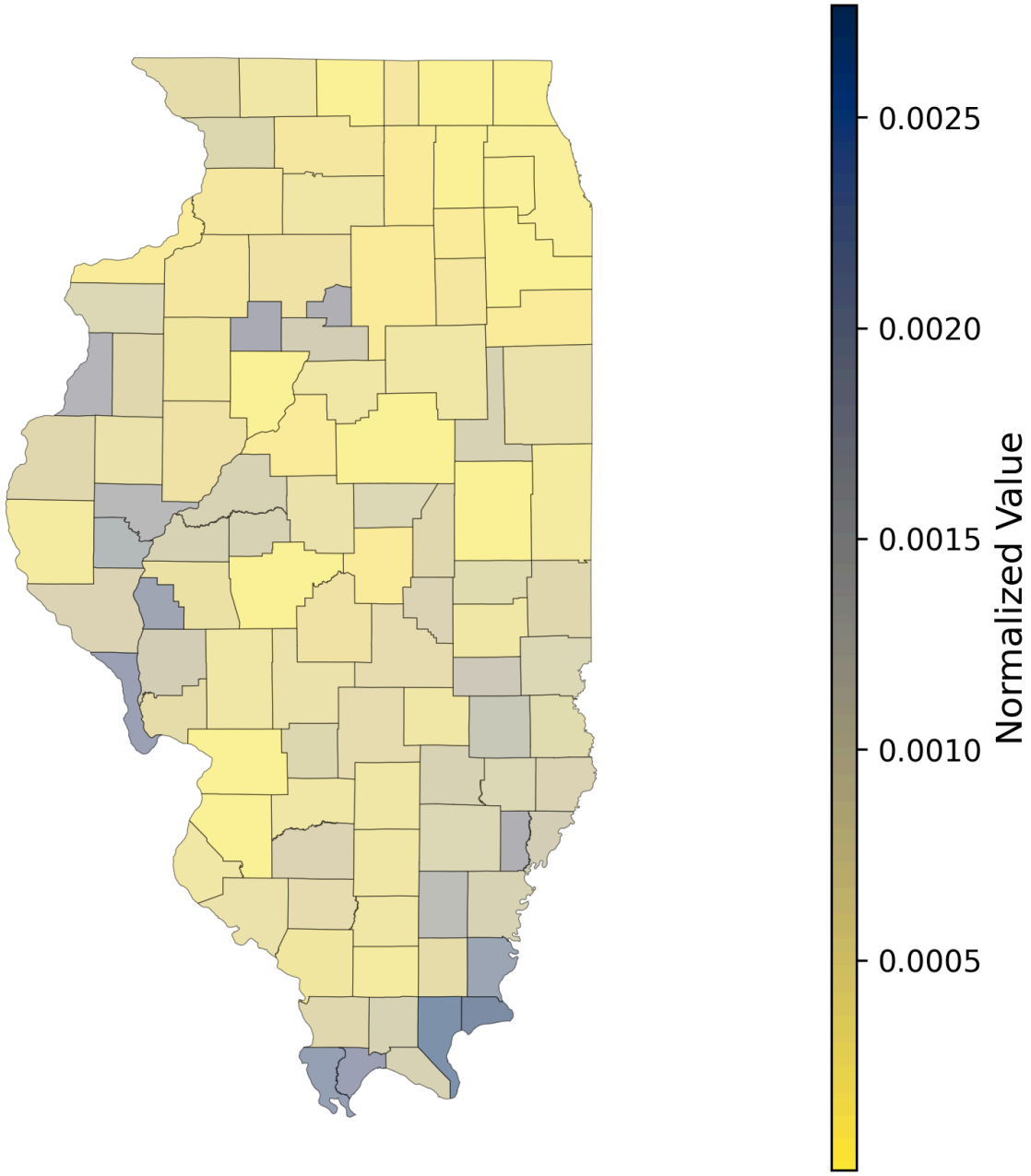
Normalized BPMED_CrudePrev - Illinois Map by County



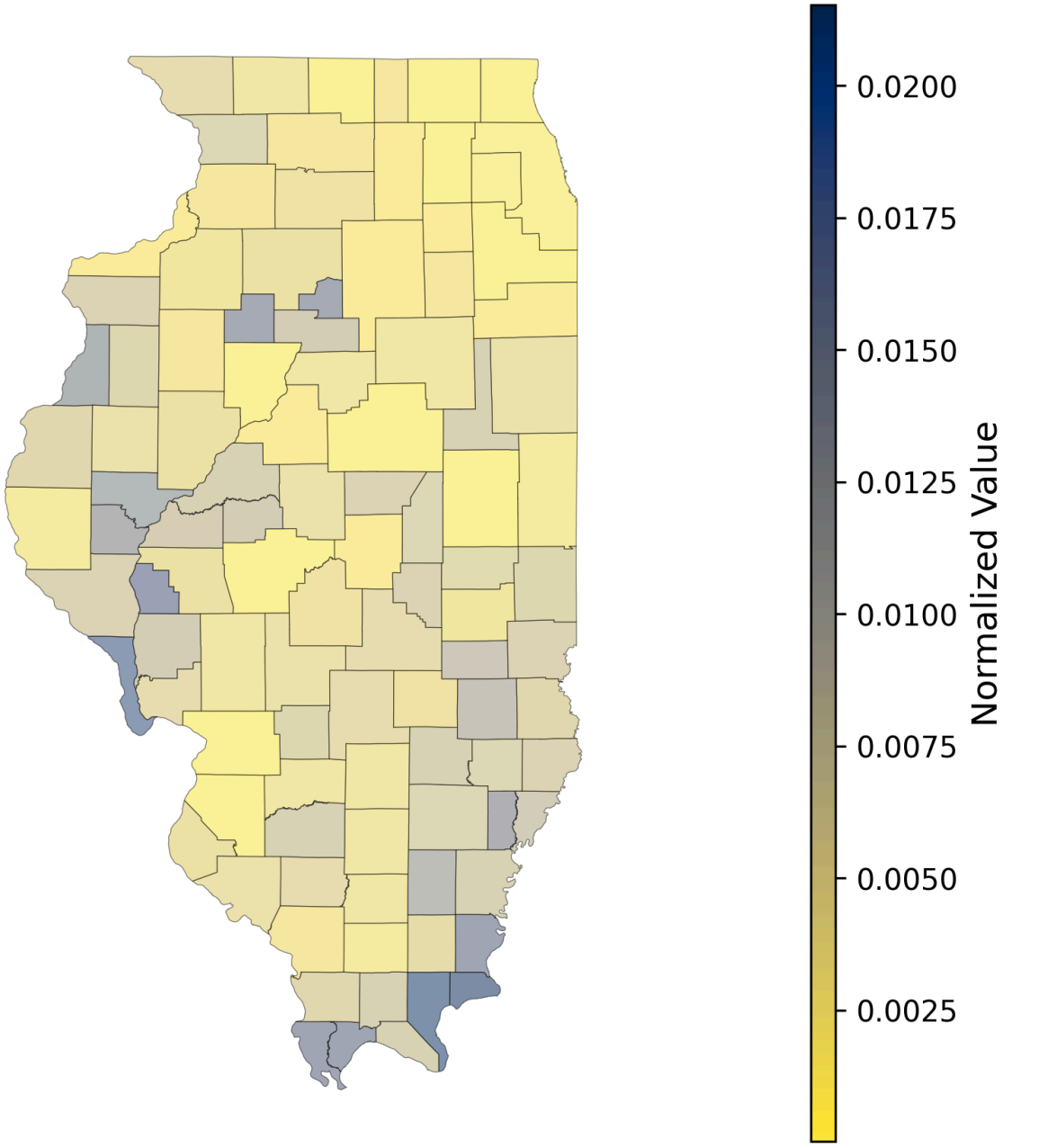
Normalized CANCER_CrudePrev - Illinois Map by County

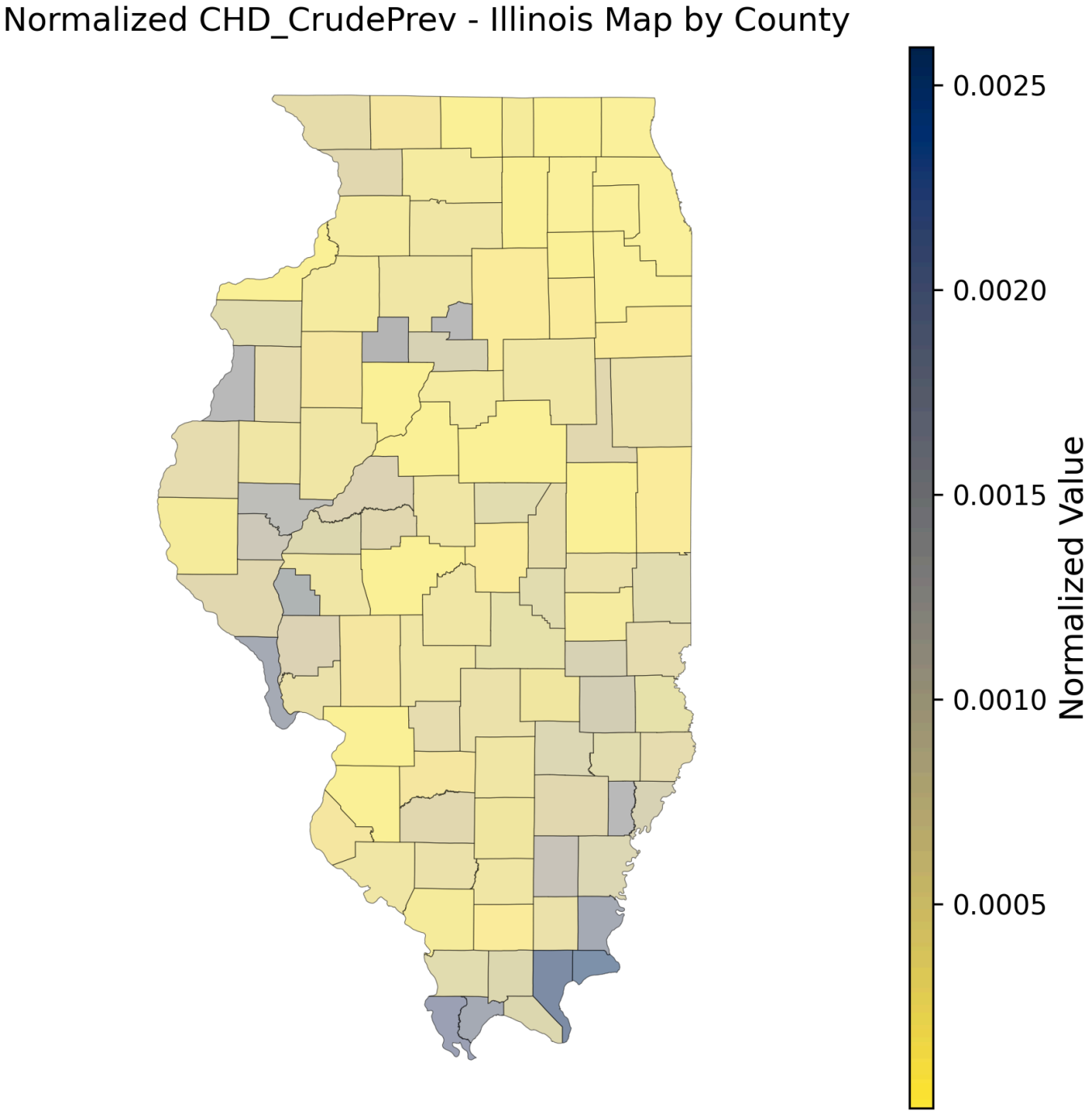


Normalized CASTHMA_CrudePrev - Illinois Map by County

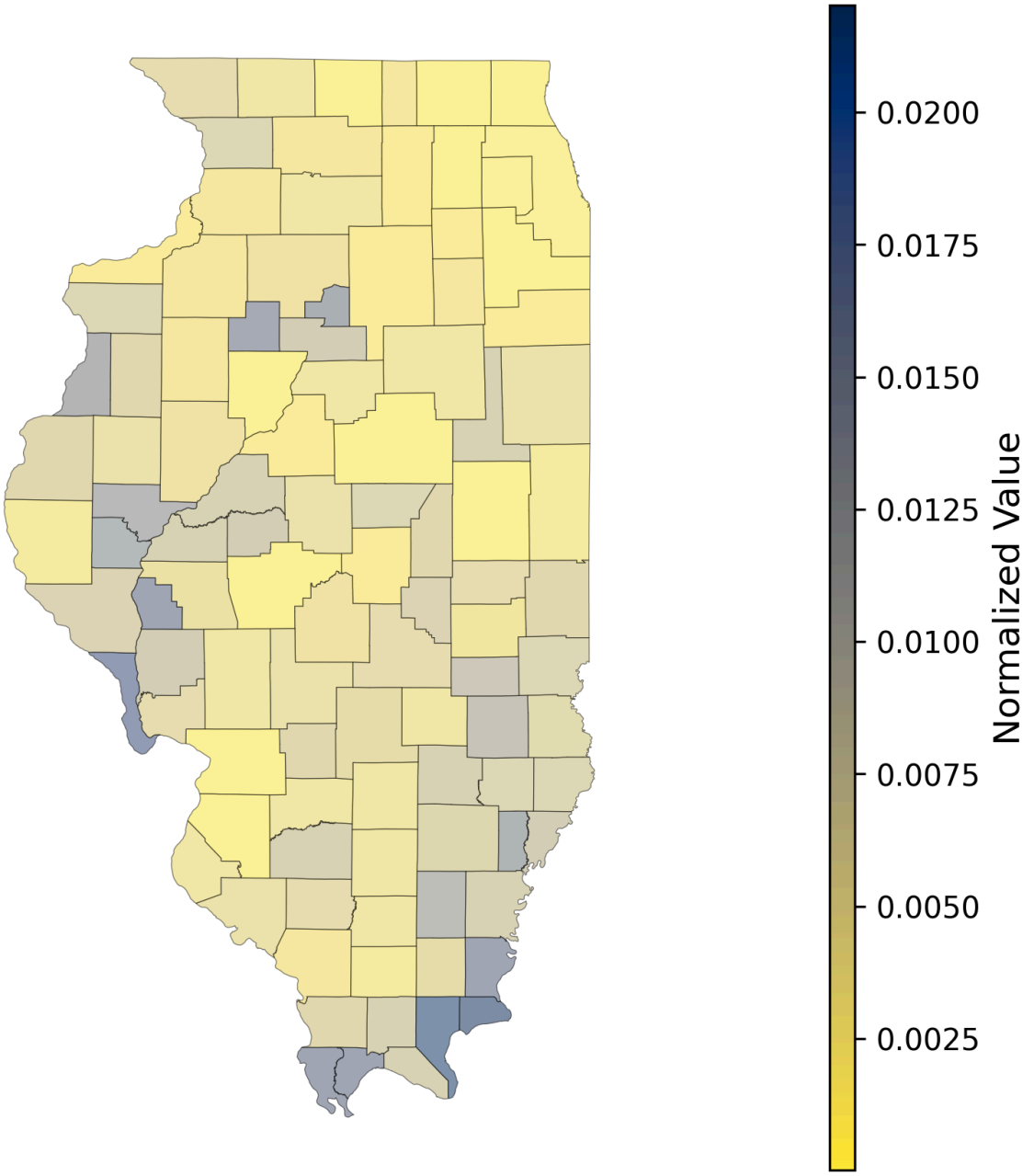


Normalized CERVICAL_CrudePrev - Illinois Map by County

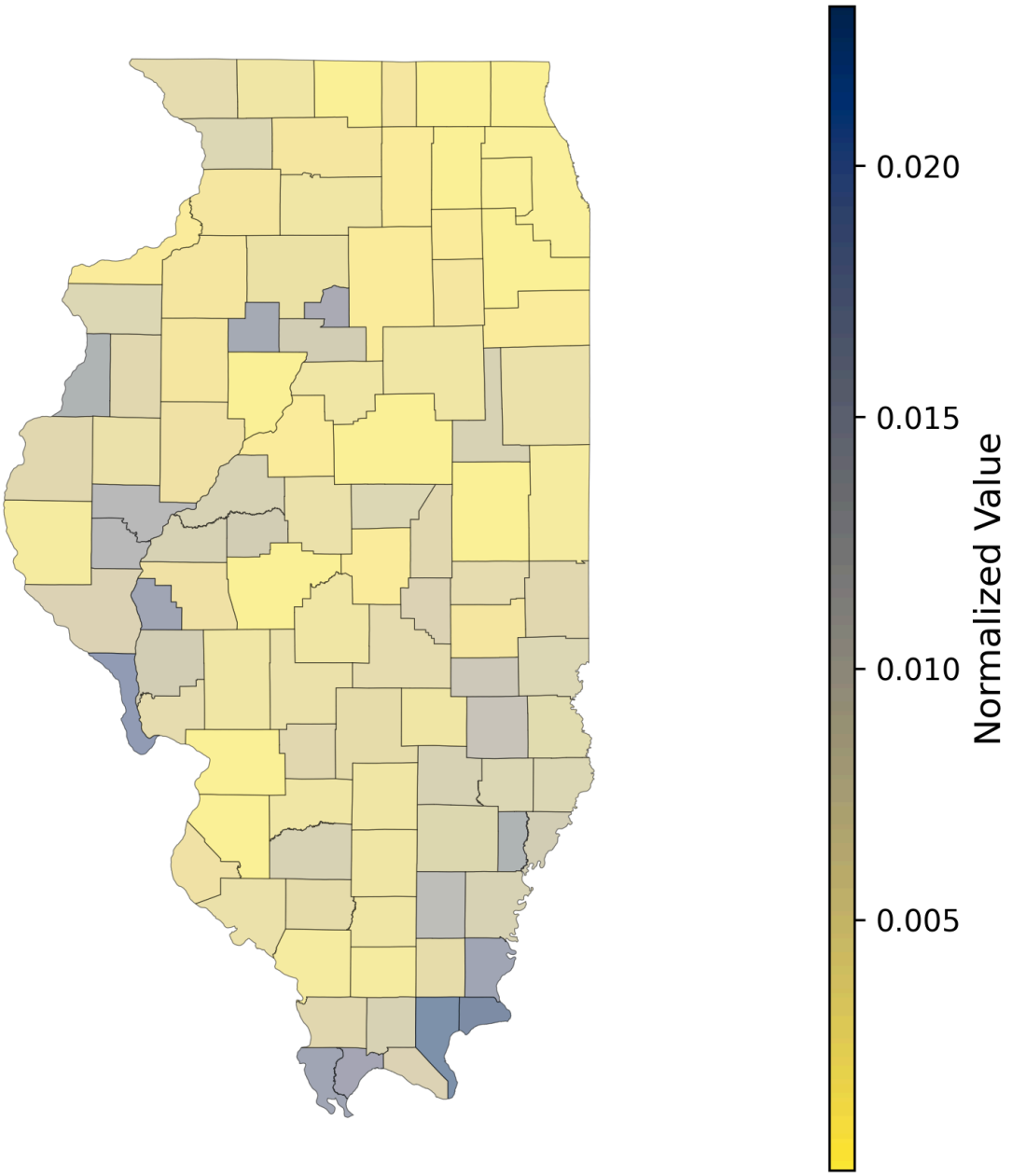



Normalized CHD_CrudePrev - Illinois Map by County
0.0025
0.0020
0.0015
0.0010
0.0005
Normalized Value

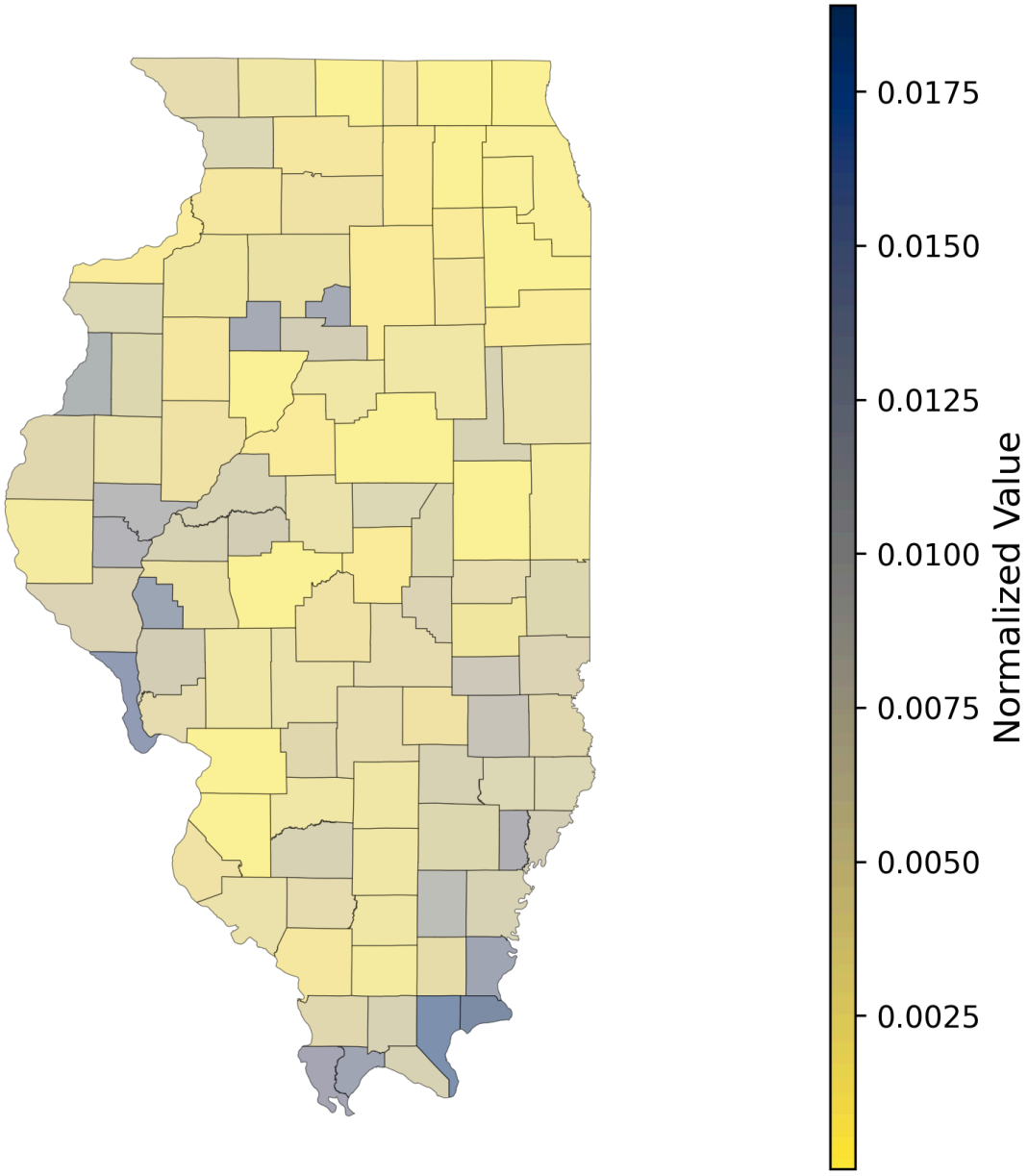
Normalized CHECKUP_CrudePrev - Illinois Map by County



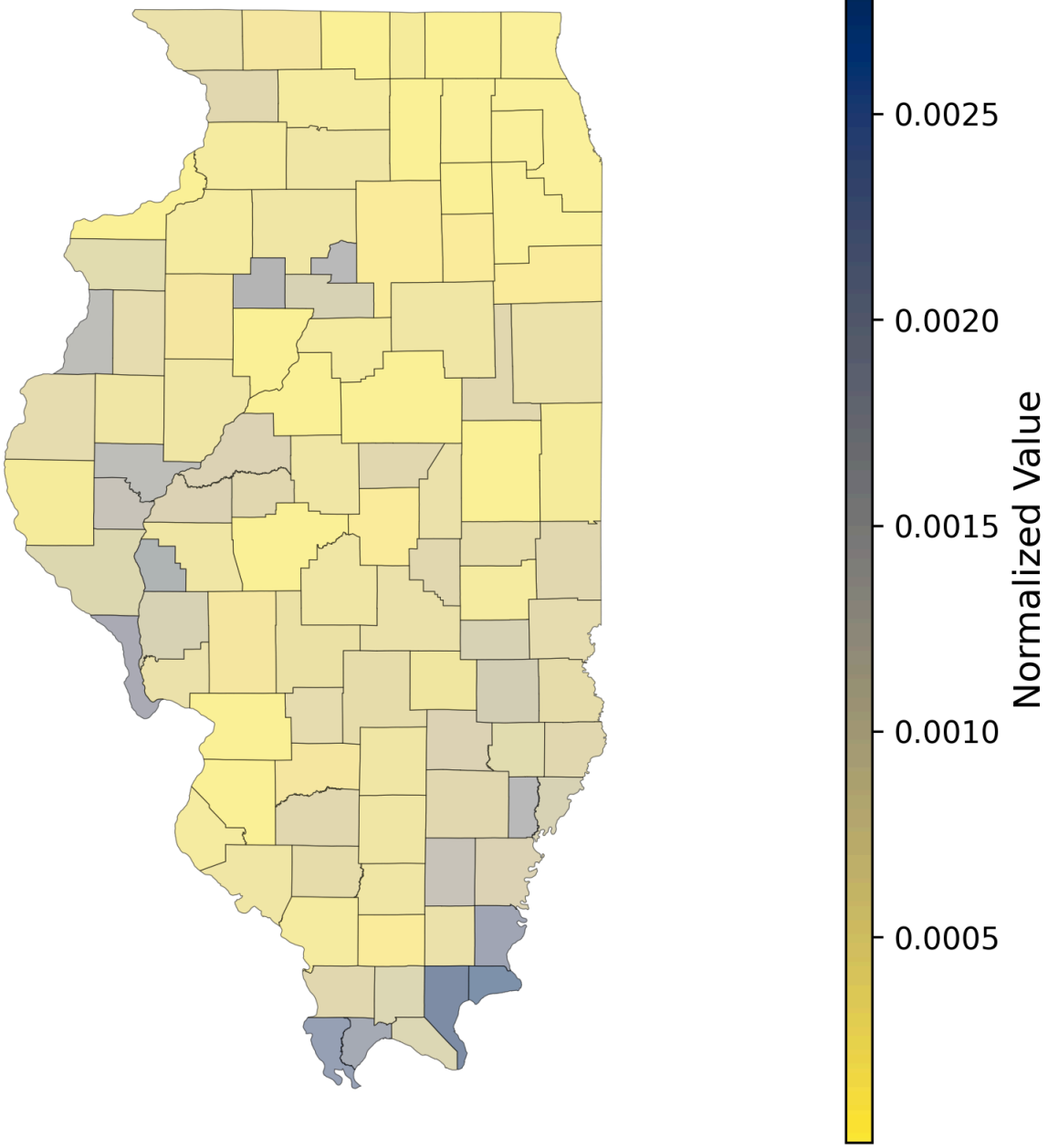
Normalized CHOLSCREEN_CrudePrev - Illinois Map by County

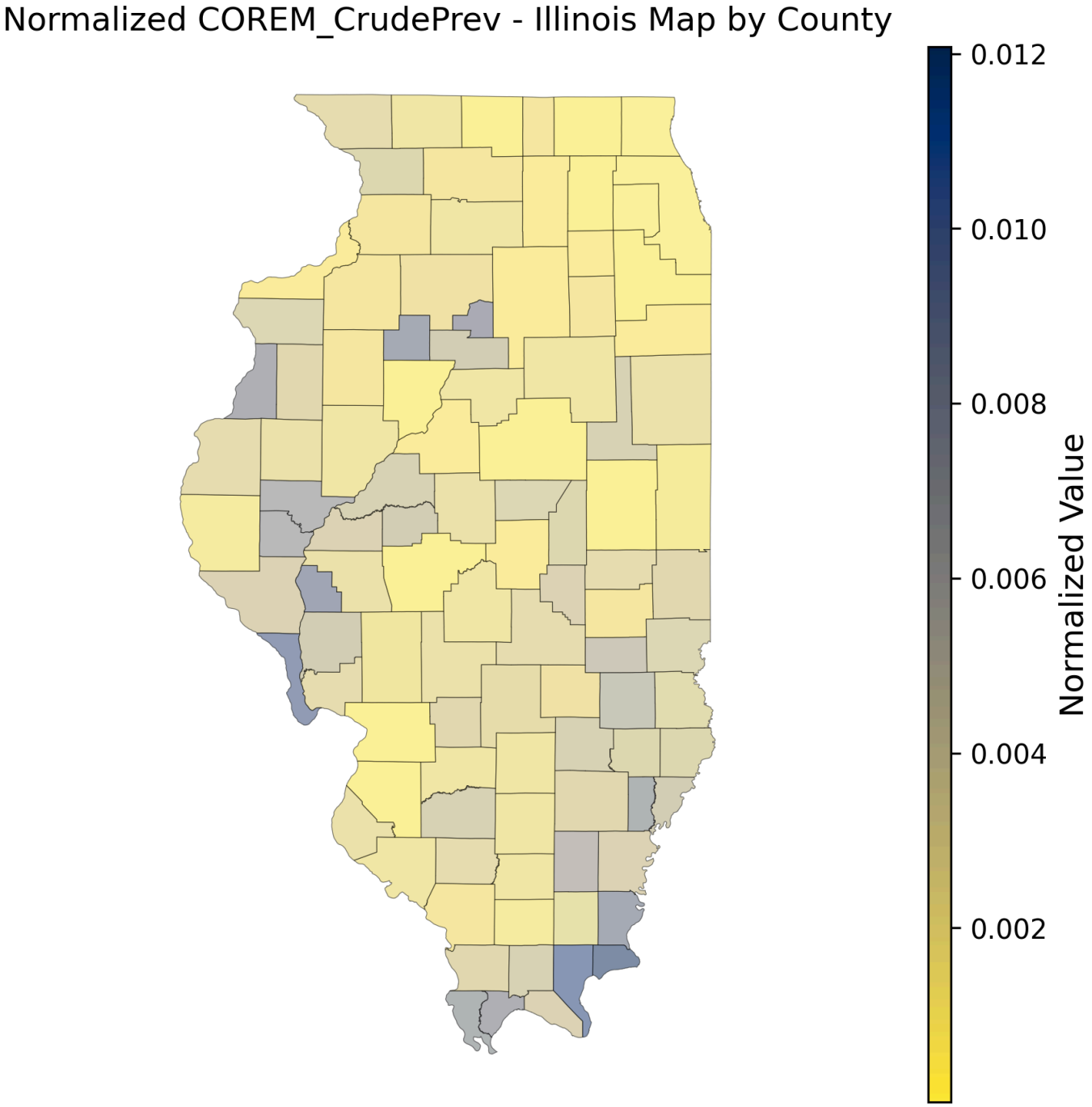


Normalized COLON_SCREEN_CrudePrev - Illinois Map by County

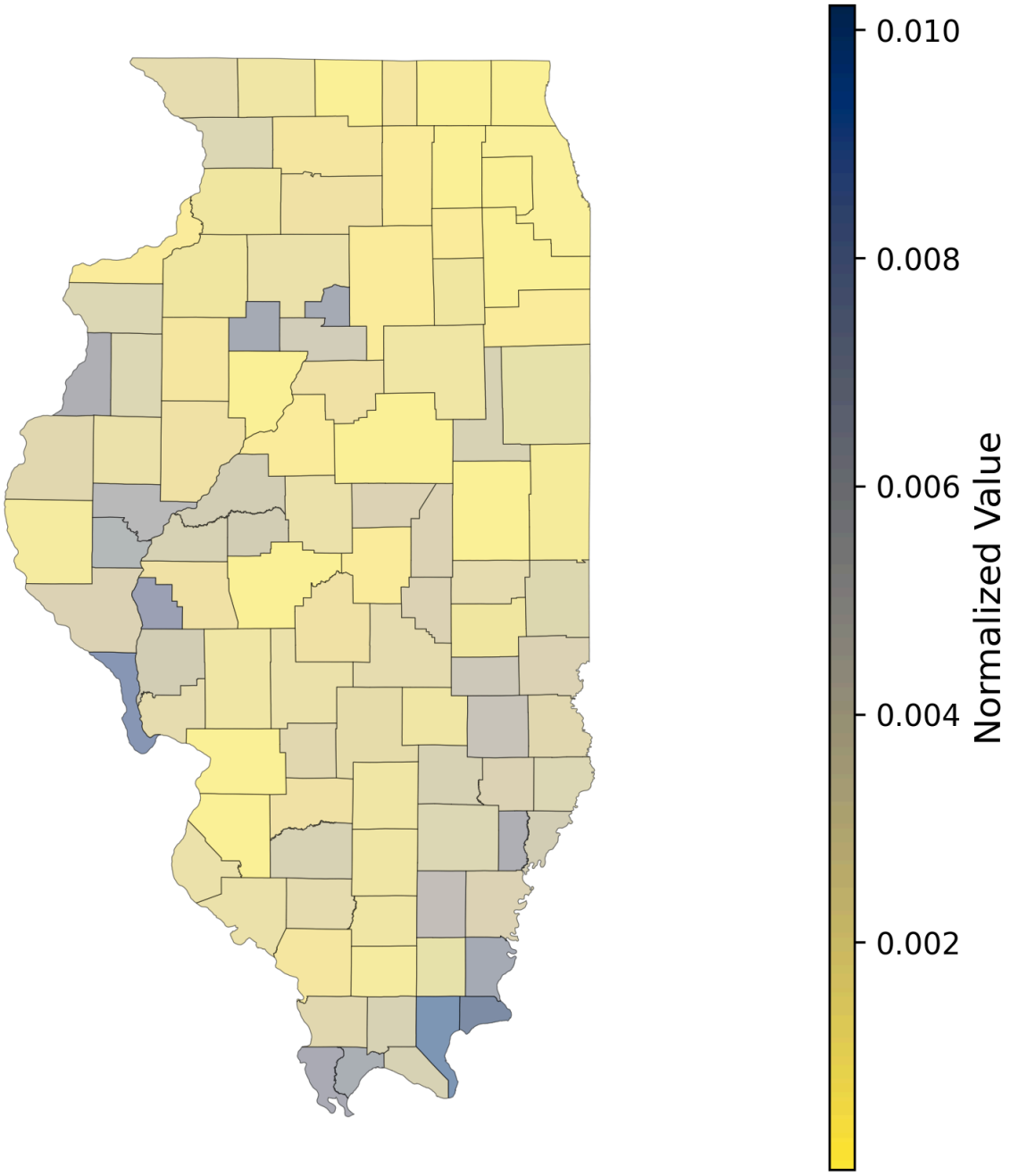


Normalized COPD_CrudePrev - Illinois Map by County

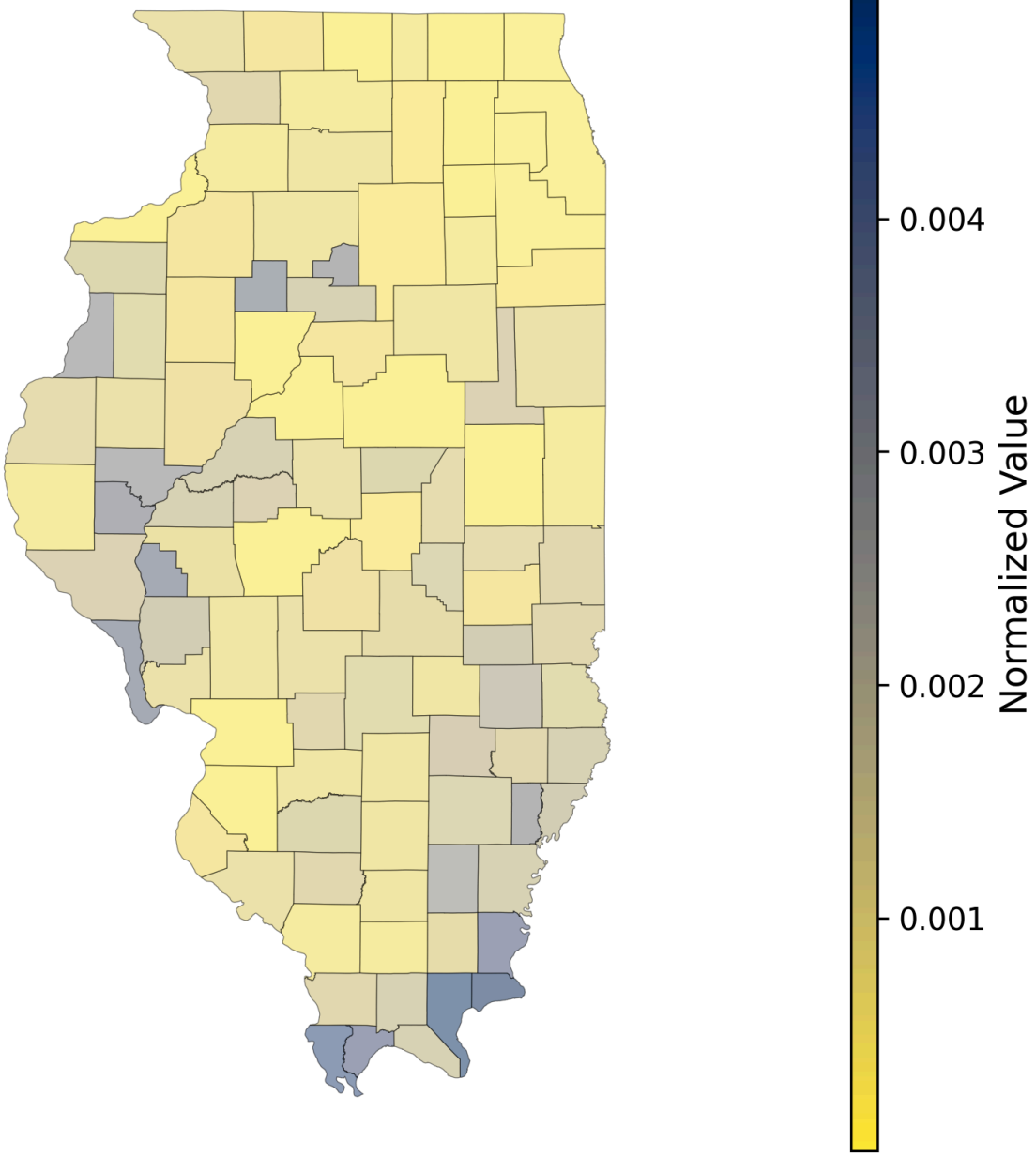



Normalized COREM_CrudePrev - Illinois Map by County
0.012
0.010
0.008
0.006
0.004
0.002
Normalized Value

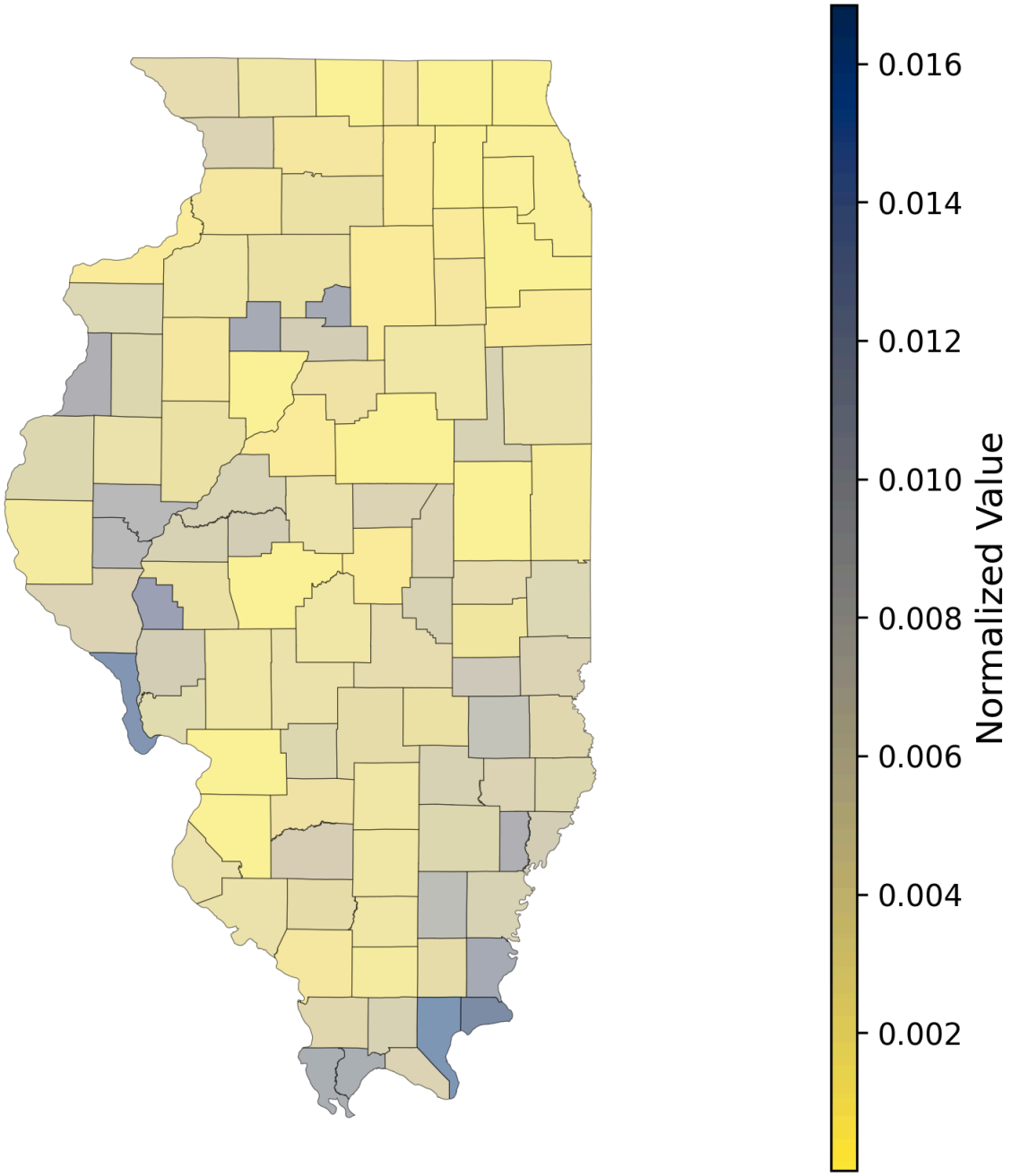
Normalized COREW_CrudePrev - Illinois Map by County



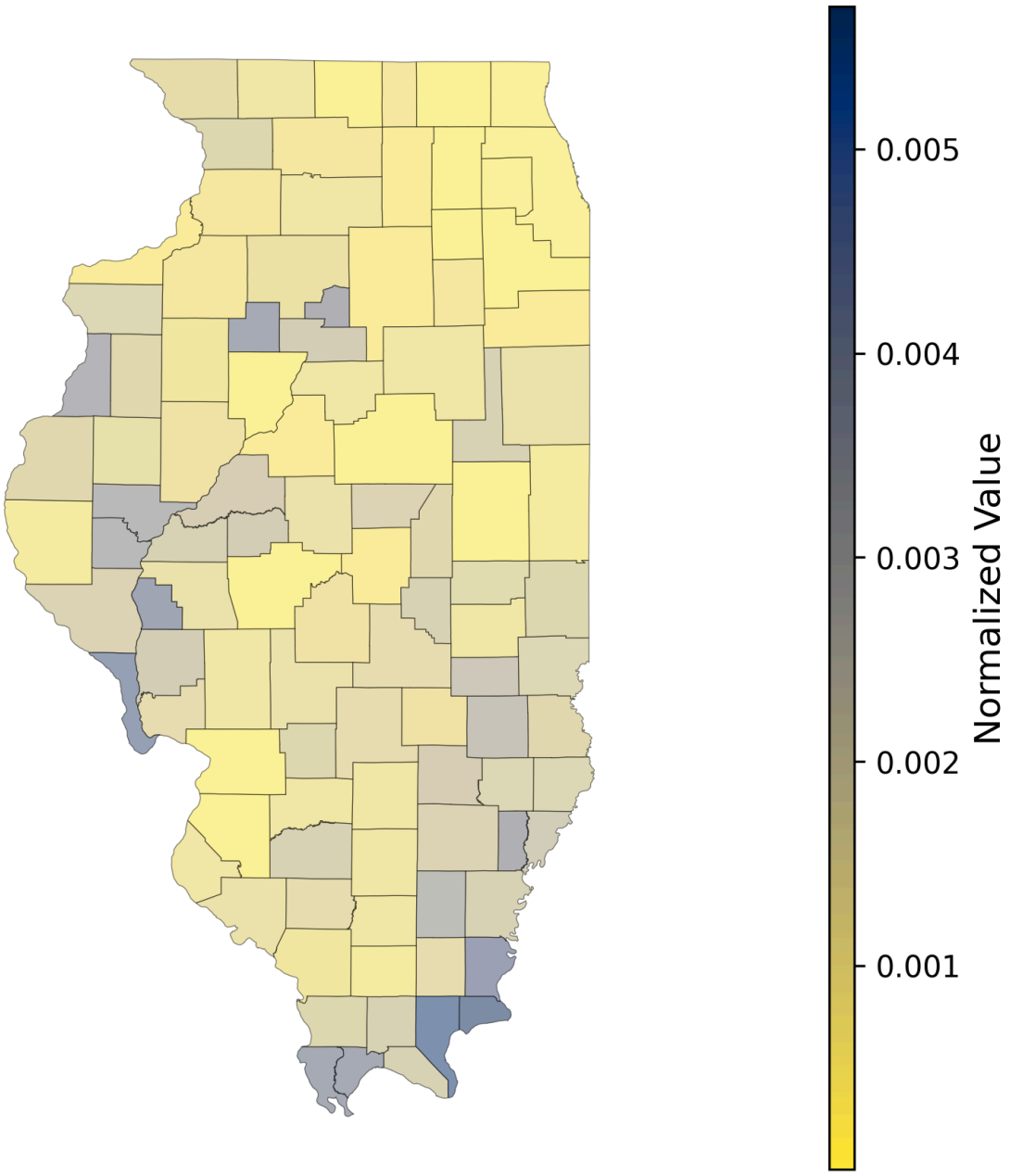
Normalized CSMOKING_CrudePrev - Illinois Map by County



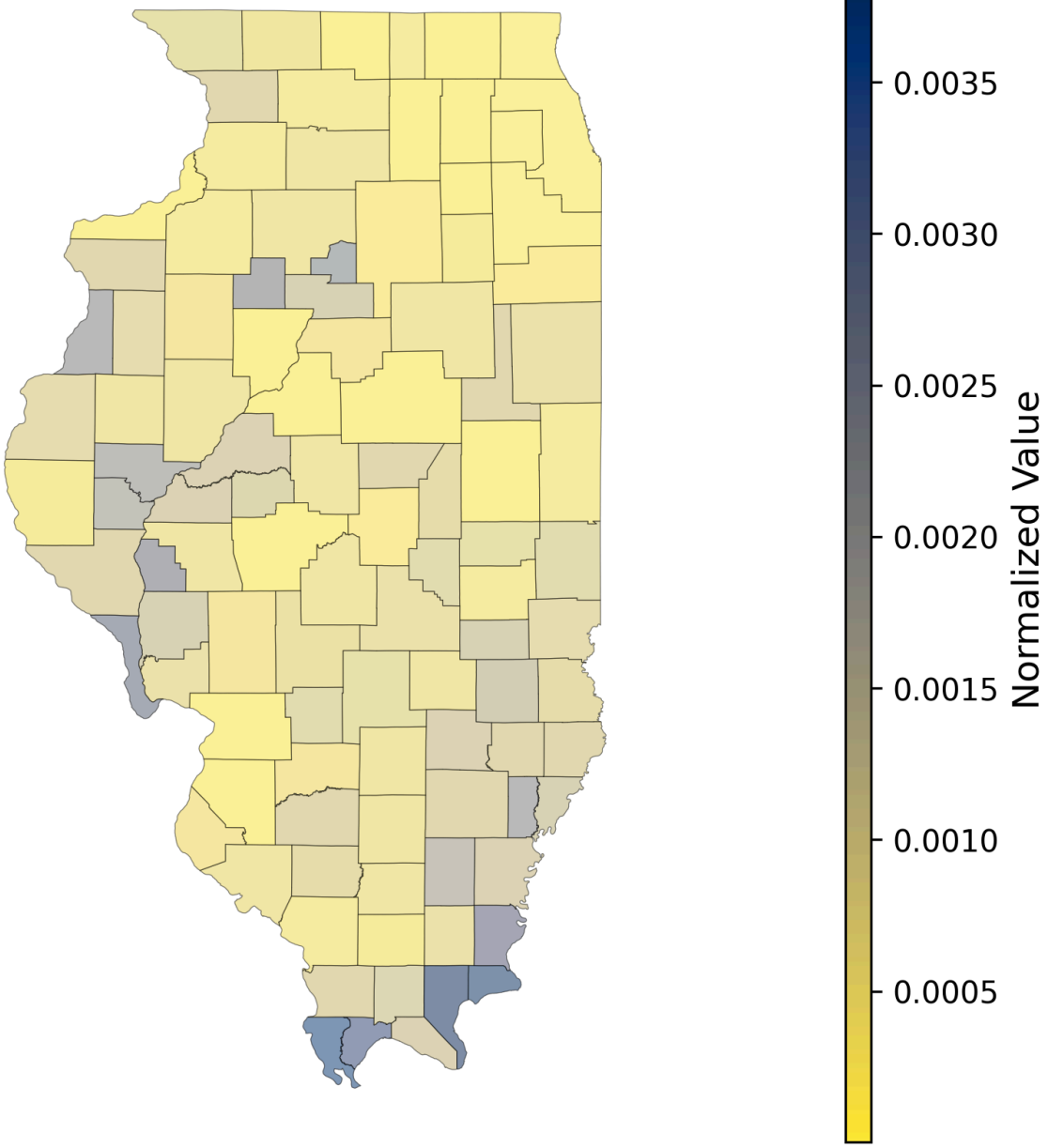
Normalized DENTAL_CrudePrev - Illinois Map by County



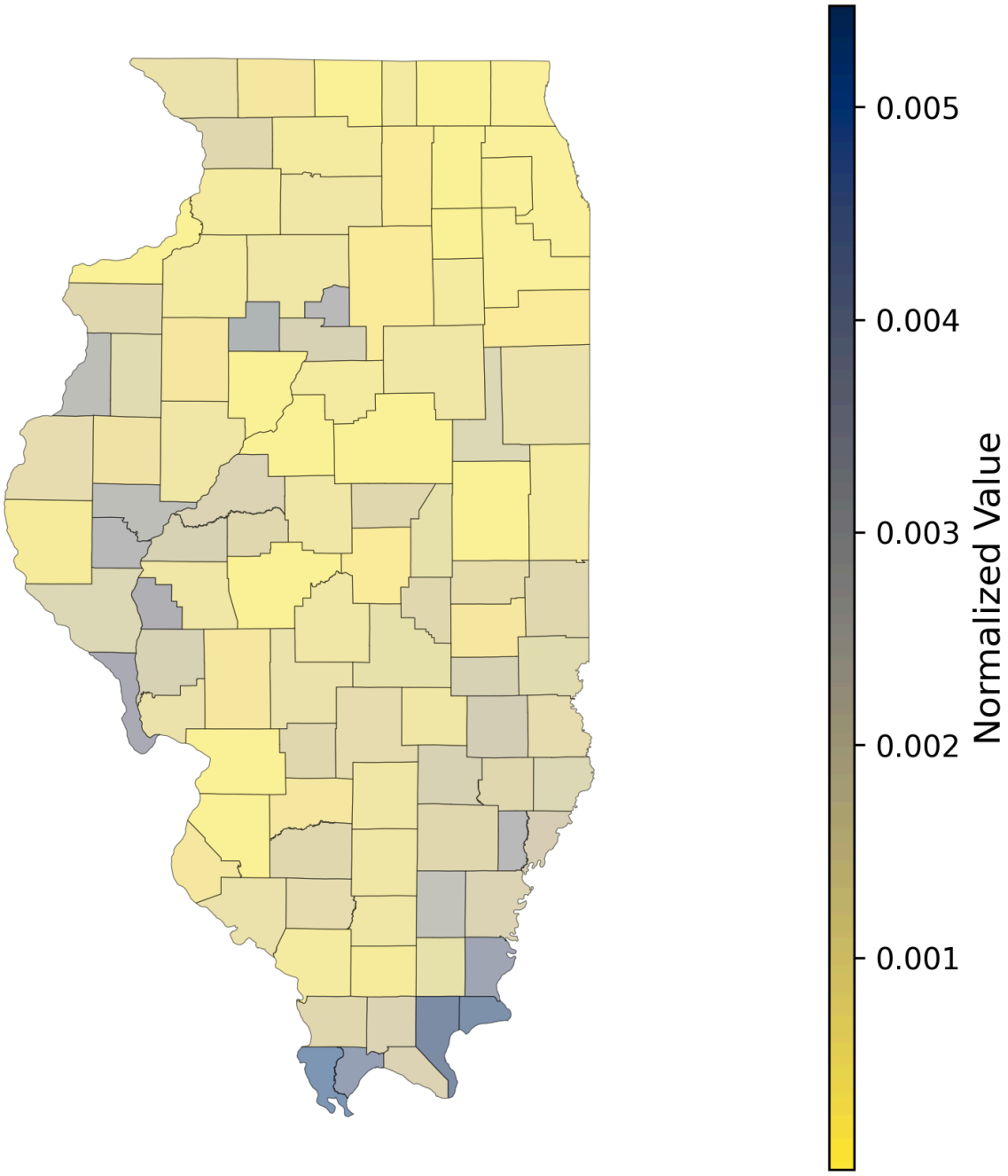
Normalized DEPRESSION_CrudePrev - Illinois Map by County



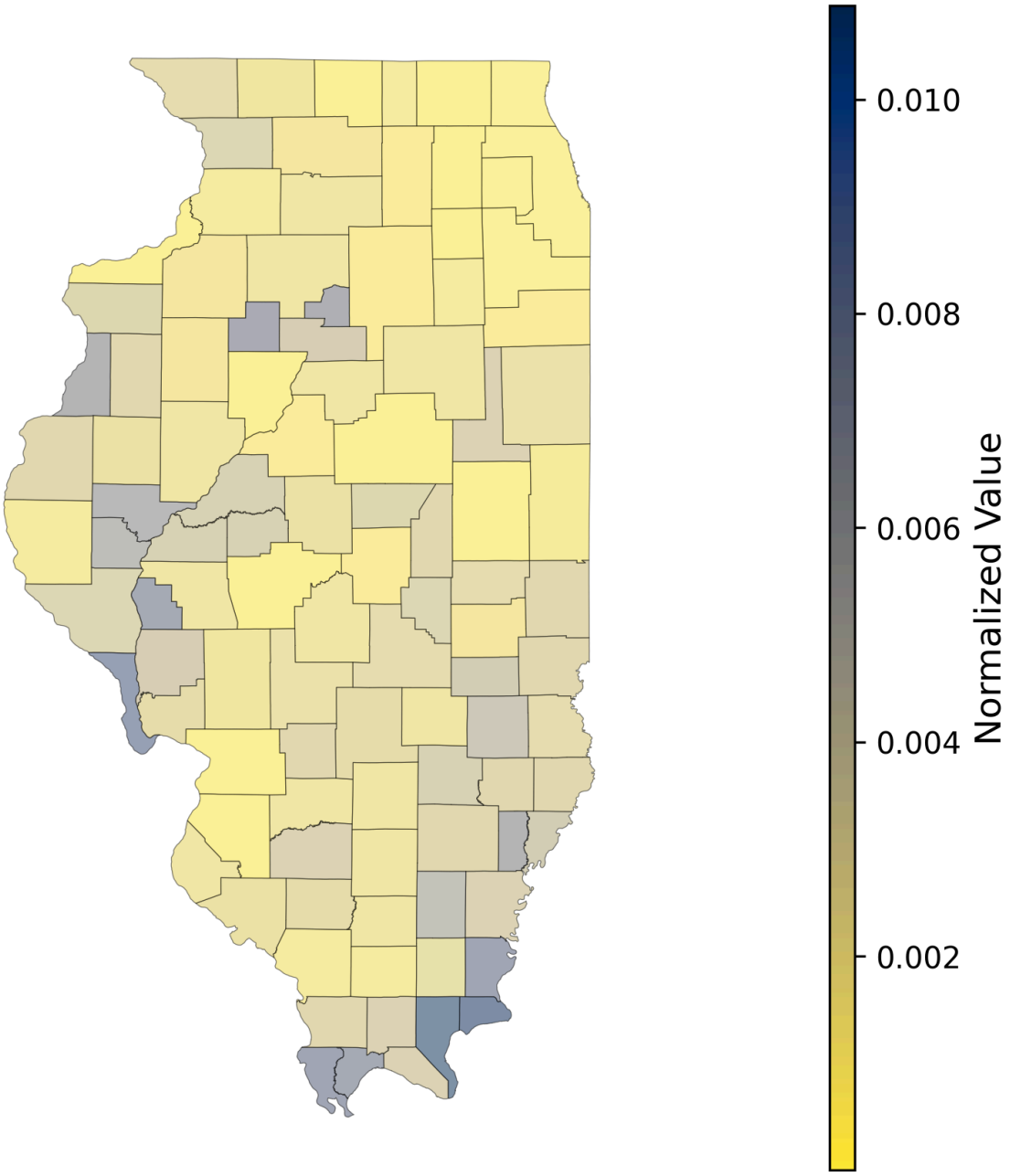
Normalized DIABETES_CrudePrev - Illinois Map by County



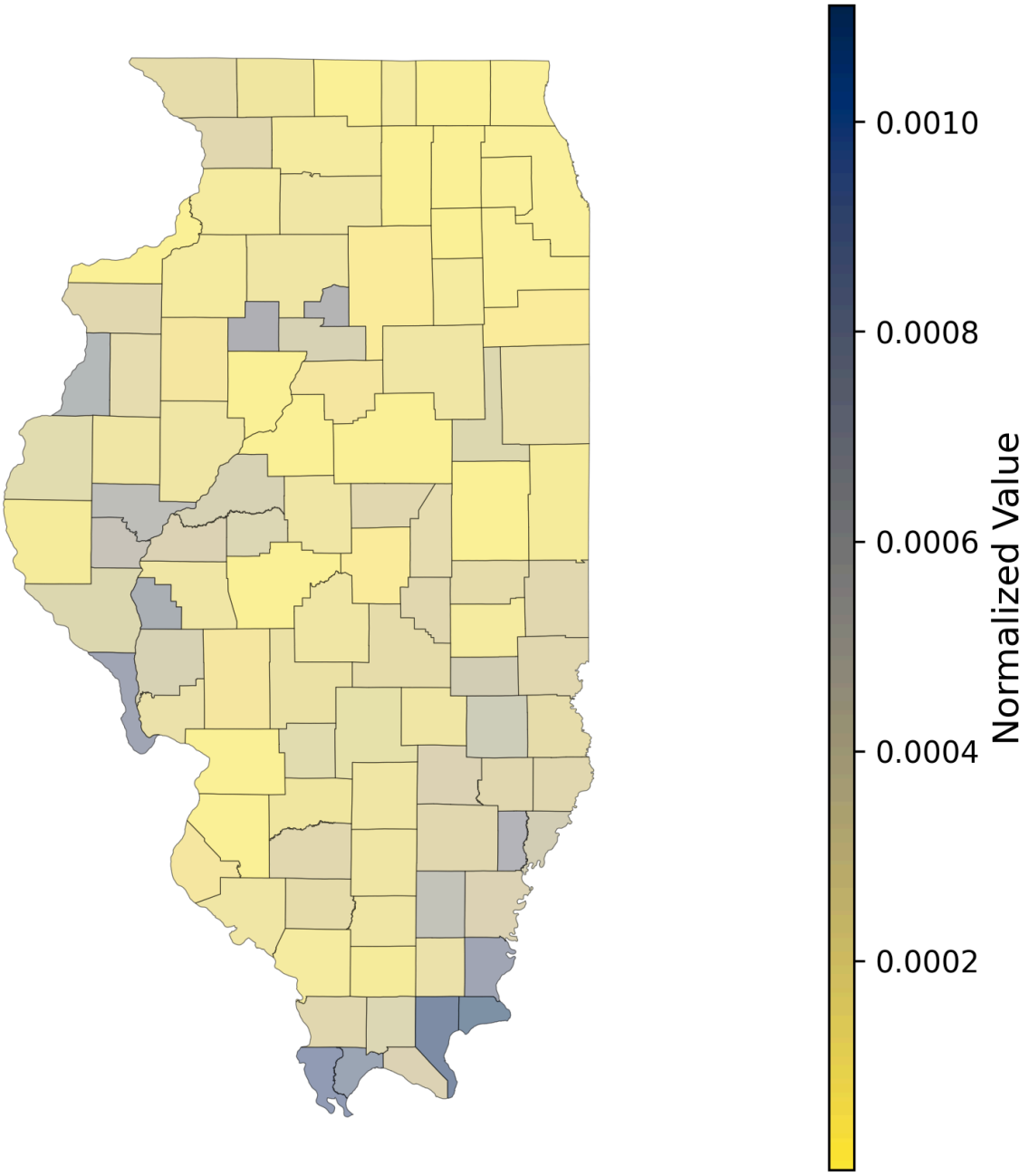
Normalized GHLTH_CrudePrev - Illinois Map by County



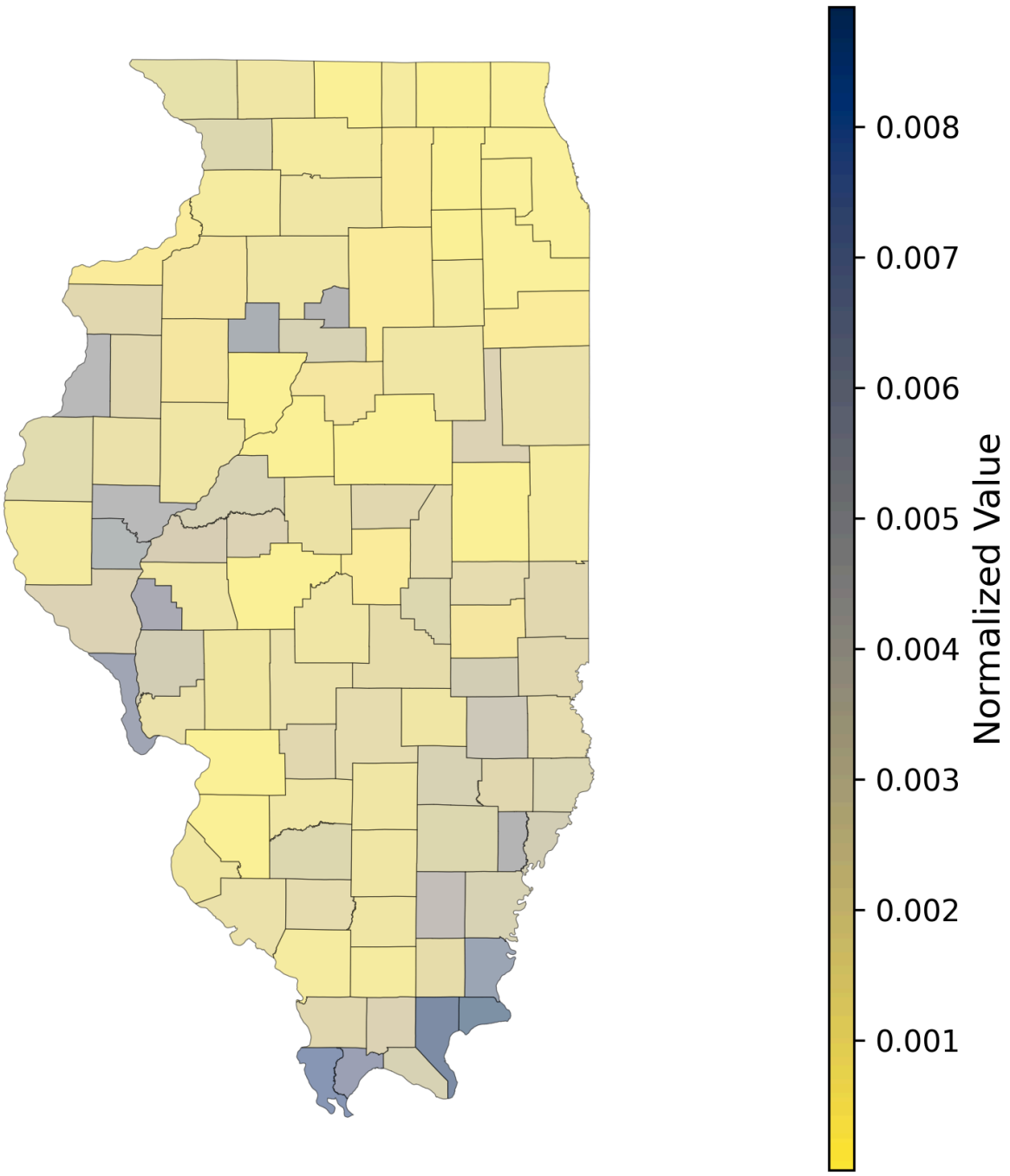
Normalized HIGHCHOL_CrudePrev - Illinois Map by County



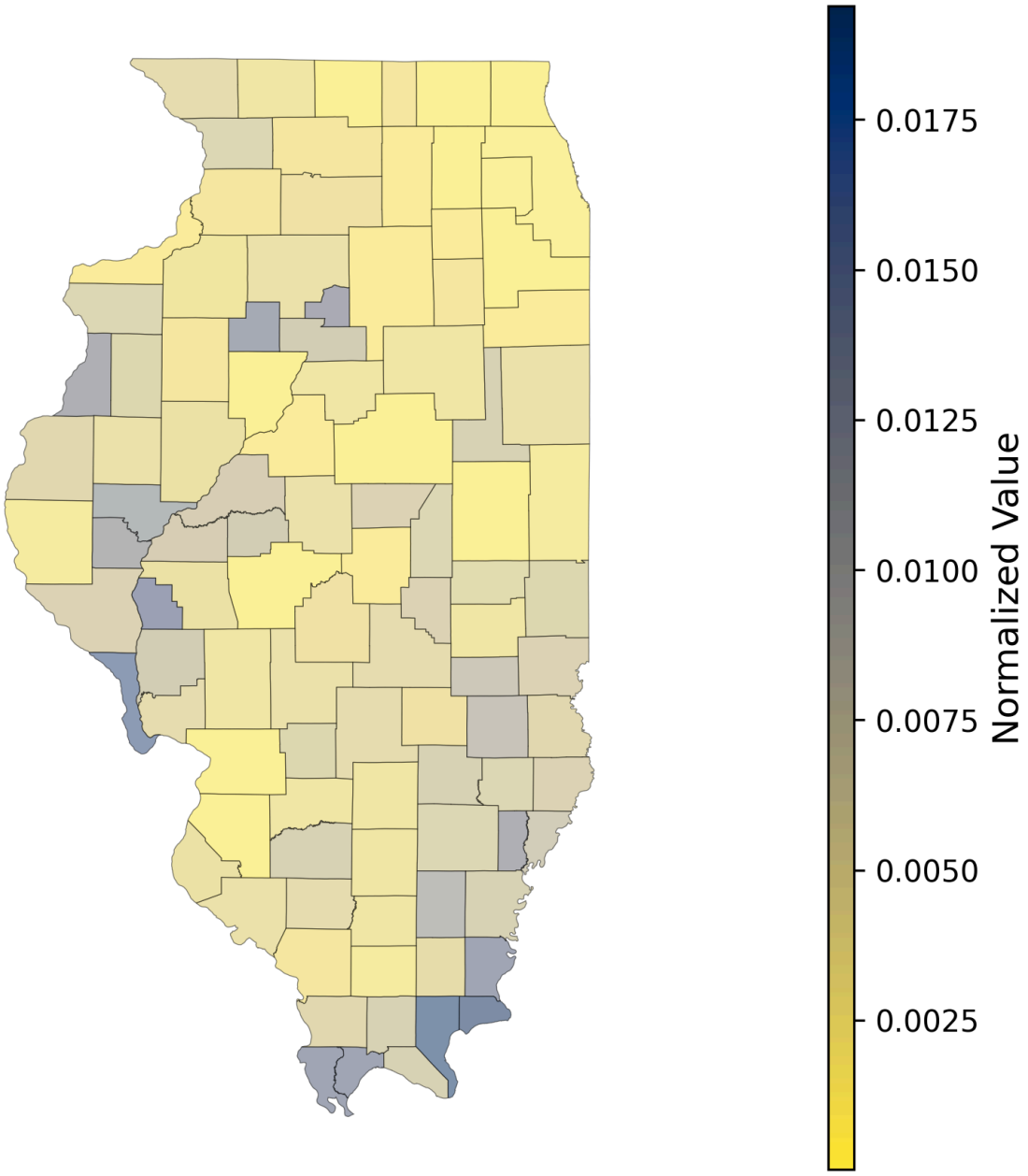
Normalized KIDNEY_CrudePrev - Illinois Map by County



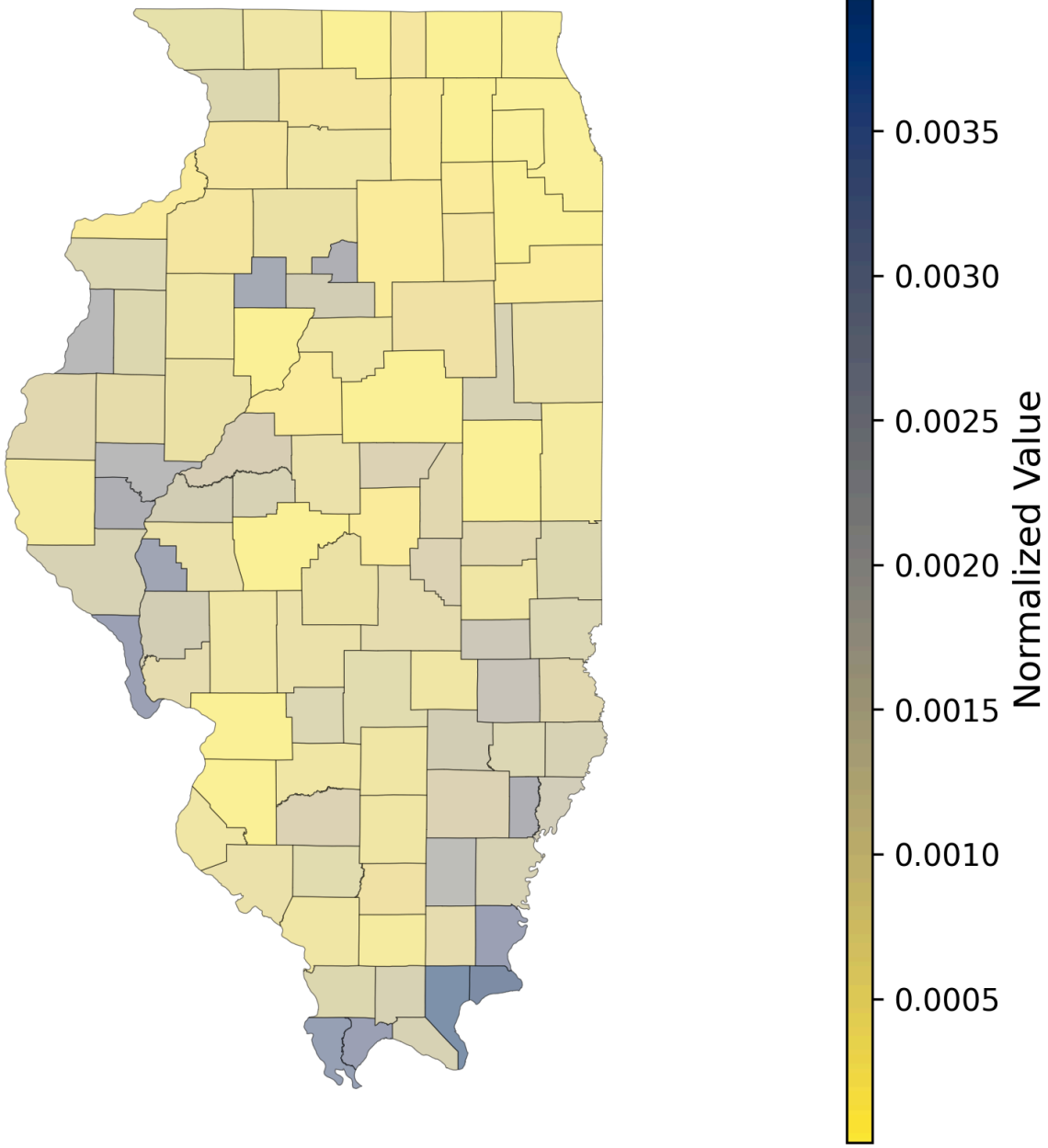
Normalized LPA_CrudePrev - Illinois Map by County



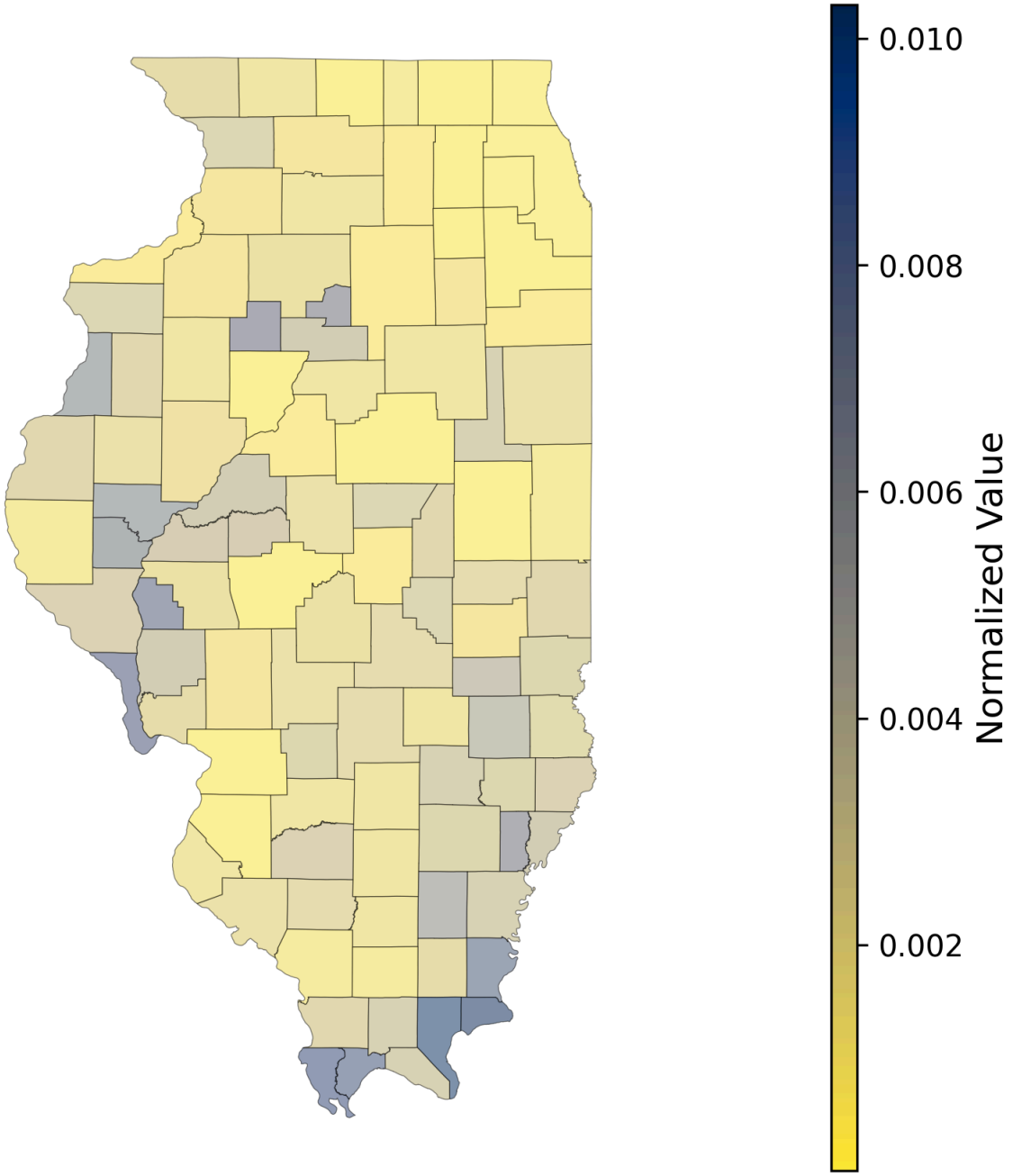
Normalized MAMMOUSE_CrudePrev - Illinois Map by County



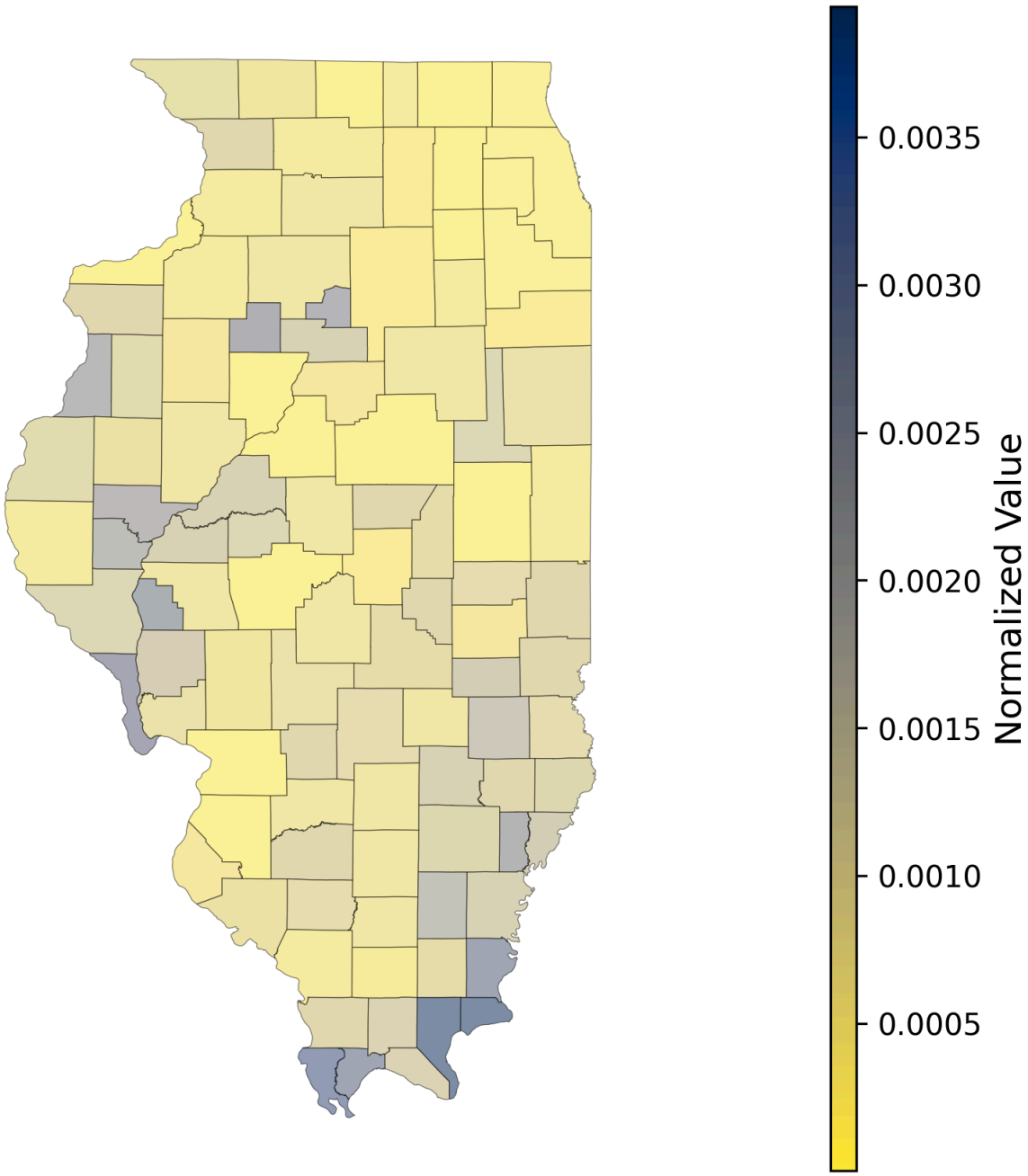
Normalized MHLTH_CrudePrev - Illinois Map by County



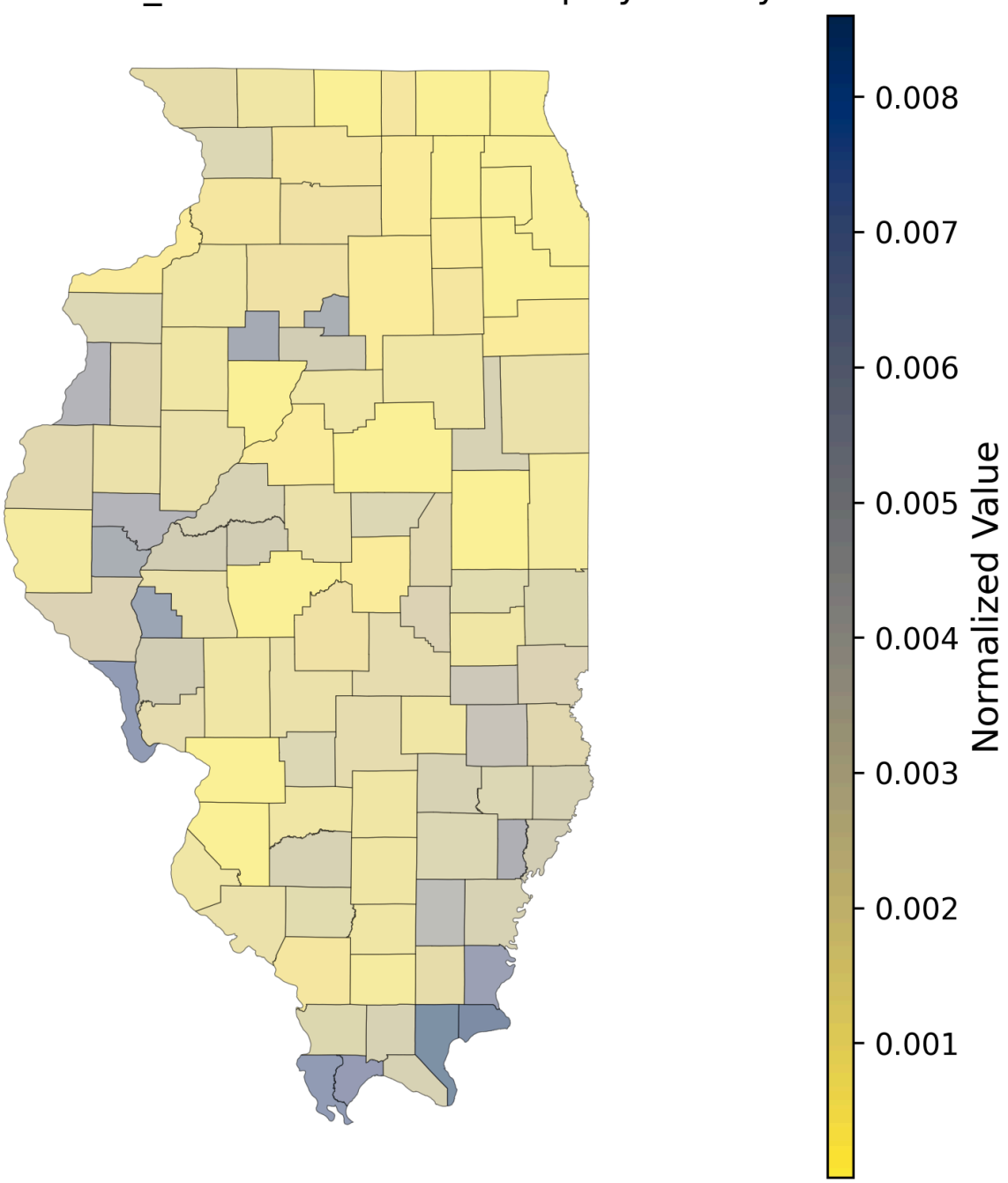
Normalized OBESITY_CrudePrev - Illinois Map by County



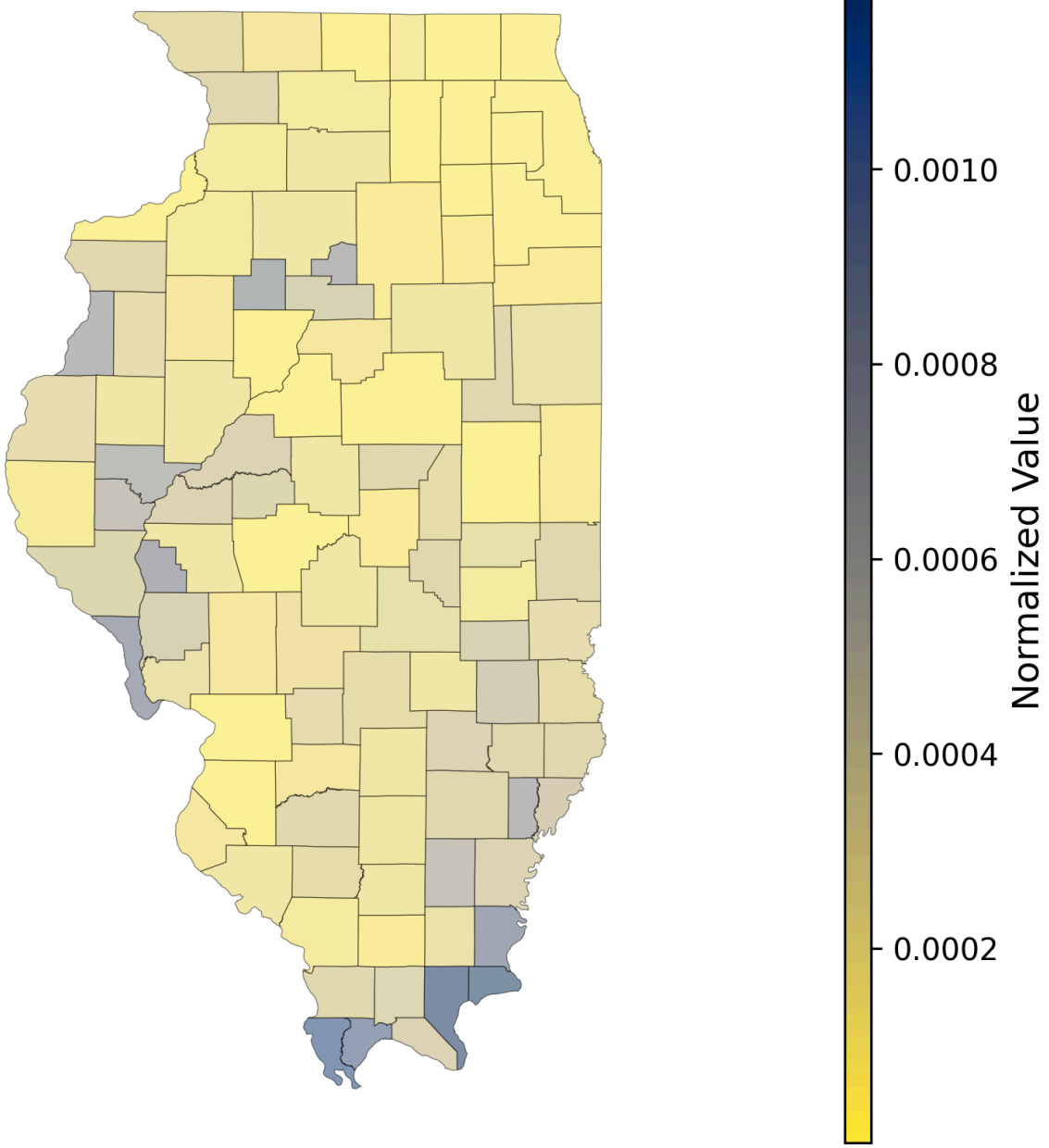
Normalized PHLTH_CrudePrev - Illinois Map by County



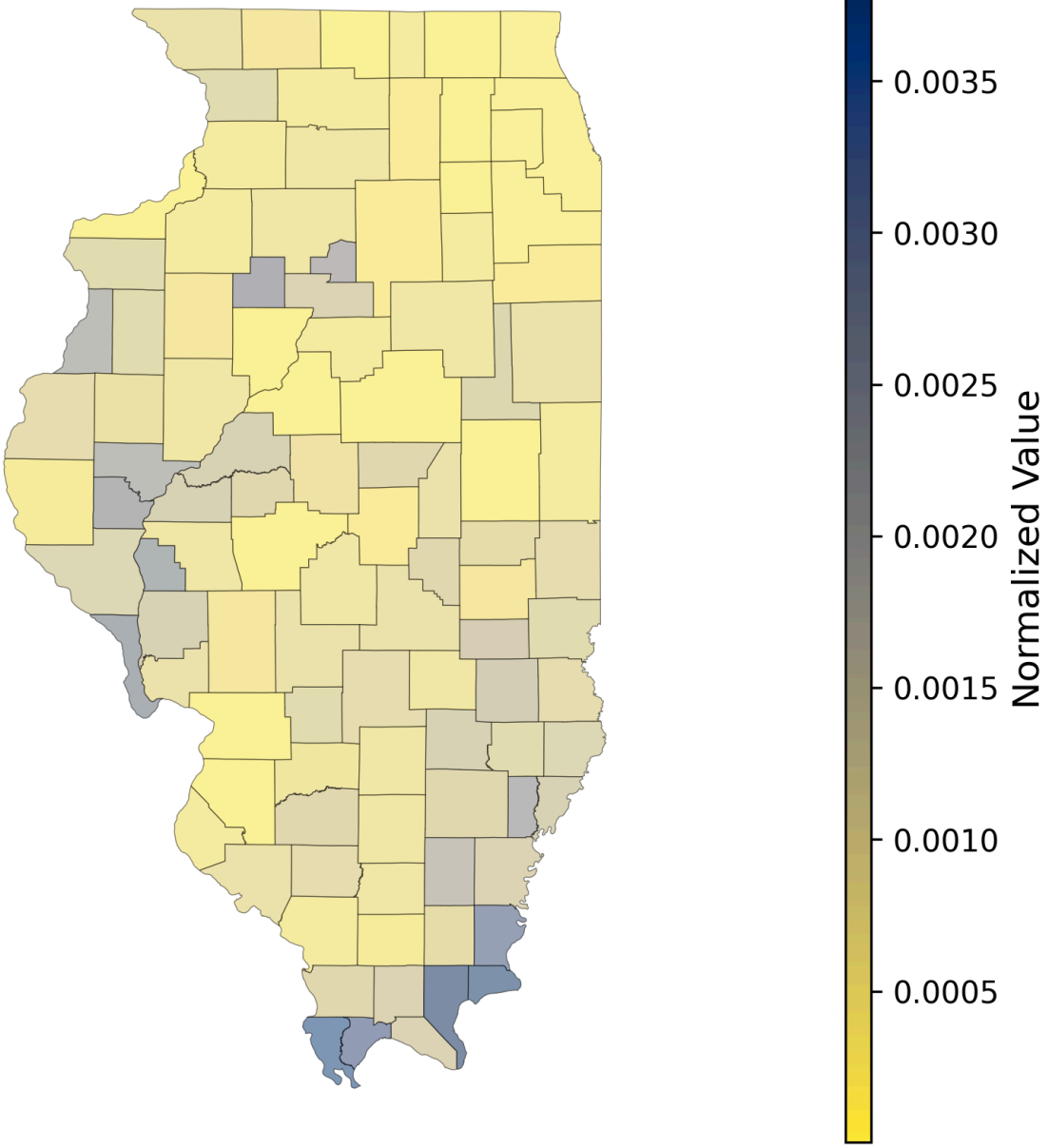
Normalized SLEEP_CrudePrev - Illinois Map by County



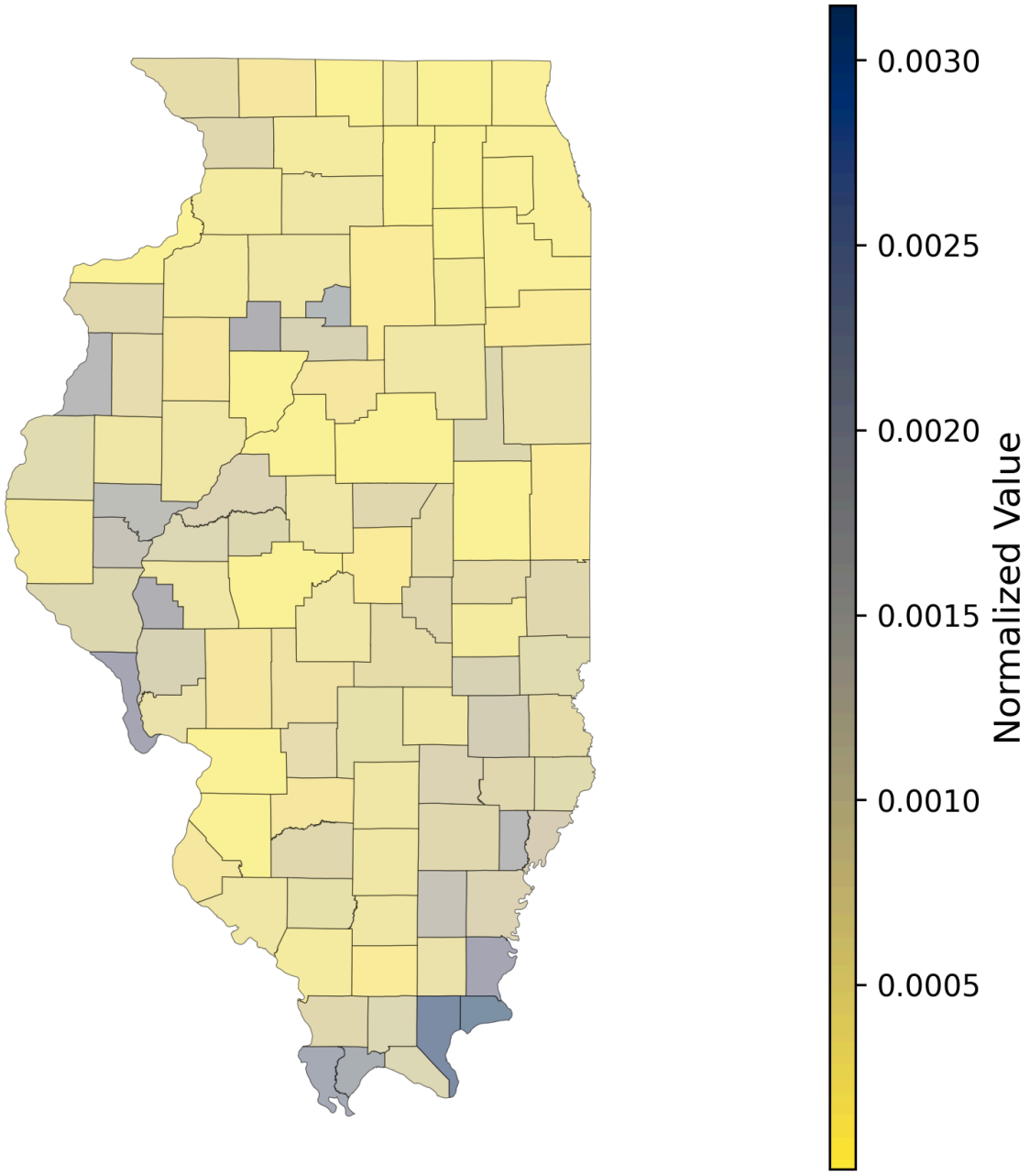
Normalized STROKE_CrudePrev - Illinois Map by County



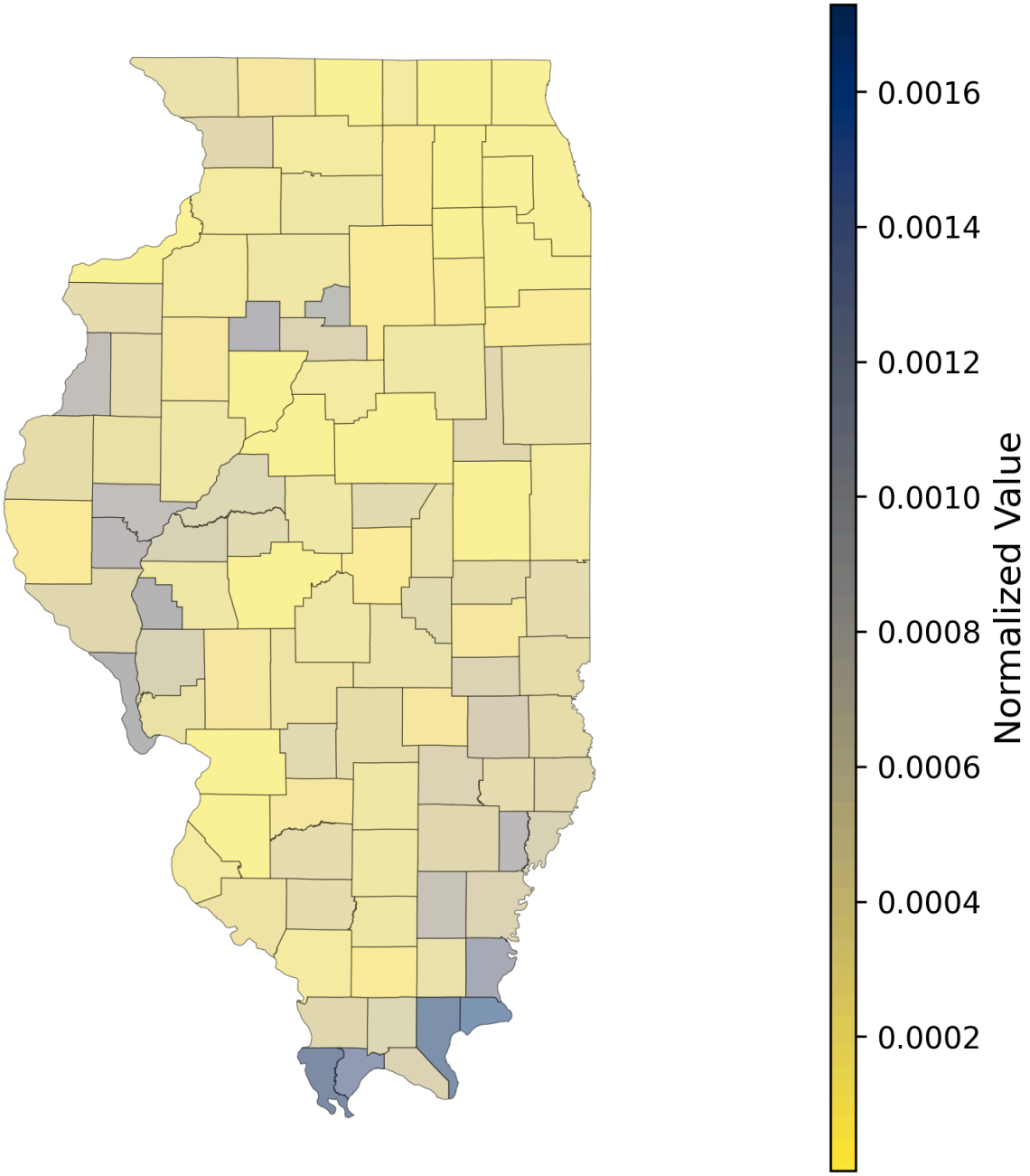
Normalized TEETHLOST_CrudePrev - Illinois Map by County



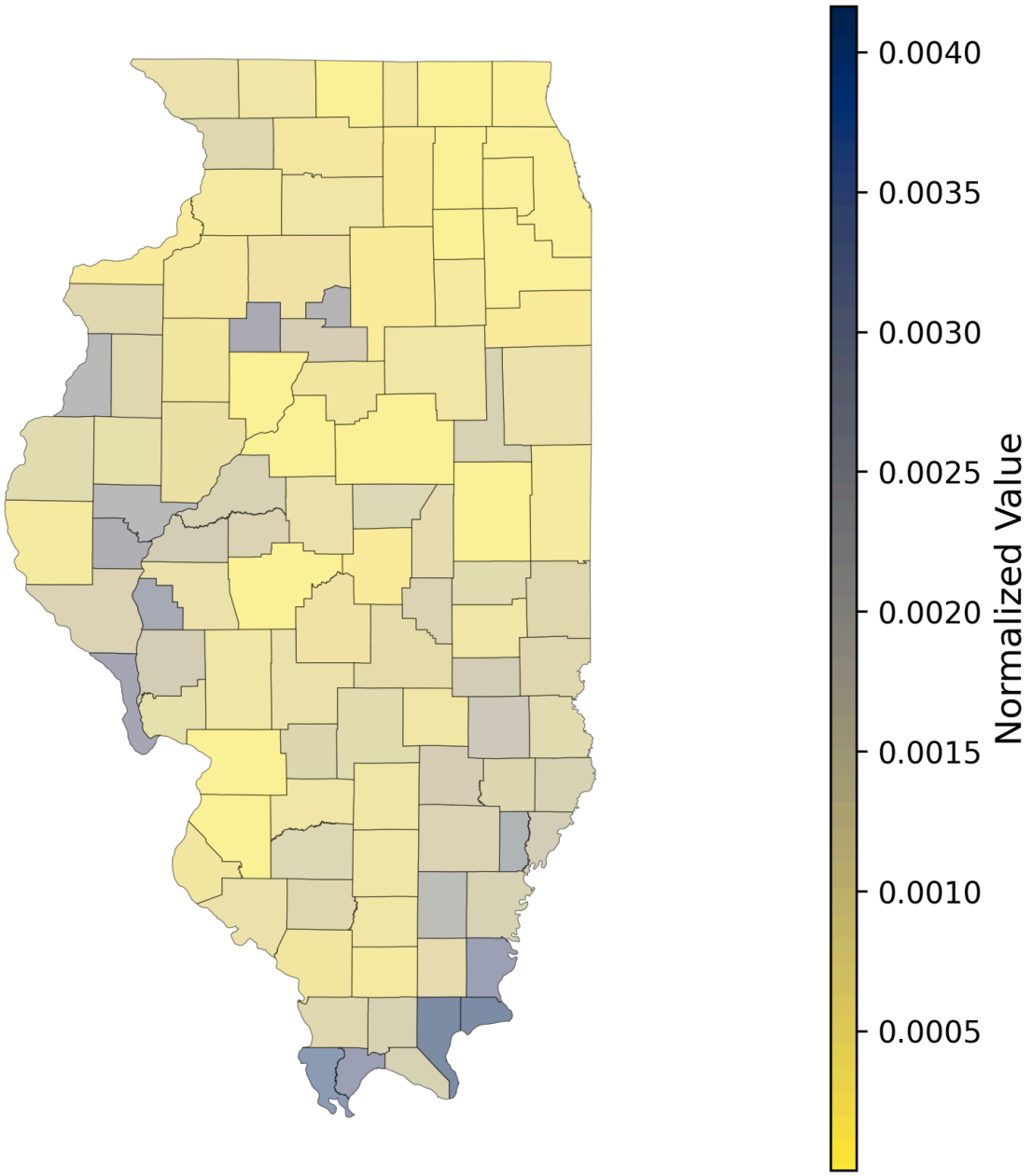
Normalized HEARING_CrudePrev - Illinois Map by County



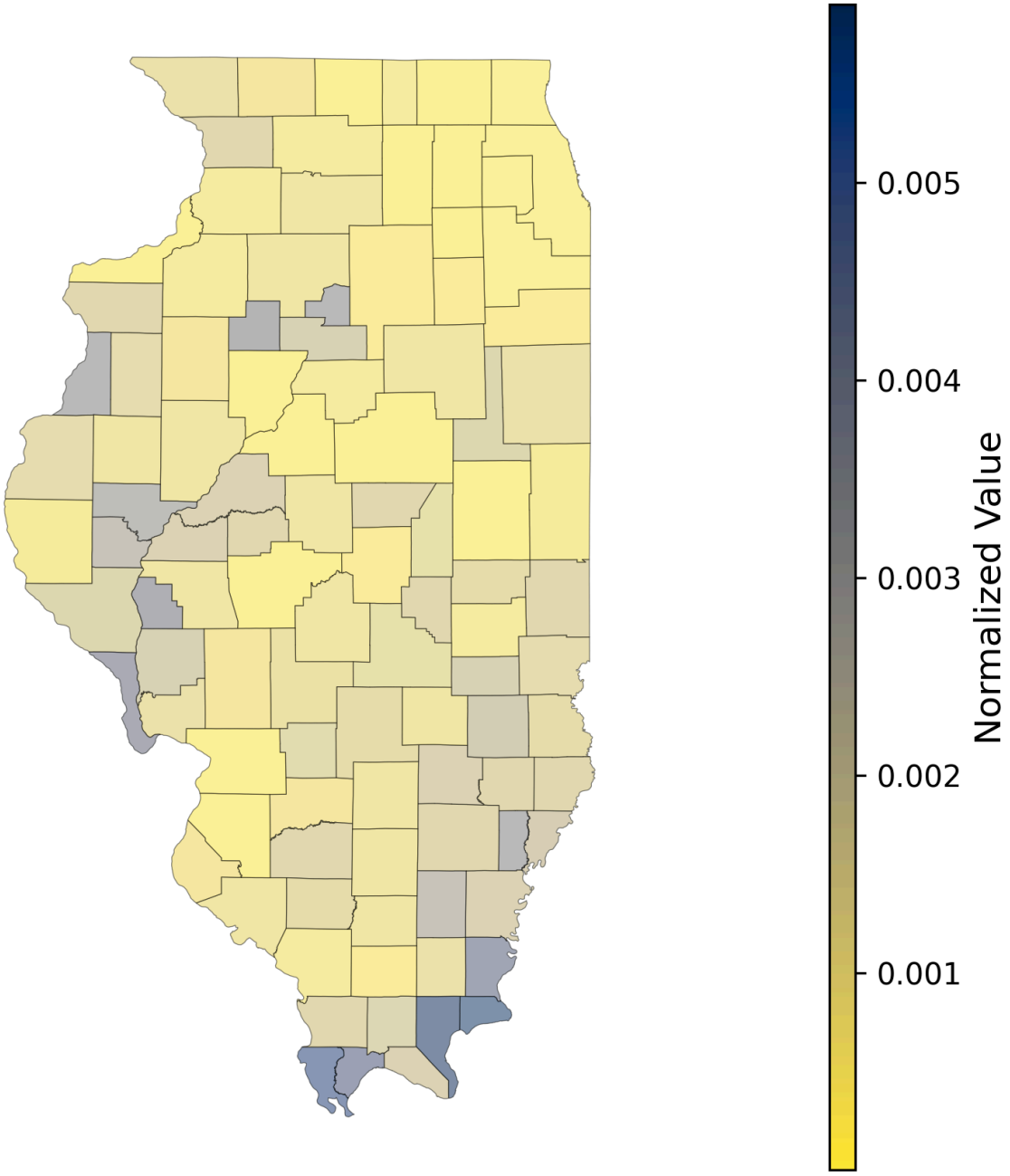
Normalized VISION_CrudePrev - Illinois Map by County

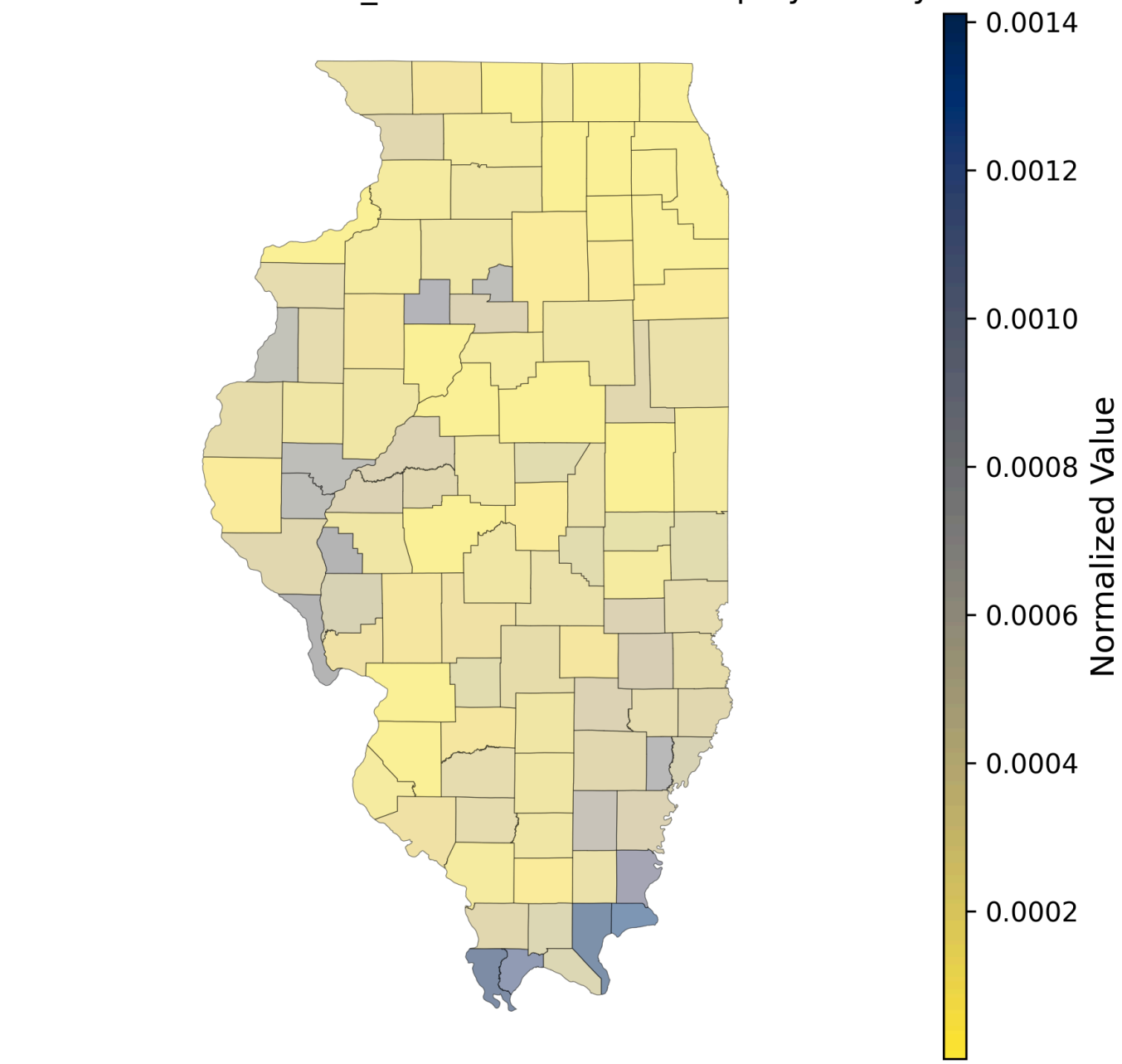


Normalized COGNITION_CrudePrev - Illinois Map by County

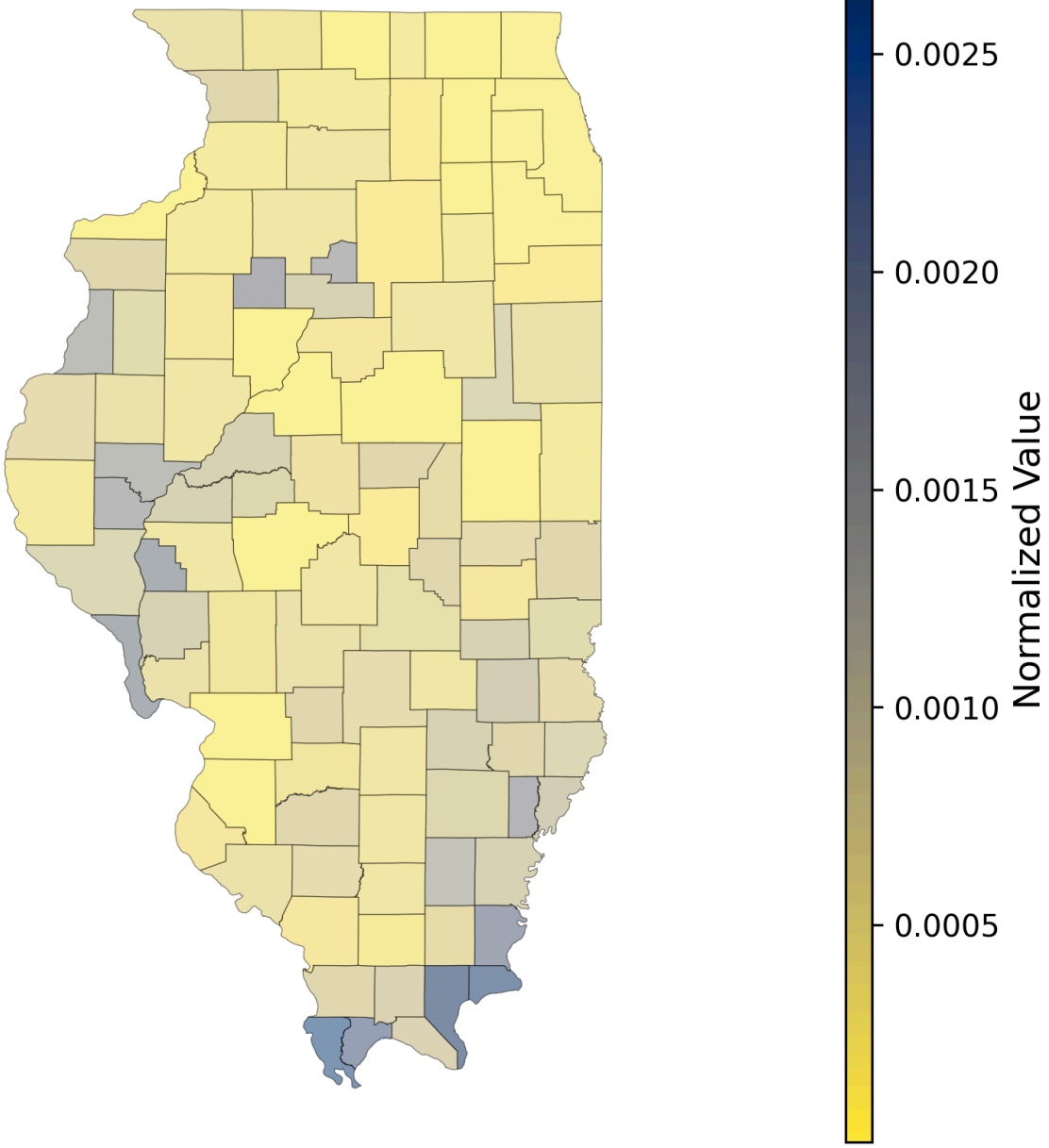


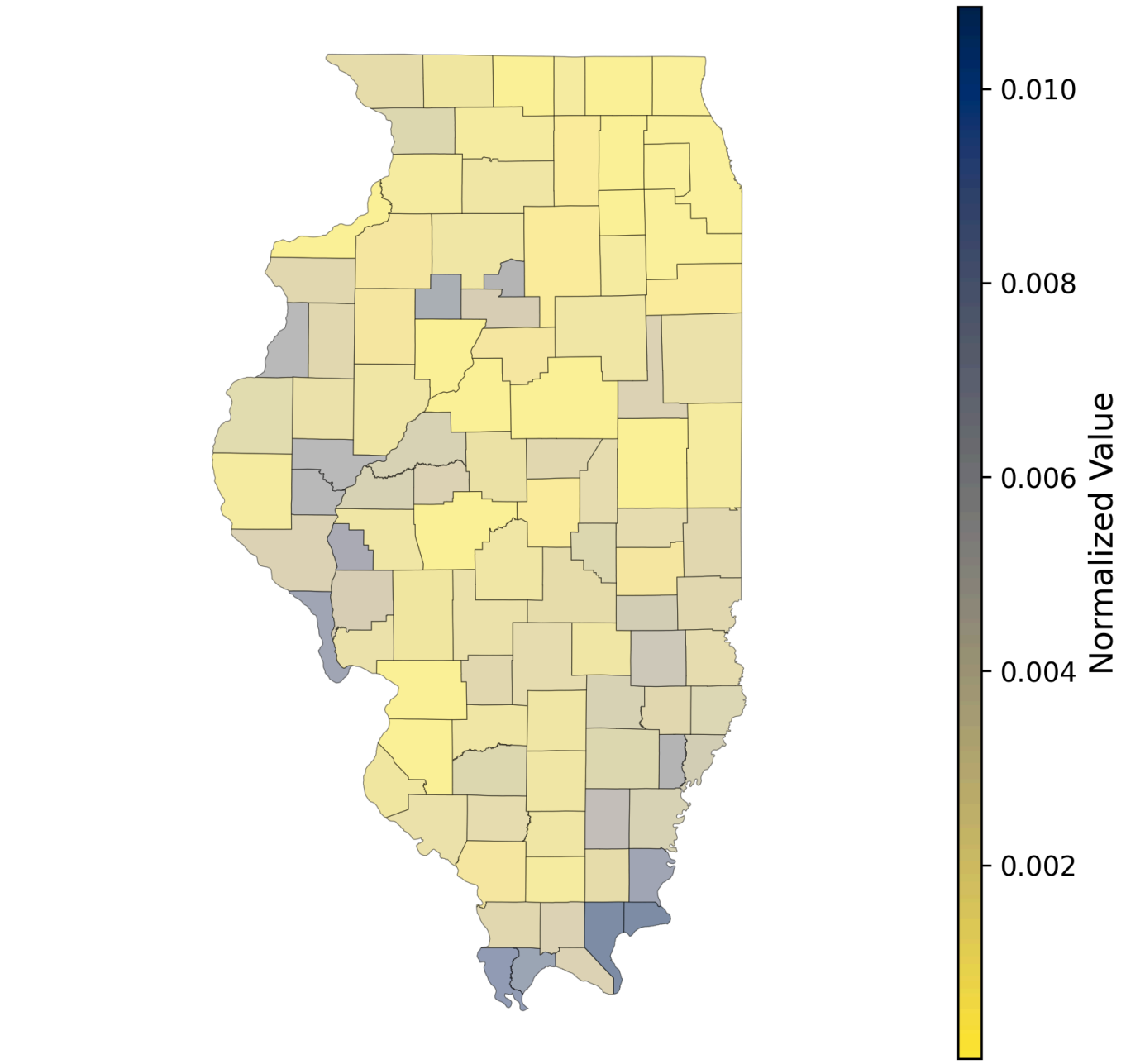
Normalized MOBILITY_CrudePrev - Illinois Map by County




Normalized SELFCARE_CrudePrev - Illinois Map by County
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
Normalized Value

Normalized INDEPLIVE_CrudePrev - Illinois Map by County





US Map at County Level

Import Shapefile

In [ ]:
```
path = "C:/Users/user/Desktop/GIS/Final Project/tl_2023_us_county/tl_2023_us_county.shp"
county_us = gpd.read_file(path)
county_us = county_us.to_crs("EPSG:4326")
```

In [ ]:
```
# Define STATEFP values for non-continental states
non_continental_statefp = ['15', '78', '69', '66', '02', '60', '72']

# Filter out non-continental states from county_us
county_us = county_us[~county_us['STATEFP'].isin(non_continental_statefp)]
```

Preprocess and Merge

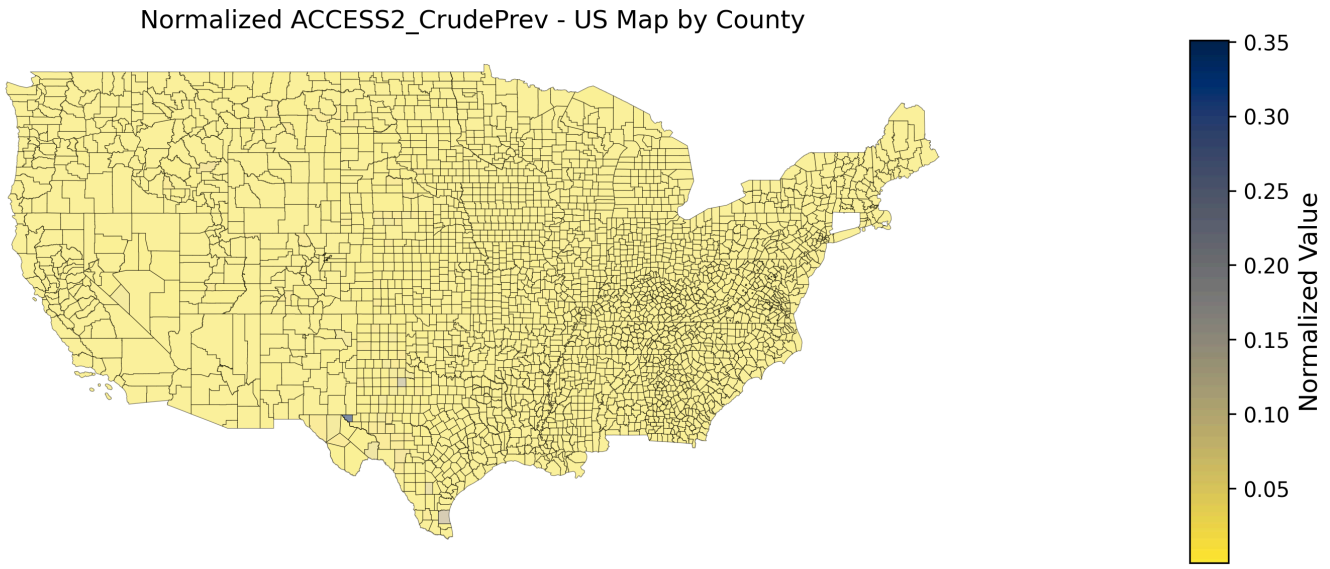
In [ ]:
```
# Add leading zero to CountyFIPS if it has only four digits
df['CountyFIPS'] = df['CountyFIPS'].astype(str)
df['CountyFIPS'] = df['CountyFIPS'].apply(lambda x: x.zfill(5))
```

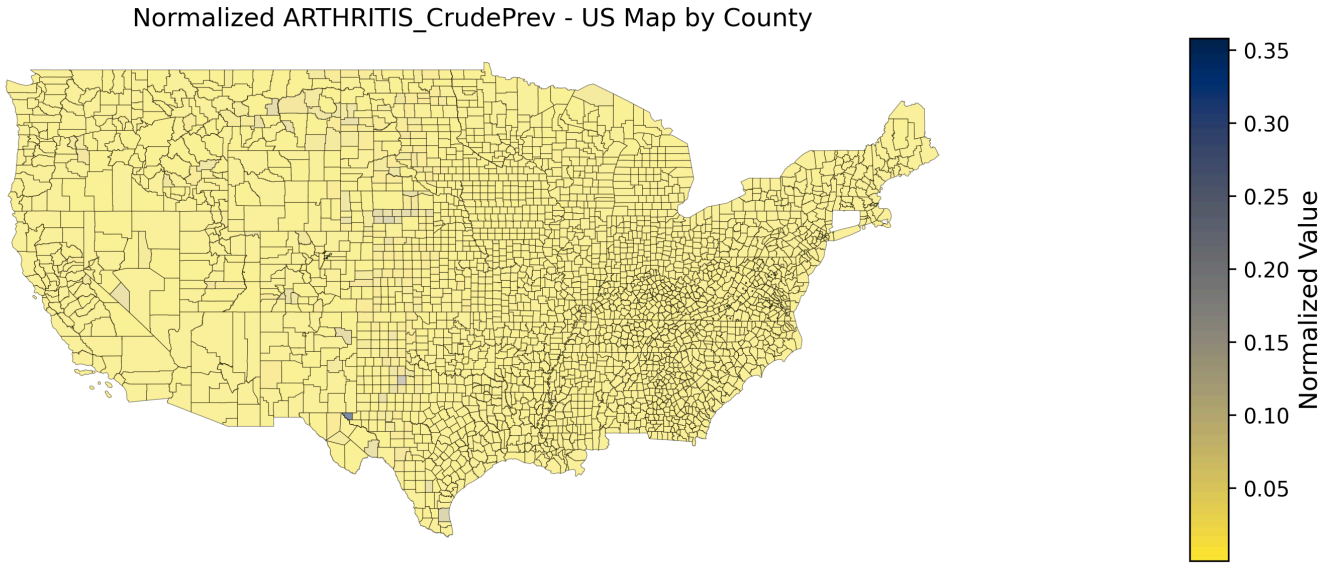
In [ ]:
```
# Merge aggregated DataFrame with the GeoDataFrame
merged_gdf = county_us.merge(df, left_on=['GEOID'], right_on=['CountyFIPS'], how='left')
```

Plotting

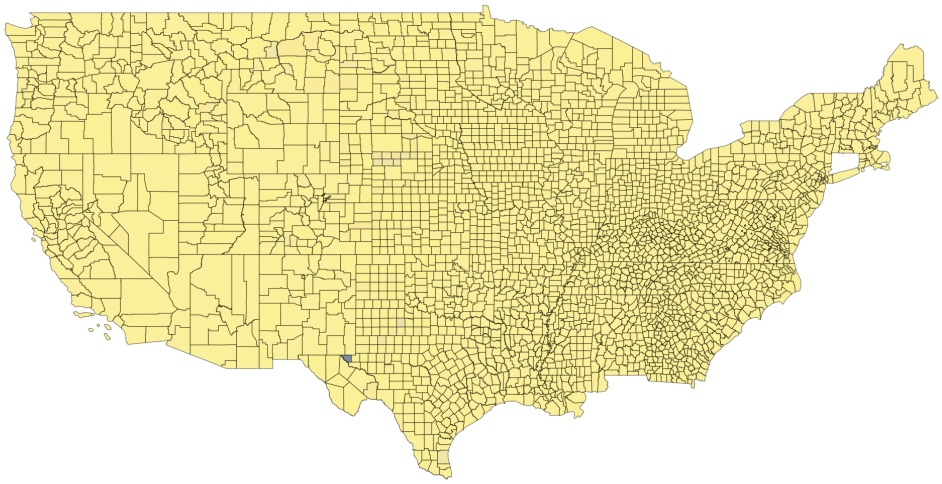
In [ ]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable

def plot_normalized_county_data(geo_df, data_columns, population_column, colormap):
    for column in data_columns:
        # Calculate normalized data by dividing by population
        normalized_column = f'Normalized_{column}'
        geo_df[normalized_column] = geo_df[column] / geo_df[population_column]

        # Plotting
        f, ax = plt.subplots(1, 1, figsize=(10, 6), sharex=True, sharey=True, dpi=300)
        f.tight_layout()
        plt.title(f'Normalized {column} - US Map by County')
        ax.set_axis_off()
        divider = make_axes_locatable(ax)
        cax = divider.append_axes("right", size="3%", pad=0.5, alpha=0.5)
        geo_df.plot(normalized_column, ax=ax, alpha=0.5, cmap=colormap, edgecolor='k', legend=True, cax=cax, linewidth=0.
        plt.ylabel('Normalized Value', fontsize=12)
        plt.show()
```

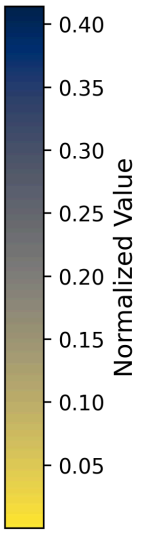
In [ ]:
```
# Get columns ending with '_CrudePrev'
columns_to_plot = [col for col in df.columns if col.endswith('_CrudePrev')]

plot_normalized_county_data(merged_gdf, columns_to_plot, 'TotalPopulation', 'cividis_r')
```



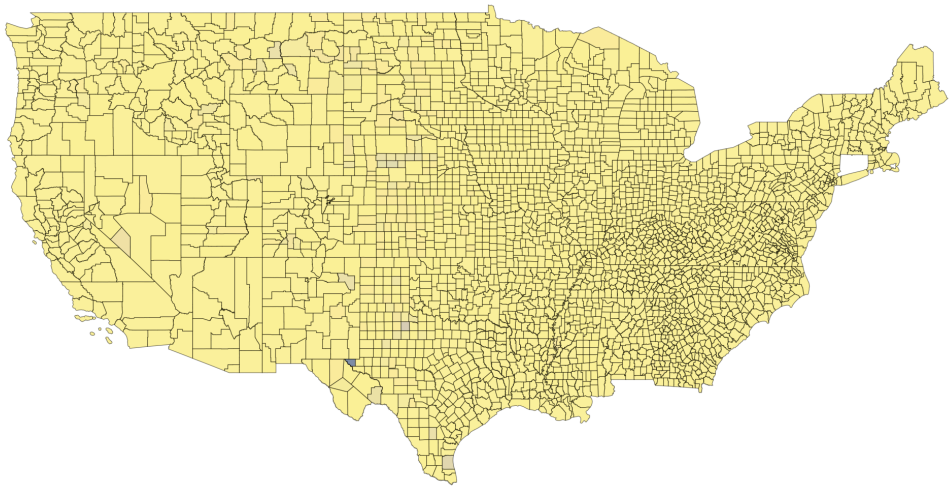


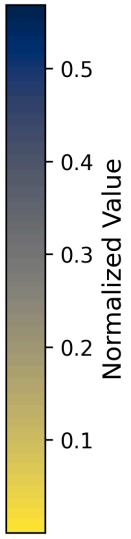
Normalized BINGE_CrudePrev - US Map by County



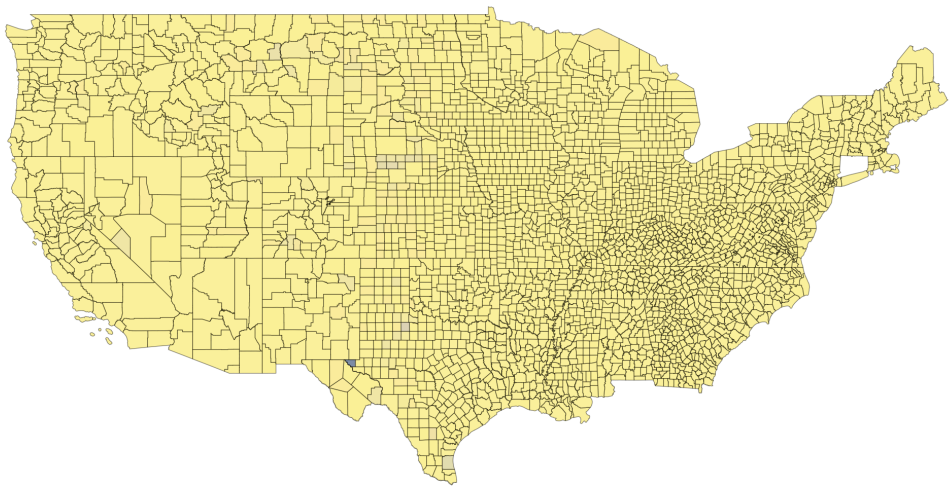


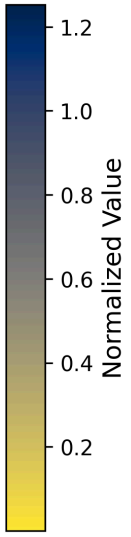
Normalized BPHIGH_CrudePrev - US Map by County



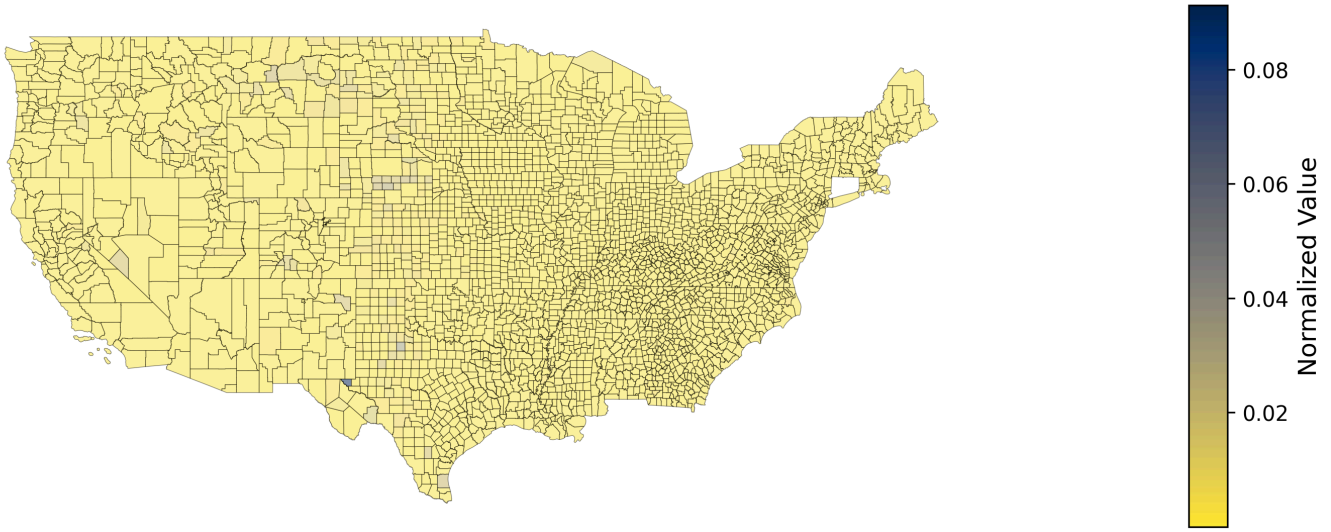


Normalized BPMED_CrudePrev - US Map by County

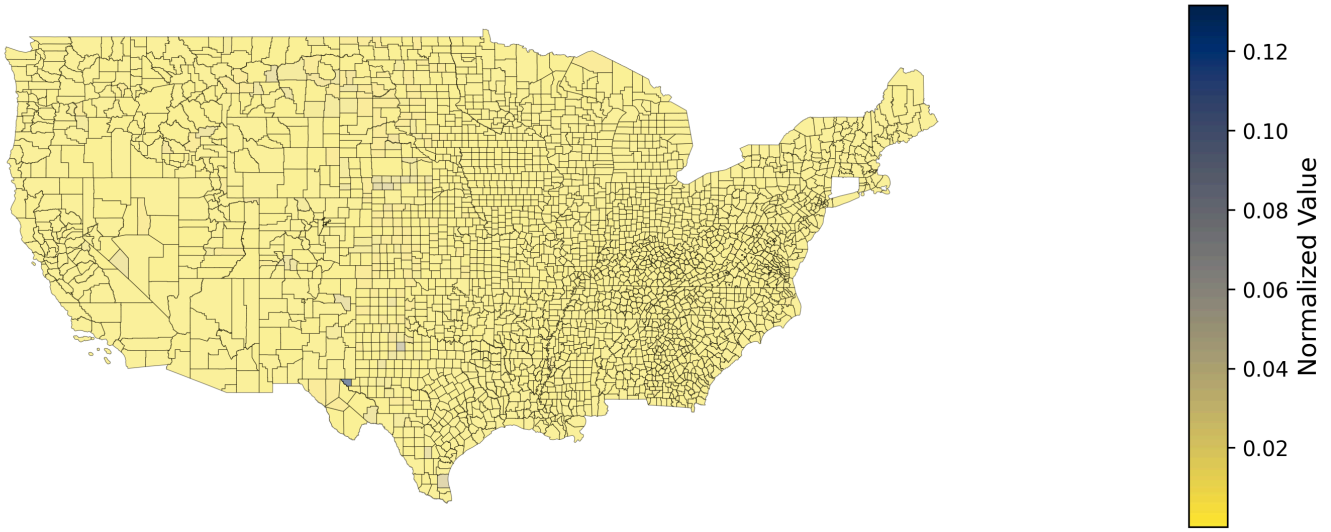




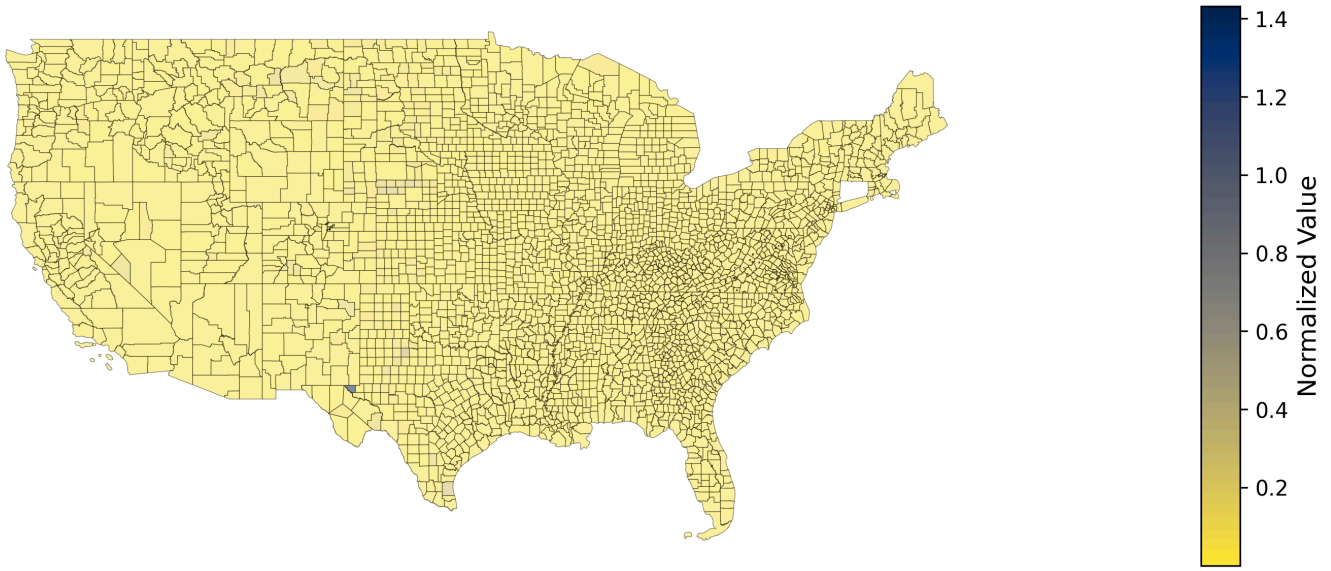
Normalized CANCER_CrudePrev - US Map by County



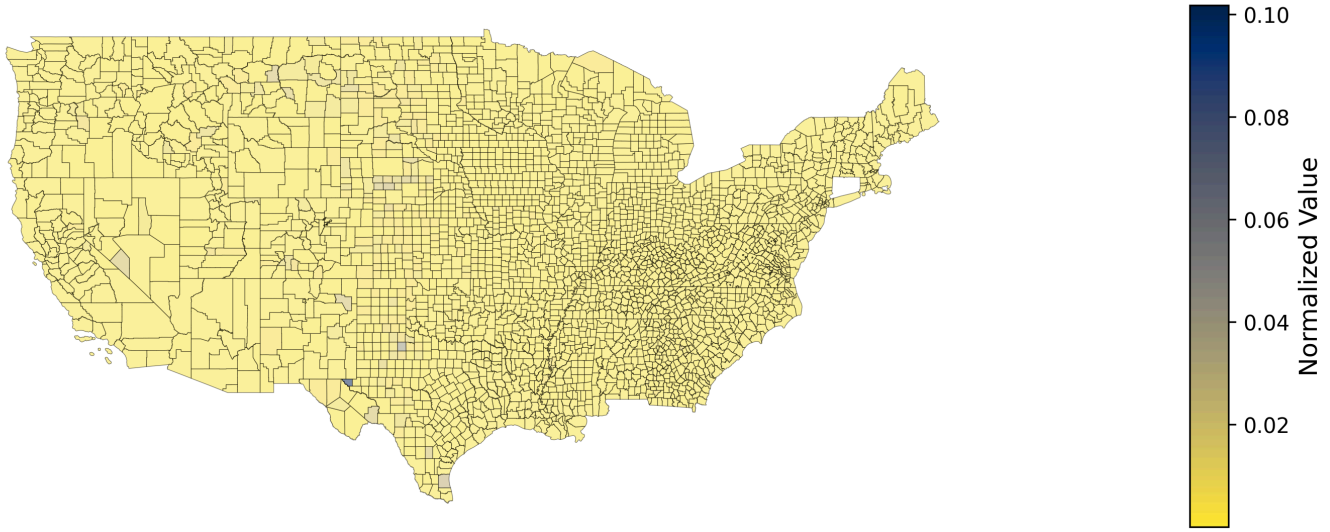
Normalized CASTHMA_CrudePrev - US Map by County



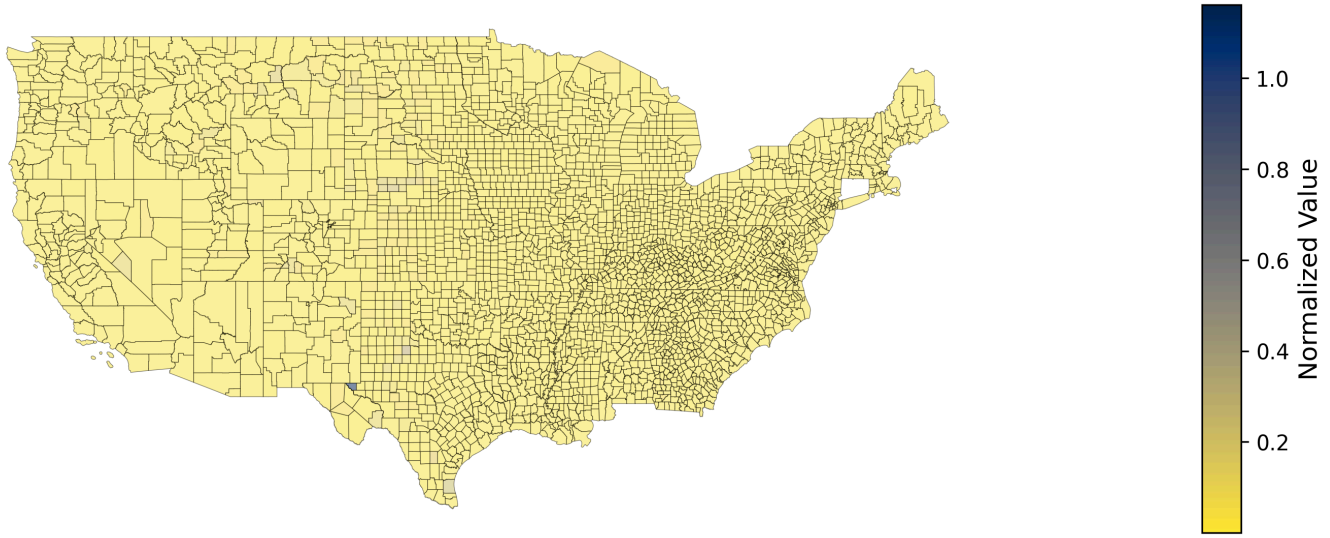
Normalized CERVICAL_CrudePrev - US Map by County



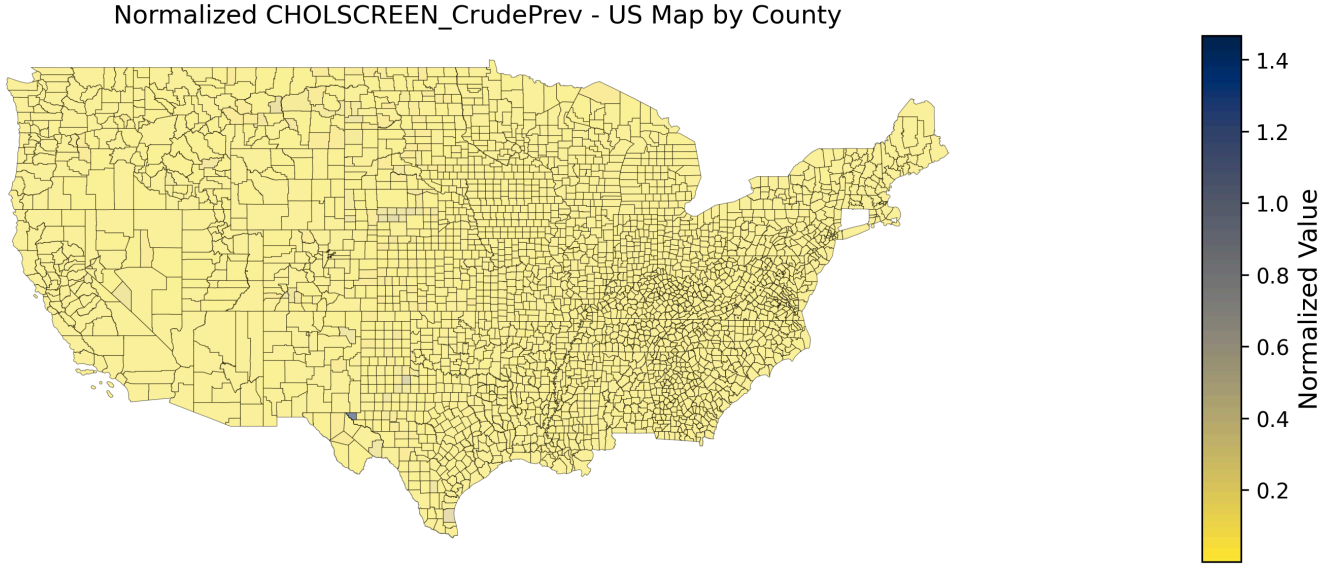
Normalized CHD_CrudePrev - US Map by County



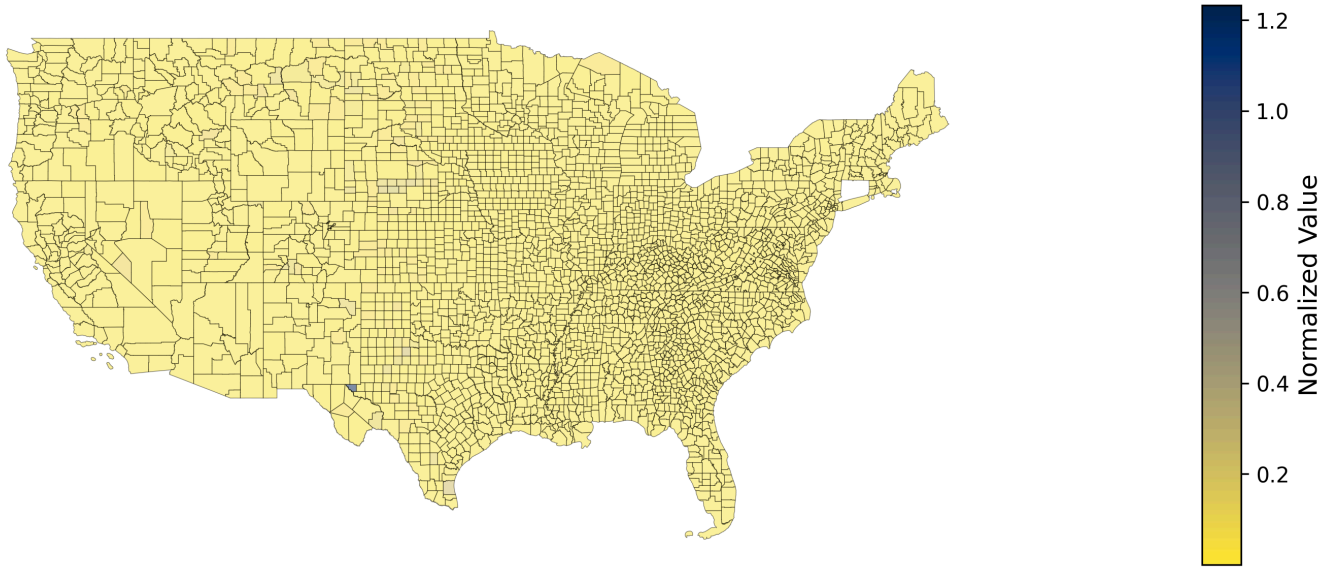
Normalized CHECKUP_CrudePrev - US Map by County



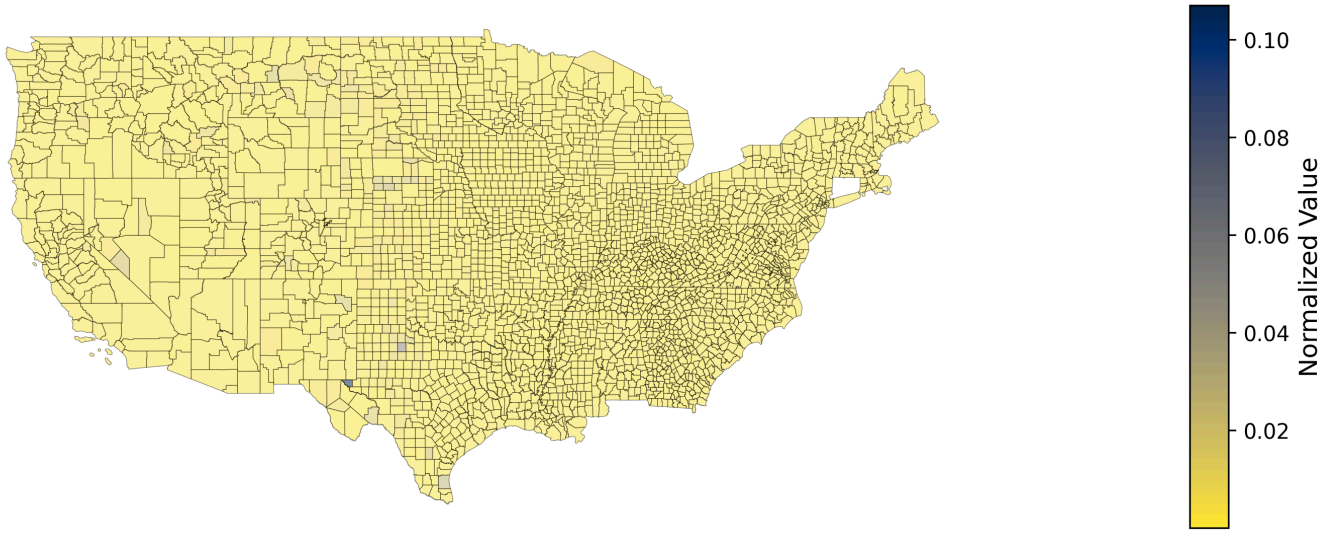
Normalized CHOLSCREEN_CrudePrev - US Map by County



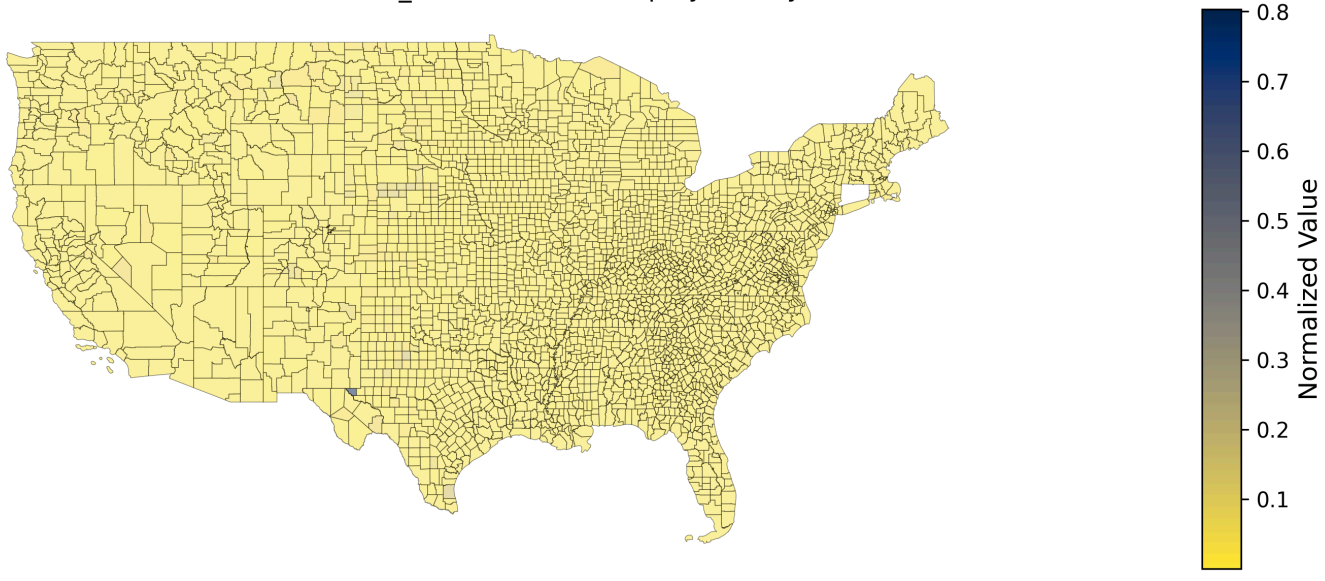
Normalized COLON_SCREEN_CrudePrev - US Map by County



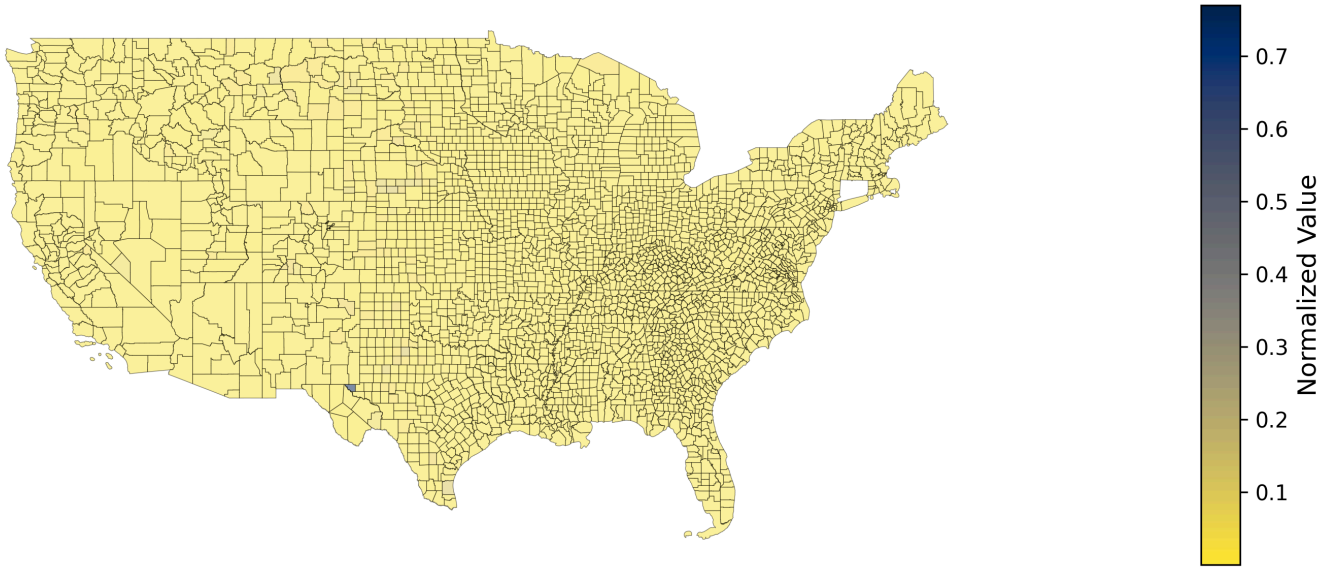
Normalized COPD_CrudePrev - US Map by County



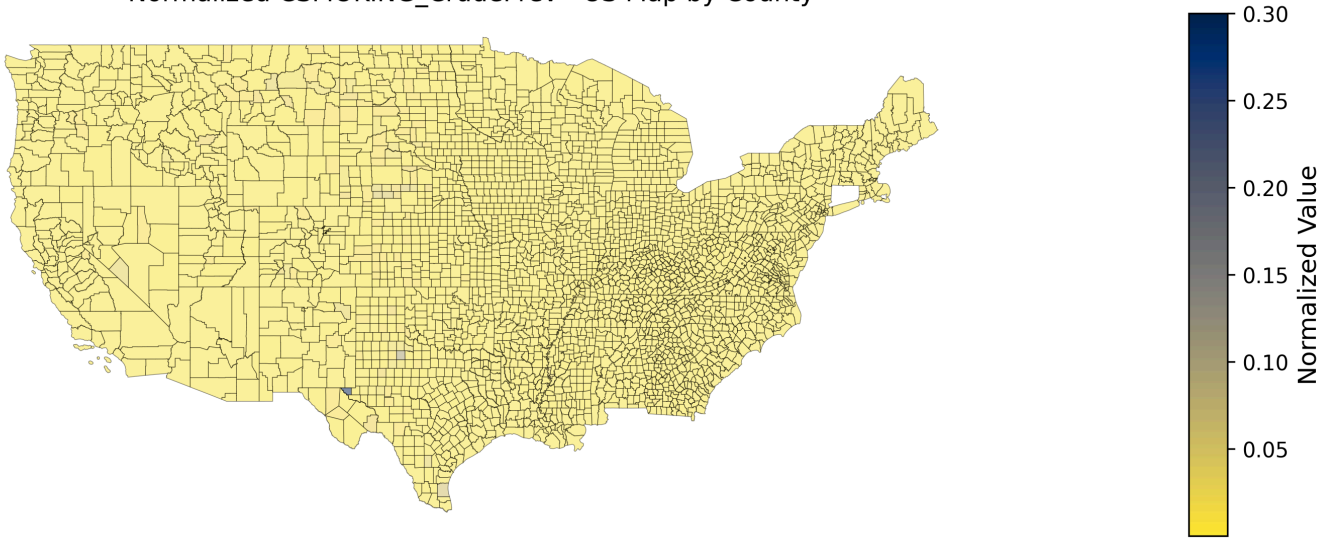
Normalized COREM_CrudePrev - US Map by County



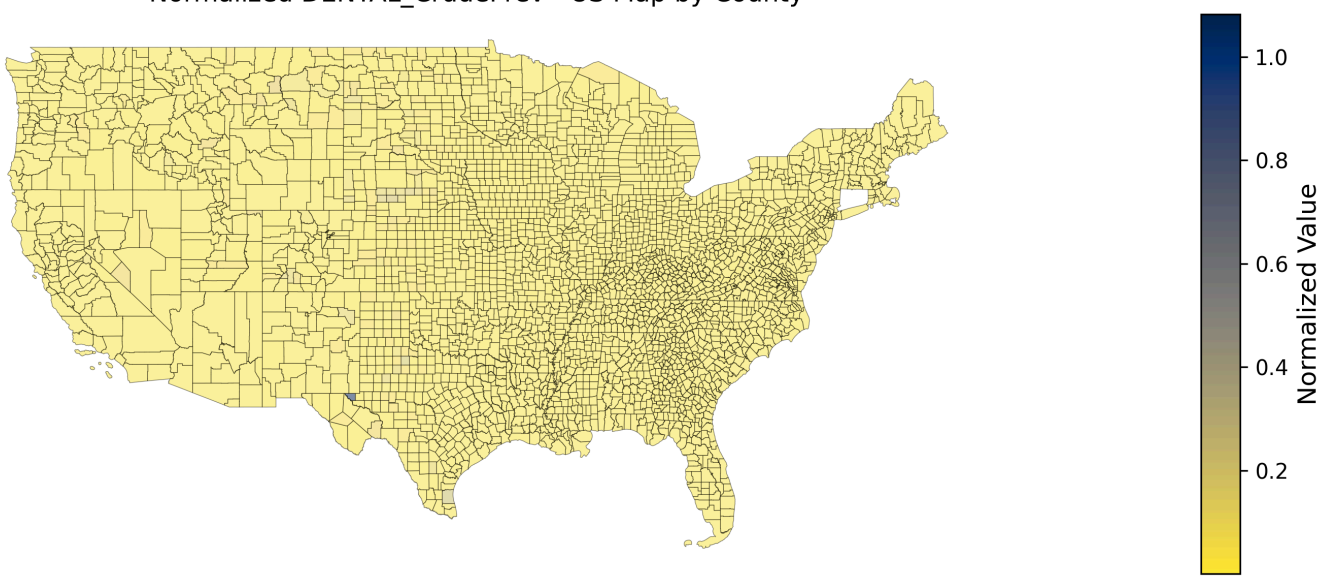
Normalized COREW_CrudePrev - US Map by County



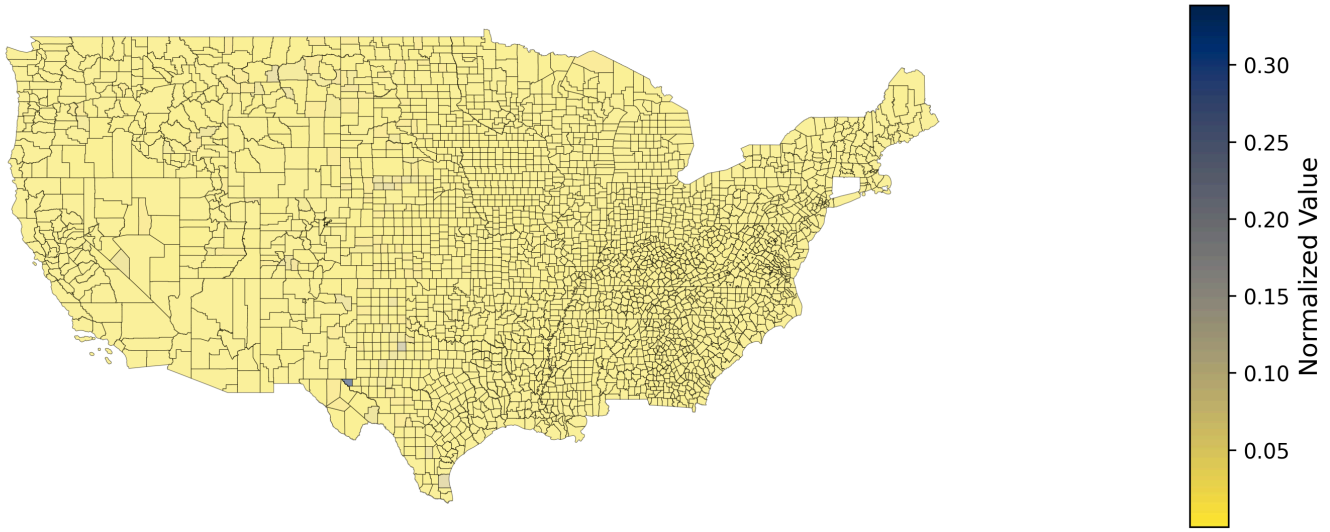
Normalized CSMOKING_CrudePrev - US Map by County



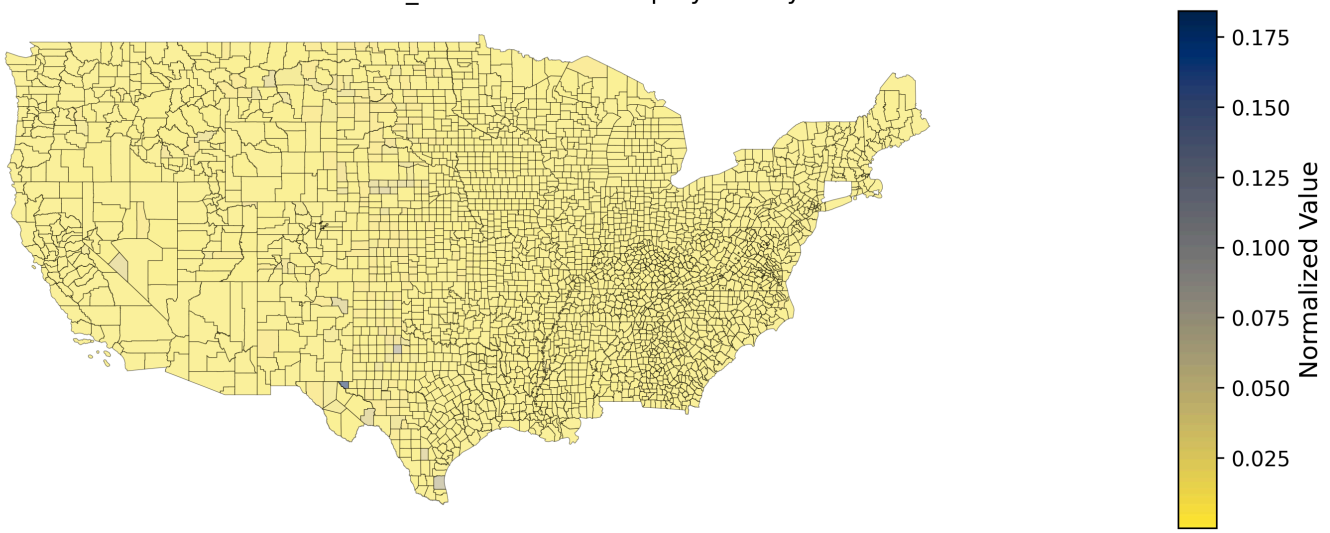
Normalized DENTAL_CrudePrev - US Map by County



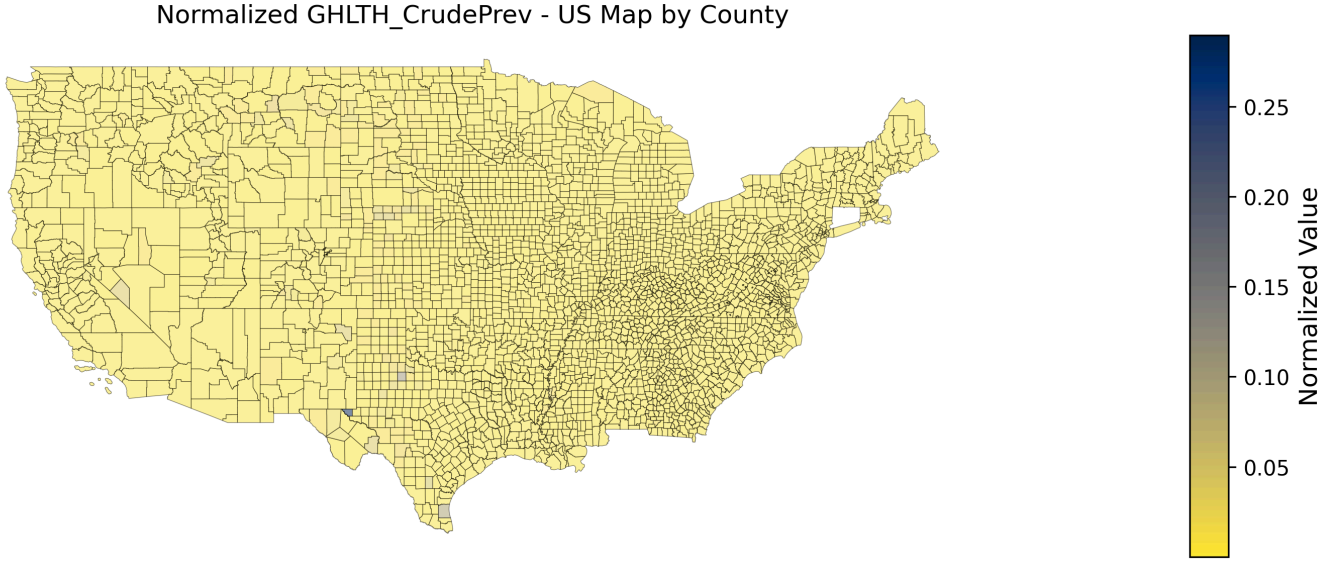
Normalized DEPRESSION_CrudePrev - US Map by County



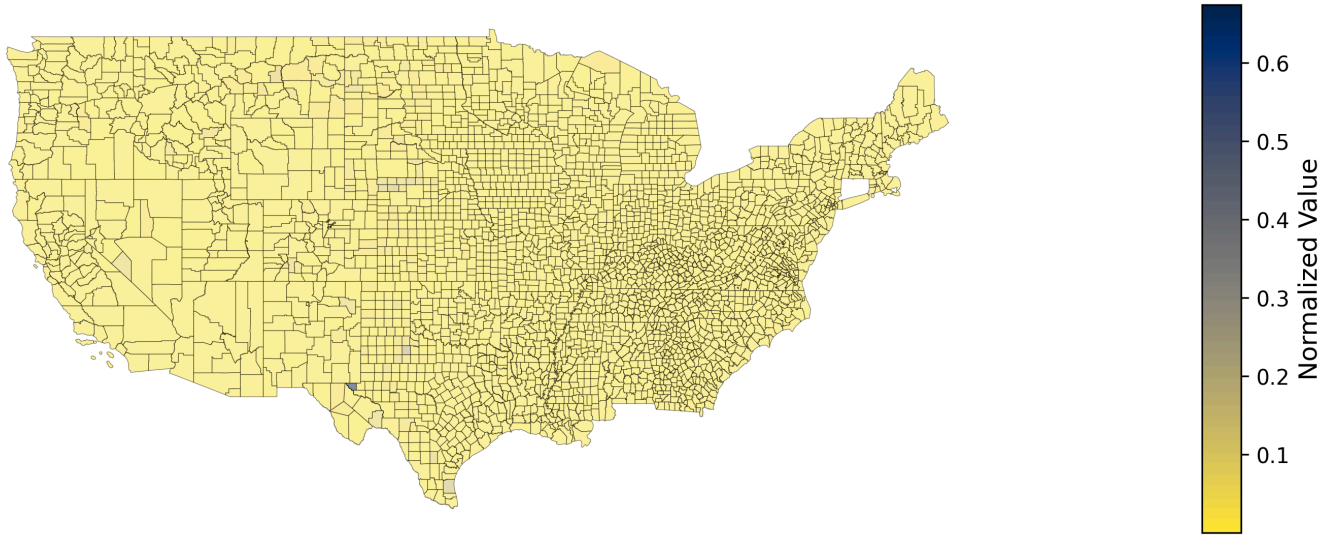
Normalized DIABETES_CrudePrev - US Map by County



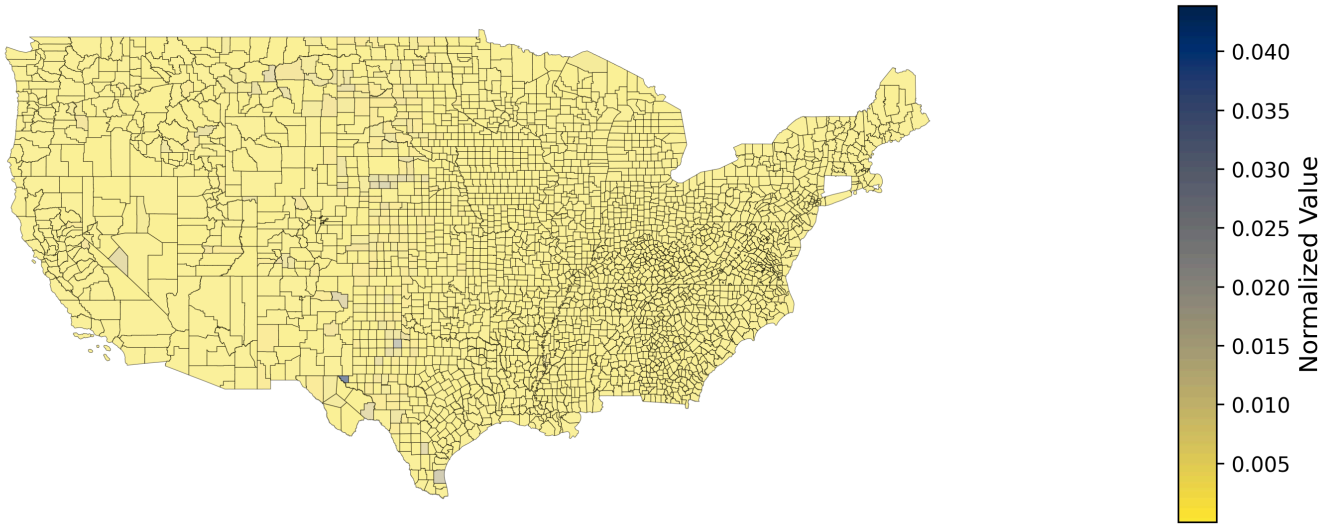
Normalized GHLTH_CrudePrev - US Map by County



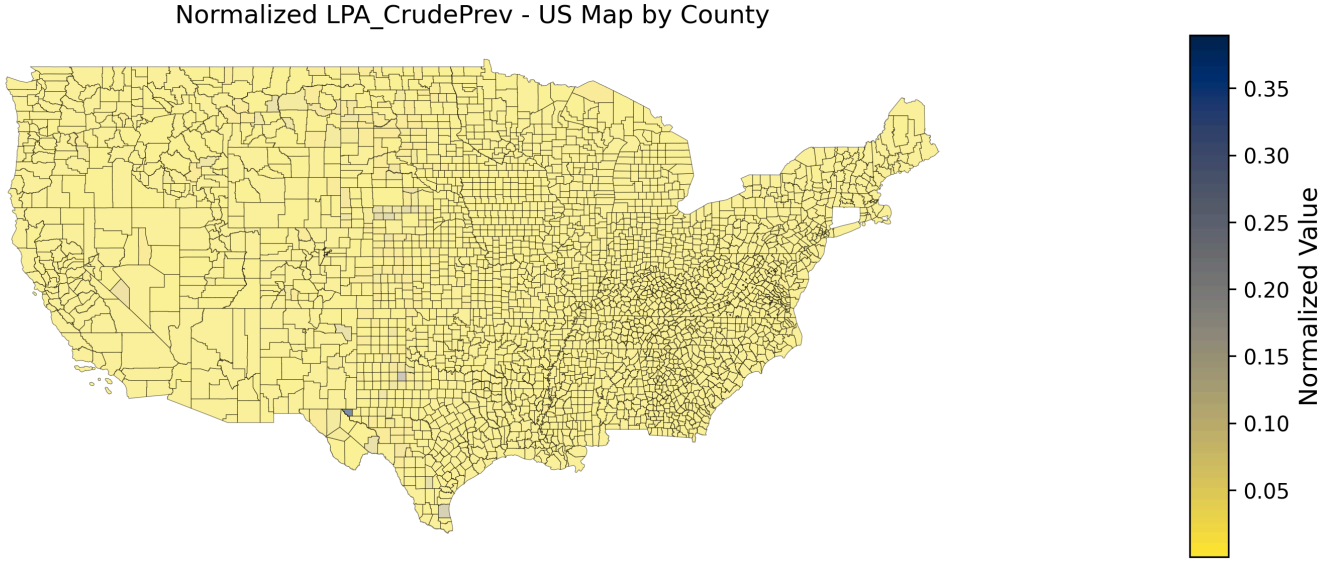
Normalized HIGHCHOL_CrudePrev - US Map by County



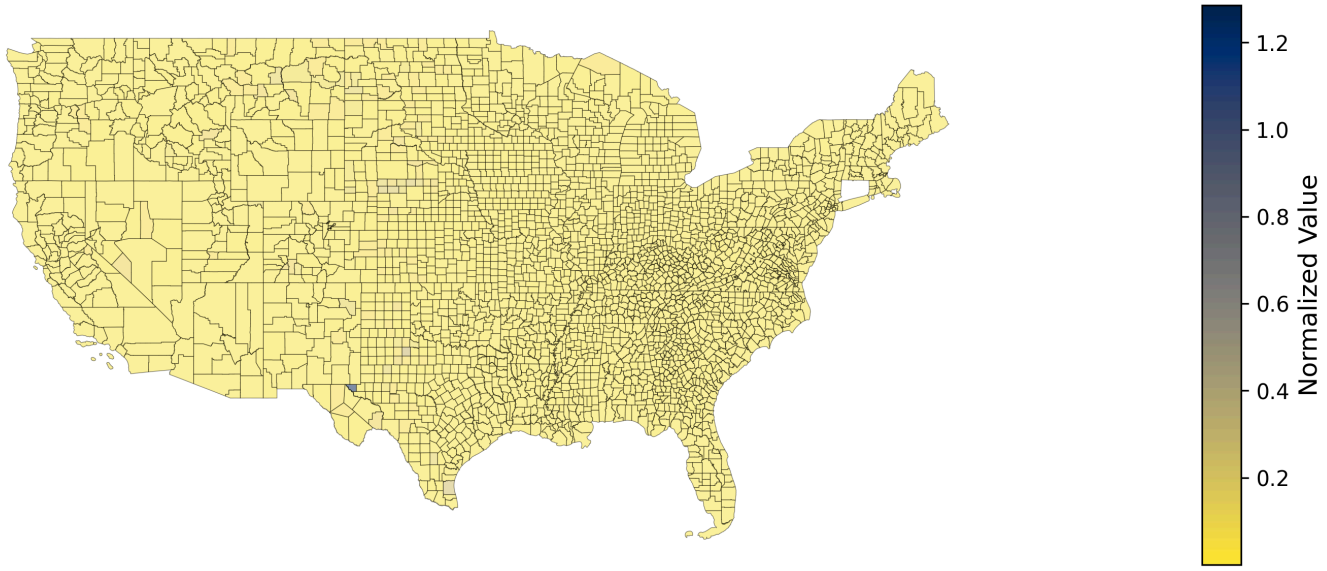
Normalized KIDNEY_CrudePrev - US Map by County



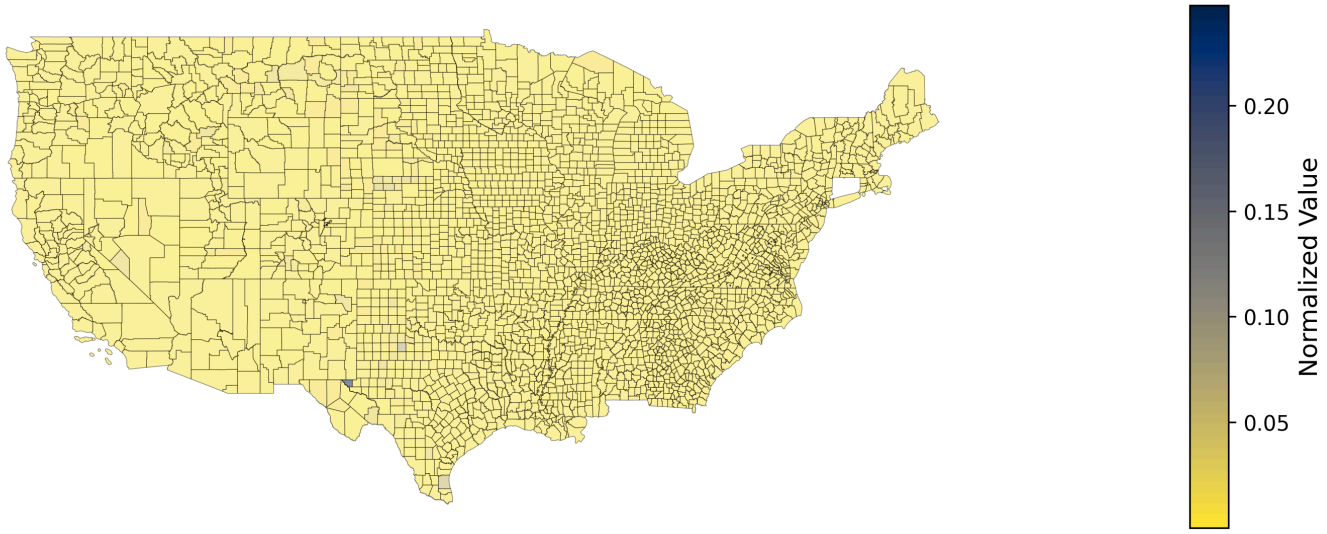
Normalized LPA_CrudePrev - US Map by County



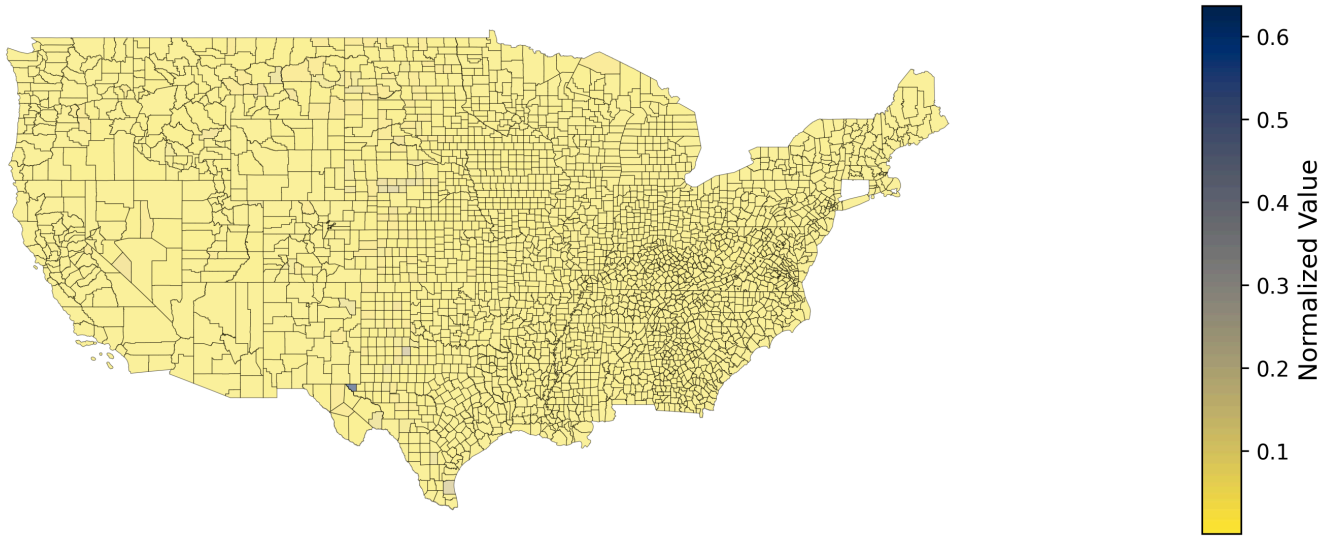
Normalized MAMMOUSE_CrudePrev - US Map by County



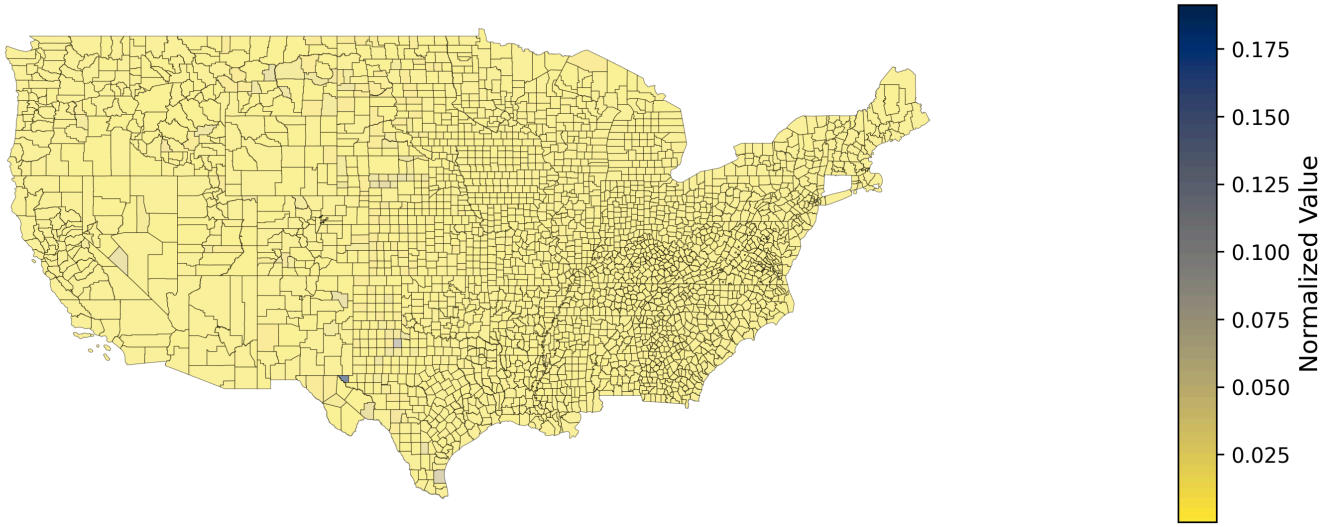
Normalized MHLTH_CrudePrev - US Map by County



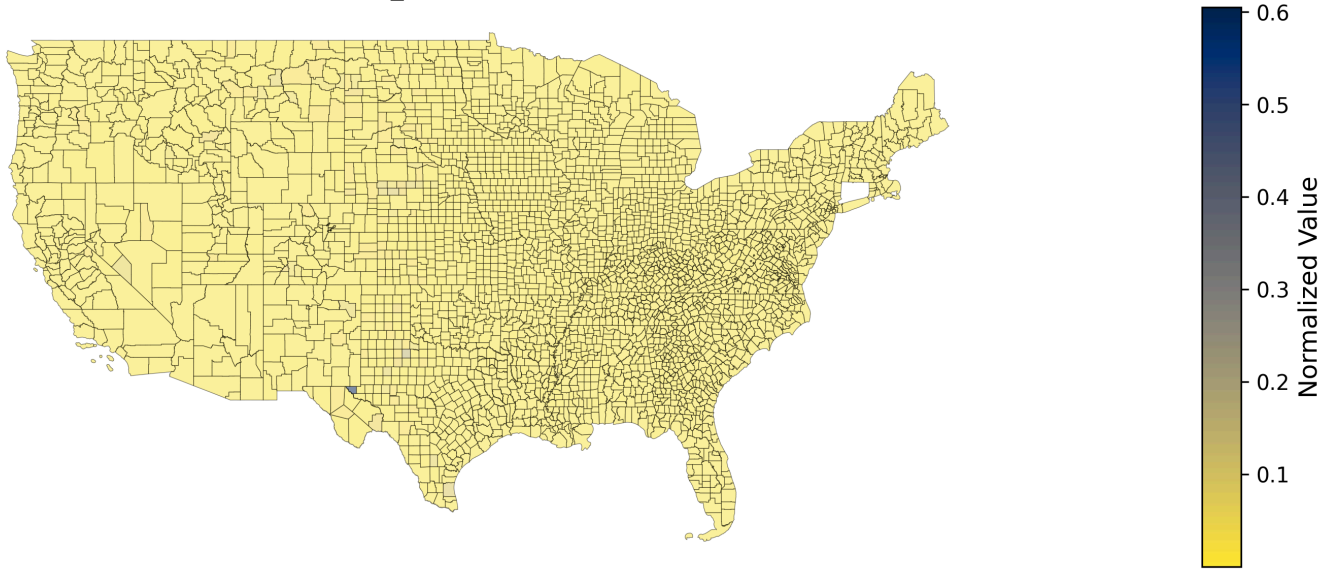
Normalized OBESITY_CrudePrev - US Map by County



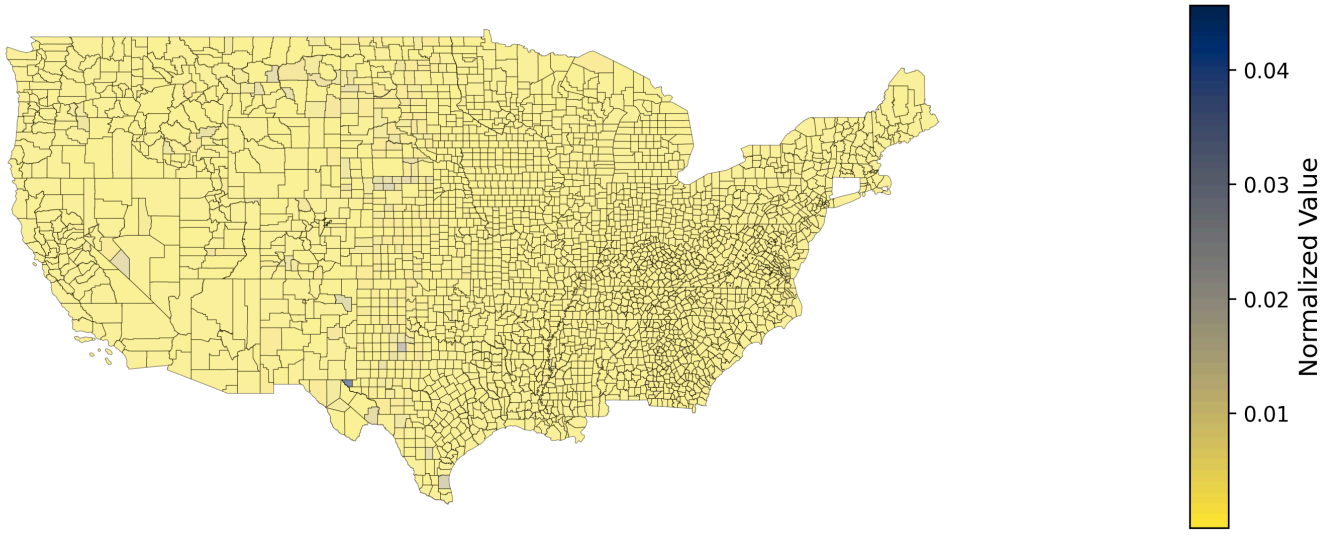
Normalized PHLTH_CrudePrev - US Map by County



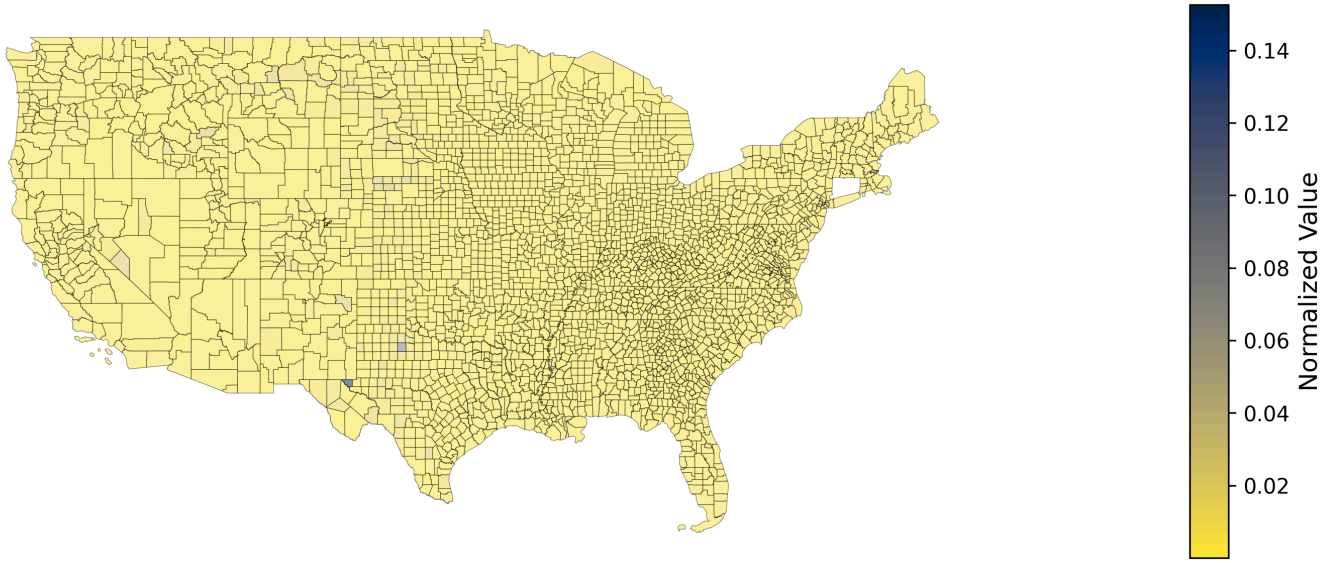
Normalized SLEEP_CrudePrev - US Map by County



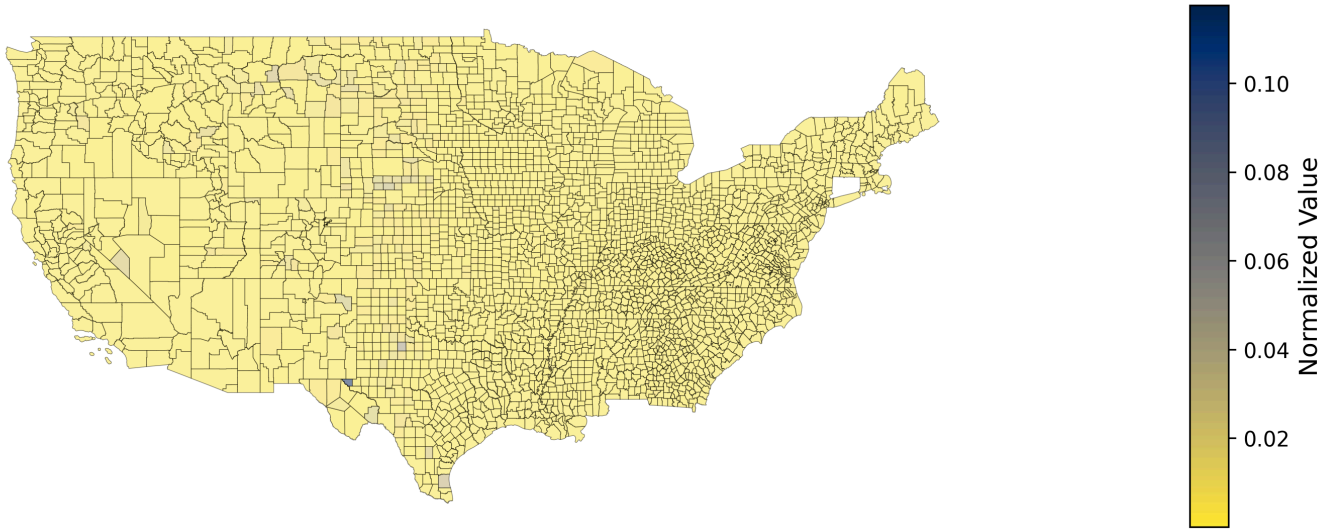
Normalized STROKE_CrudePrev - US Map by County



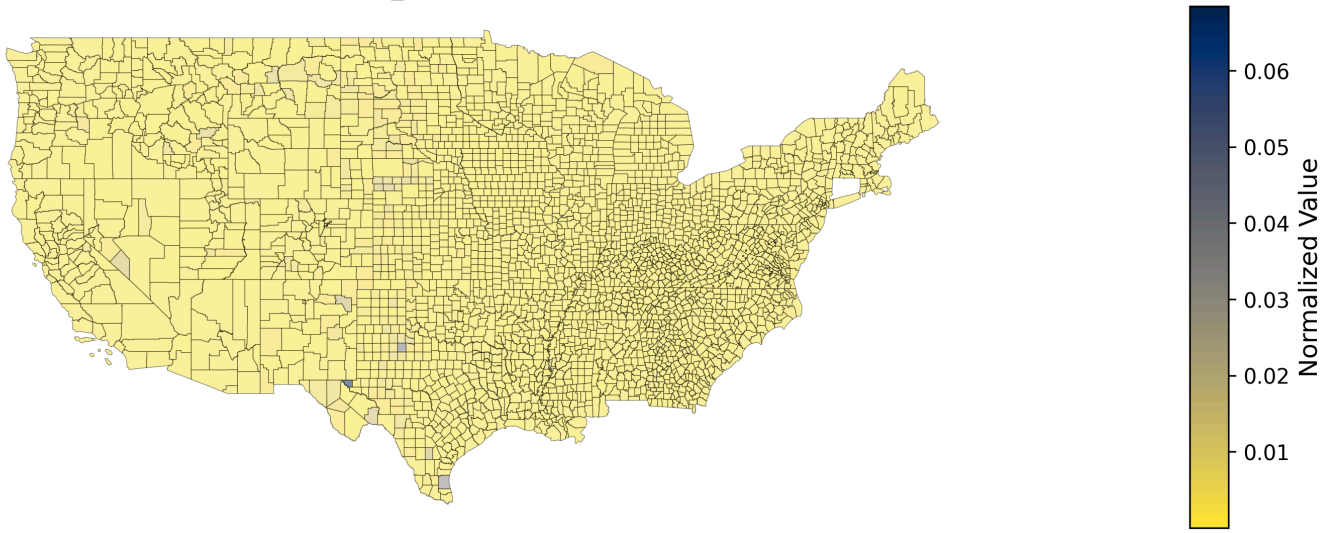
Normalized TEETHLOST_CrudePrev - US Map by County



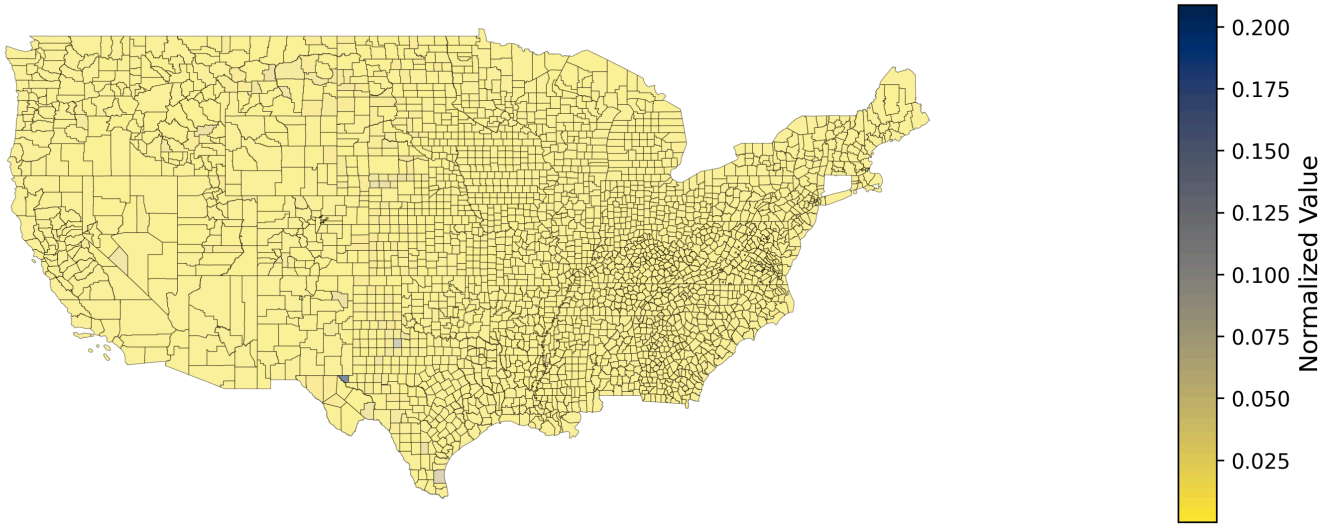
Normalized HEARING_CrudePrev - US Map by County



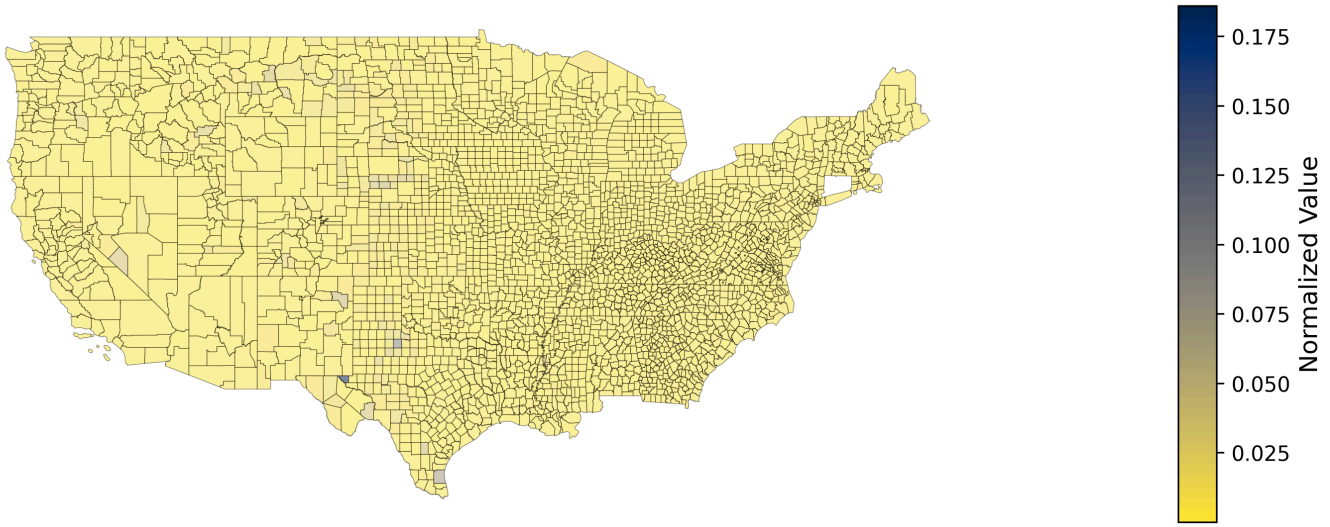
Normalized VISION_CrudePrev - US Map by County



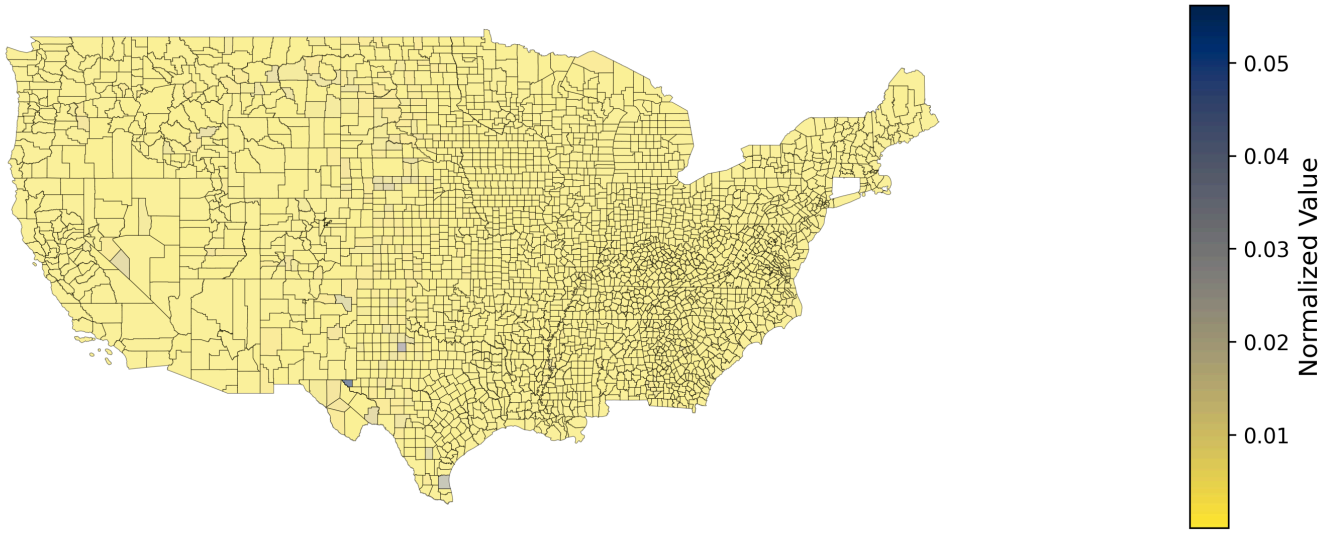
Normalized COGNITION_CrudePrev - US Map by County



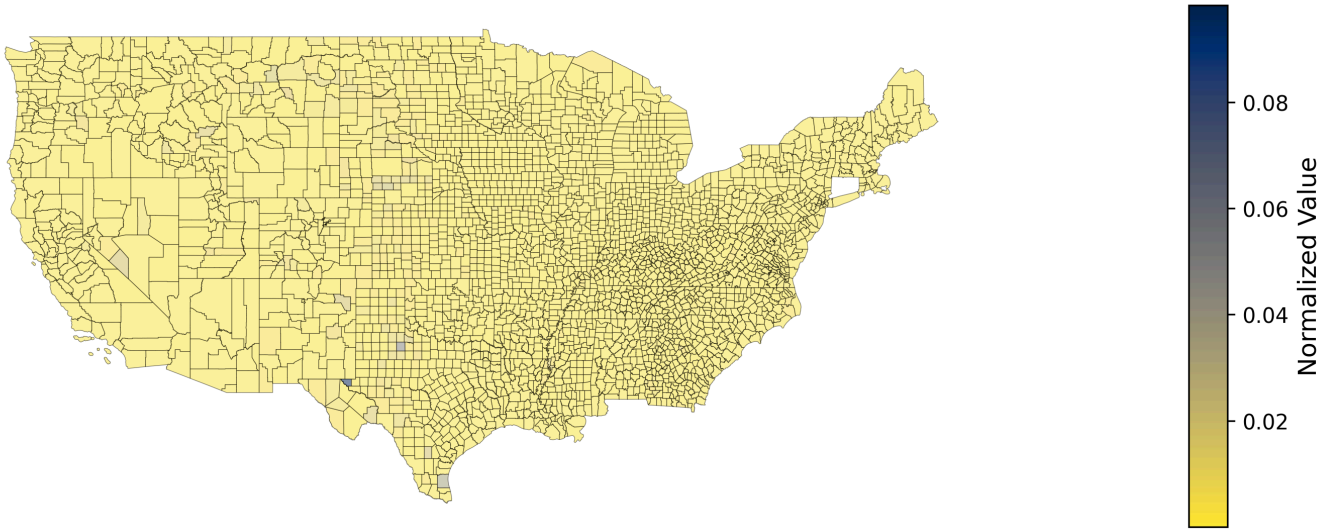
Normalized MOBILITY_CrudePrev - US Map by County



Normalized SELFCARE_CrudePrev - US Map by County



Normalized INDEPLIVE_CrudePrev - US Map by County



Normalized DISABILITY_CrudePrev - US Map by County