Household Hardships and Mental Health Outcomes in the United States

Allyson Totpal

DSC680

January 18, 2026

Business Problem

Mental health conditions, including anxiety and depression, represent a growing public health concern in the United States. Policymakers and public health agencies face the challenge of identifying which economic stressors most strongly relate to changes in mental health outcomes in order to allocate limited resources effectively. Existing national mental health statistics often lack the temporal granularity needed to respond quickly to emerging economic conditions, and many analyses focus on cross-sectional differences rather than changes within states.

Background/History

Mental health outcomes in the United States have long been shaped by economic conditions, with periods of financial instability associated with increased psychological distress, anxiety, and depression. The COVID-19 pandemic intensified these dynamics and in response to the need for more timely public health data, the U.S. Census Bureau launched the Household Pulse Survey (HPS) as a rapid-response instrument to measure social and economic conditions, and provides frequent, state-level estimates that allow researchers and policymakers to observe changes as conditions evolve (United States Census Bureau, 2025). Alongside, structural socioeconomic factors can also shape baseline vulnerability to mental health challenges and influence how populations respond to short-term economic shocks.

Data Explanation

This analysis integrated multiple publicly available datasets to construct a state-by-month panel spanning July 2021 to September 2024. Mental health outcomes and food insecurity measures were obtained from the U.S. Census Bureau’s HPS (2025). The primary outcome variable is the state-level prevalence of adults reporting symptoms of anxiety or depressive disorder. Food insecurity was measured using state-level estimates of recent difficulty affording food, serving as a proxy for household material hardship.

Economic indicators were sourced from the U.S. Bureau of Labor Statistics (2025), including monthly state unemployment rates and national year-over-year inflation derived from the Consumer Price index (U.S. Bureau of Labor Statistics, 2025). Structural socioeconomic characteristics were obtained from the American Community Survey (United States Census Bureau, 2025), including median household income, housing cost burden, and health insurance coverage. Because ACS estimates are reported annually, values were expanded to monthly observations and the most recent available year was carried forward to cover the full study period.

Data preparation involved aggregating weekly Pulse survey estimates to monthly averages, aligning sources temporally at the monthly level, and standardizing state identifiers across datasets. Observations were restricted to states with complete data across key variables to ensure internal consistency. The resulting dataset enables analysis of within state changes in economic conditions and mental health outcomes over time while accounting for both short-term and longer-term structural context.

Data Dictionary

Field name	Description	Source	Type
`state`	U.S. state name	Household Pulse Survey	string
`month`	Month (first day of month timestamp)	Derived	date
`Value`	Prevalence of adults reporting symptoms of anxiety or depressive disorder	Household Pulse Survey	numeric
`food_insecurity_proxy`	Proxy measure for food hardship	Household Pulse	numeric

		Survey	
`unemployment_rate`	State unemployment rate	BLS (LAUS via API)	numeric
`inflation_yoy`	Year-over-year inflation rate (CPI-based)	BLS CPI	numeric
`median_household_income`	Median household income	ACS 1-year	numeric
`pct_housing_cost_burden`	Percent of households with housing cost burden	ACS 1-year	numeric
`pct_uninsured`	Percent uninsured	ACS 1-year	numeric
`food_insecurity_proxy_lag1`	Prior month food insecurity proxy	Derived	numeric
`unemployment_rate_lag1`	Prior month unemployment rate	Derived	numeric
`inflation_yoy_lag1`	Prior month inflation YoY	Derived	numeric

Methods

The primary analysis employed linear regression models estimated on a state-by-month panel dataset. A pooled regression model was first used to establish baseline associations between economic stressors and mental health outcomes. To account for unobserved, time-invariant differences across states, fixed-effects models were estimated with state indicators. Extended specifications incorporated month fixed effects to control for national shocks affecting all states simultaneously. To assess temporal dynamics, lagged versions of key economic predictors were estimated in a fixed-effects framework, evaluating whether prior-month conditions were associated with current mental health outcomes. Structural socioeconomic variables from the ACS were introduced in robustness models to assess whether estimated associations persisted after accounting for baseline economic context. All models used standard errors clustered at the state level to account for within-state correlation over time.

Analysis

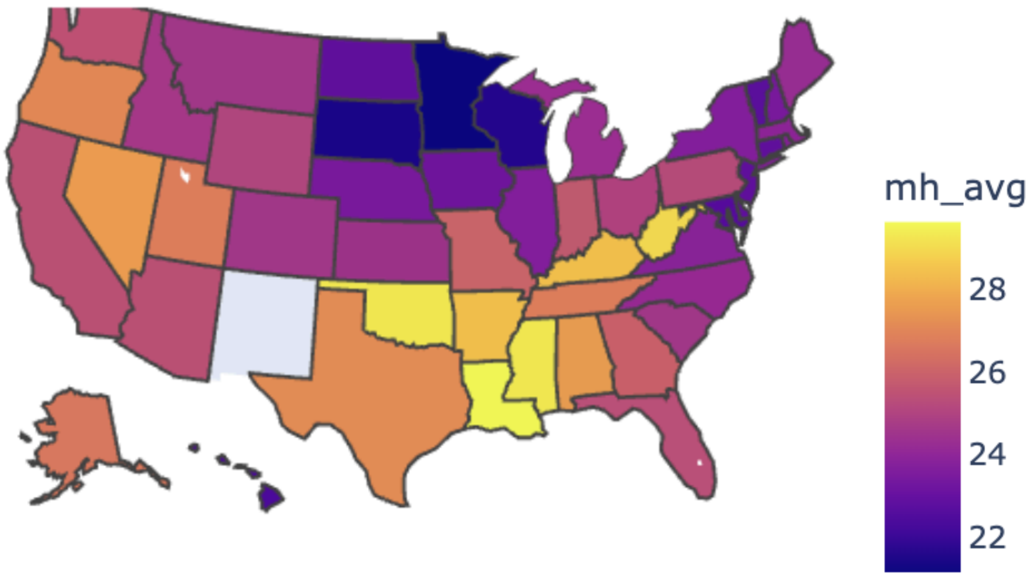

Figure 1: Choropleth Map of Mental Health Average (time period 2021 to 2024)

State-level averages reveal substantial geographic variation in mental health prevalence (Figure 1), supporting the use of fixed-effects models to focus on within-state changes over time rather than cross-sectional differences. Regression results consistently identify food insecurity as the economic stressor most strongly associated with mental health outcomes. In two-way fixed-effects models controlling for state and month effects, increases in food insecurity are associated with statistically significant increases in the prevalence of anxiety and depressive symptoms.

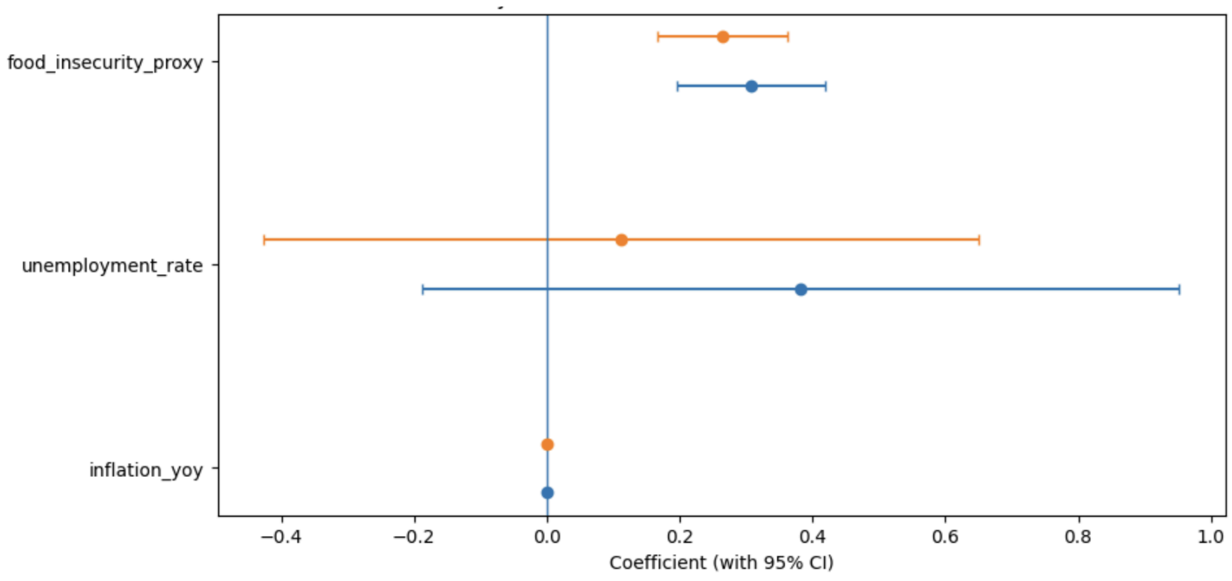

Figure 2: Key Coefficients: Model 2B vs Model 2B + ACS

This relationship remains stable after incorporating structural socioeconomic controls from the ACS (Figure 2). In contrast, unemployment and inflation do not exhibit robust independent associations once food insecurity and fixed effects are included.

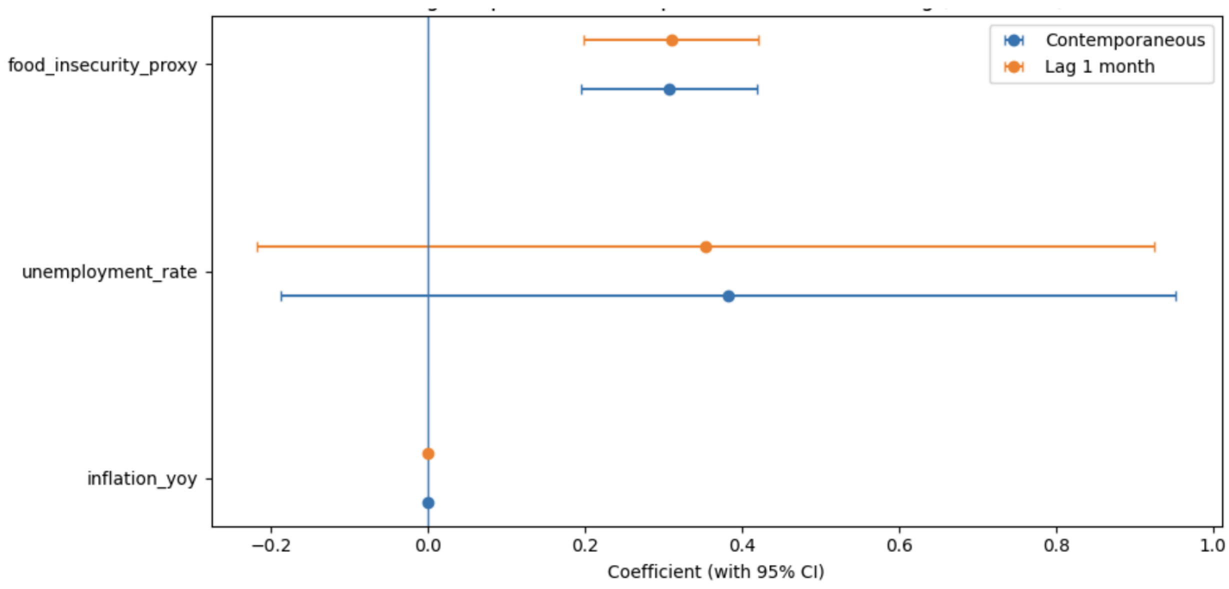

Figure 3: Timing Comparison: Contemporaneous vs 1-Month Lag (FE Models)

Lagged models further demonstrate that the association between food insecurity and mental health persists when food insecurity is measured in the prior month, suggesting that the psychological impacts of material hardship may accumulate over time rather than dissipate immediately (Figure 3).

Conclusion

This analysis finds that food insecurity is the economic condition most consistently associated with changes in state-level mental health outcomes over time. Across multiple model specifications, increases in food insecurity are linked to higher prevalence of anxiety and depressive symptoms, even after accounting for labor market conditions, inflation, and structural socioeconomic context. These findings suggest that household-level material hardship may serve as a more proximate indicator of mental health risk than traditional macroeconomic measures.

Assumptions, Limitations, & Challenges

This analysis assumes that state-level estimates from the HPS reasonably reflect underlying trends in population mental health and economic hardship. Aggregating weekly survey data to the monthly level is assumed to reduce short-term noise while preserving meaningful time variation. The fixed-effects modeling framework further assumes that unobserved state characteristics influencing mental health are time-invariant.

Several limitations should be noted. Due to data availability and suppression in the HPS, the analytic sample is restricted to states with complete data across key variables and may not be nationally representative. The observational design of the study precludes causal, and estimated associations should be interpreted as correlational. In addition, structural socioeconomic variables from the ACS are reported annually and may not fully capture short-term changes in economic conditions.

Key challenges included integrating multiple data sources with differing time measurements, aligning weekly, monthly, and annual measures, and managing missing or suppressed state-level estimates.

Future Uses/Additional Applications

The analytic framework developed in this project can be extended to support ongoing public health monitoring and policy evaluation. Future work could examine subgroup-level patterns or finer geographic units to better identify populations most vulnerable to the mental health impacts of economic hardship. The approach may also be used to assess the effectiveness of food assistance of economic relief programs by evaluating changes in food insecurity and mental health outcomes before and after policy implementation. More probably, integrating food insecurity and mental health indicators into routine surveillance systems could support early identification of emerging risks and enable more timely, targeted public health responses.

Recommendations

Based on the findings, public health agencies should prioritize food insecurity as a key indicator of population mental health risk. Because food insecurity consistently demonstrates a strong and robust association with anxiety and depressive symptoms, interventions that improve food access, such as strengthening nutrition assistance programs, expanding eligibility during periods of economic stress, and reducing administrative barriers, may also yield mental health benefits. Policymakers should consider integrating food insecurity metrics into routine mental health surveillance and early warning systems to identify emerging risks more quickly. Targeting resources toward households experiencing material hardship may be more effective than relying solely on broad macroeconomic indicators such as unemployment and inflation.

Implementation Plan

Implementation would involve incorporating state-level food insecurity indicators into existing public health monitoring workflows alongside mental health surveillance data. Agencies could establish a monthly reporting cadence that tracks changes in food insecurity and mental health prevalence concurrently, enabling early identification of concerning trends. Data pipelines could leverage existing CDC, Census Bureau, and BLS sources, minimizing additional data collection burden. Findings should be disseminated through concise dashboards or briefing reports to inform decision-maker and guide targeted deployment of food assistance and mental health support services. Over time, this framework could be used to evaluate whether policy interventions aimed at reducing food insecurity correspond with improvements in population mental health outcomes.

Ethical Assessment

This analysis relies exclusively on publicly available, aggregated data and does not involve individual-level or personally identifiable information. All data sources used adhere to establish ethical standards for data collection and dissemination.

Care was taken to avoid stigmatizing interpretations of mental health outcomes or economic hardship. Results are presented at the state level and framed as associations rather than causal claims, reducing the risk of misrepresentation or inappropriate policy conclusions. The analysis emphasizes structural and economic conditions rather than individual behaviors, aligning with ethical best practices in public health research.

Finally, ethical considerations extend to responsible communication of findings. While food insecurity is identified as a key correlate of mental health outcomes, recommendations focus on expanding support and access to resources rather than assigning blame. This approach supports evidence-based decision-making while promoting equity, transparency, and respect for affected populations.

References

United States Census Bureau 1. (2025). Household Pulse Survey: Measuring Emergent Social and Economic Matters Facing U.S. Households. *United States Census Bureau.* https://www.census.gov/data/experimental-data-products/household-pulse-survey.html

United States Census Bureau 2. (2025). Household Pulse Survey Public Use File (PUF). *United States Census Bureau.* https://www.census.gov/programs-surveys/household-pulse-survey/data/datasets.html

United States Census Bureau 3. (2025). American Community Survey 1-year estimates. *United States Census Bureau.* https://www.census.gov/programs-surveys/acs

U.S. Bureau of Labor Statistics 1. (2025). Local Area Unemployment Statistics. *U.S. Bureau of Labor Statistics.* https://www.bls.gov/lau/

U.S. Bureau of Labor Statistics 2. (2025). Consumer Price Index. *U.S. Bureau of Labor Statistics.* https://www.bls.gov/cpi/

https://pmc.ncbi.nlm.nih.gov/articles/PMC10893396/

Appendix

```
import pandas as pd
import requests
import time

# read in mental health data
mh =
pd.read_csv('/Users/smooshii/DSC680/Indicators_of_Anxiety_or_Depression_Based_on_Repor
ted_Frequency_of_Symptoms_During_Last_7_Days.csv')
mh.head()

mh['start_date'] = pd.to_datetime(mh['Time Period Start Date'])
mh['WEEK'] = mh['Time Period']

# restrict to phase 3.2+
mh = mh[mh['start_date'] >= '2021-07-01'].copy()

# create month for later aggregation
mh['month'] = mh['start_date'].dt.to_period('M').dt.to_timestamp()

# build list of (year, week) pairs needed
mh['year'] = mh['start_date'].dt.year.astype(str)
weeks_needed = mh[['year', 'WEEK']].drop_duplicates()

def fetch_foodscarce(year: str, week) -> pd.DataFrame:
    '''Fetches state-level food scarcity rates from the U.S. Census HPS API for
specified year and survey week.

    Parameters:
        year (str) - Survey year to query
        week (int or str) - HPS week number

    Returns:
        DataFrame containing state-level food scarcity rates and collection date
metadata or None if the request fails'''
    url = 'https://api.census.gov/data/timeseries/hps'
    params = {
        'get': 'NAME,FOODSCARCE_RATE,COL_START_DATE,COL_END_DATE',
        'for': 'state:*',
        'time': year,
        'WEEK': str(week),
    }
    r = requests.get(url, params = params, timeout = 60)

    if r.status_code == 204:
        return None
    if r.status_code != 200:
        print('ERROR', r.status_code, r.url, r.text[:200])
        return None

    data = r.json()
    # first row contains column name; remaining rows contain data
    out = pd.DataFrame(data[1:], columns = data[0])
    out['year'] = year
    out['WEEK'] = str(week)
    # convert food scarcity rate to numeric, coercing invalid values to NaN
    out['FOODSCARCE_RATE'] = pd.to_numeric(out['FOODSCARCE_RATE'], errors = 'coerce')
    # convert collection date fields to datetime objects
    out['COL_START_DATE'] = pd.to_datetime(out['COL_START_DATE'], errors = 'coerce')
    out['COL_END_DATE'] = pd.to_datetime(out['COL_END_DATE'], errors = 'coerce')
    return out

# pull all needed weeks
```

```
food_frames = []
for year in [2021, 2022, 2023, 2024]:
    for week in range(1, 90):
        df_week = fetch_foodscarce(year, week)
        if df_week is not None:
            food_frames.append(df_week)
        time.sleep(0.15)

food = pd.concat(food_frames, ignore_index = True)

# rename to project variable name
food = food.rename(columns = {'NAME': 'State',
                              'FOODSCARCE_RATE': 'food_insecurity_proxy'})

# use collection start date to create month
food['month'] = food['COL_START_DATE'].dt.to_period('M').dt.to_timestamp()

# aggregate weekly -> monthly (state-month)
food_state_month = (food.groupby(['State', 'month'], as_index =
False)['food_insecurity_proxy'].mean())

food_state_month.head()

# FIPs codes required to construct valid BLS time series IDs
state_fips = {
    "Alabama": "01", "Alaska": "02", "Arizona": "04", "Arkansas": "05",
    "California": "06", "Colorado": "08", "Connecticut": "09",
    "Delaware": "10", "District of Columbia": "11",
    "Florida": "12", "Georgia": "13", "Hawaii": "15",
    "Idaho": "16", "Illinois": "17", "Indiana": "18",
    "Iowa": "19", "Kansas": "20", "Kentucky": "21",
    "Louisiana": "22", "Maine": "23", "Maryland": "24",
    "Massachusetts": "25", "Michigan": "26", "Minnesota": "27",
    "Mississippi": "28", "Missouri": "29", "Montana": "30",
    "Nebraska": "31", "Nevada": "32", "New Hampshire": "33",
    "New Jersey": "34", "New Mexico": "35", "New York": "36",
    "North Carolina": "37", "North Dakota": "38", "Ohio": "39",
    "Oklahoma": "40", "Oregon": "41", "Pennsylvania": "42",
    "Rhode Island": "44", "South Carolina": "45", "South Dakota": "46",
    "Tennessee": "47", "Texas": "48", "Utah": "49",
    "Vermont": "50", "Virginia": "51", "Washington": "53",
    "West Virginia": "54", "Wisconsin": "55", "Wyoming": "56"
}

series_ids = {state: f'LASST{fips}' + '0'*10 + '003'
              for state, fips in state_fips.items()
}

list(series_ids.items())[:5]

# api pull for bls unemployment data
bls_url = 'https://api.bls.gov/publicAPI/v2/timeseries/data/'

def fetch_bls(series_ids, startyear = '2021', endyear = '2024', api_key = None):
    '''Fetch unemployment rate time series data from U.S. Bureau of Labor Statistics
API for multiple states.

    Parameters:
        series_id (dict) - mapping of state names to BLS series IDs
        startyear (str) - first year of data to retrieve
        endyear (str) - last year of data to retrieve
        api_key (str) - BLS API registration key
    Returns:
        dict - parsed JSON response containing time series data'''
    payload  = {'seriesid': list(series_ids.values()),
                'startyear': startyear,
                'endyear': endyear}
    if api_key:
        payload['registrationkey'] = api_key
    r = requests.post(bls_url, json = payload, timeout = 60)
    r.raise_for_status()
    return r.json()

data = fetch_bls(series_ids)

rows = []
# loop over each state-level time series
for series in data['Results']['series']:
    sid = series['seriesID']
    # identify state corresponding to current series ID
    state = [k for k, v in series_ids.items() if v == sid][0]

    # loop over individual monthly observations within the series
    for obs in series['data']:
        period = obs['period']
        if not period.startswith('M') or period == 'M13':
            continue

        year = int(obs['year'])
        month = int(period[1:])
        # construct standardized datetime object (first day of month)
        date = pd.to_datetime(f'{year}-{month:02d}-01')

        rows.append({'State': state,
                     'month': date,
                     'unemployment_rate': float(obs['value'])})

unemp = pd.DataFrame(rows)

unemp.columns

# ensure date range and sort values
unemp = unemp[(unemp['month'] >= '2021-07-01') & (unemp['month'] <= '2024-09-01')]
unemp.sort_values(['State', 'month']).head()

# call bls for cpi data
payload = {'CPI_U_All_Items': 'CUSR0000SA0'}

list(series_ids.items())

def fetch_bls_cpi(series_ids, startyear = '2020', endyear = '2024', api_key = None):
    '''Retrieves CPI time series data from U.S. Bureau of Labor Statistics API

    Parameters:
        series_id (dict) - mapping of series labels to BLS CPI series IDs
        startyear (str) - first year of data to retrieve
        endyear (str) - last year of data to retrieve
        api_key (str) - BLS API registration key

    Returns:
    dict - parsed JSON response containing CPI time series data'''
    payload = {'seriesid': list(series_ids.values()),
               'startyear': startyear,
               'endyear': endyear}
    if api_key:
        payload['registrationkey'] = api_key

    r = requests.post(bls_url, json = payload, timeout = 60)
    r.raise_for_status()
    return r.json()

data = fetch_bls_cpi(series_ids)

rows = []
# loop over each CPI series returned by API
for series in data['Results']['series']:
    sid = series['seriesID']
    # identify state corresponding to current series ID
    series_name = [k for k, v in series_ids.items() if v == sid][0]

    # loop through individual observations within series
    for obs in series['data']:
        period = obs['period']
        # keep only standard monthly observations
        if not period.startswith('M') or period == 'M13':
            continue

        year = int(obs['year'])
        month = int(period[1:])
        # contruct standarized datetime object (first day of month)
        date = pd.to_datetime(f'{year}-{month:02d}-01')

        rows.append({'month': date,
                     'cpi_index': float(obs['value'])})

cpi = pd.DataFrame(rows).sort_values('month').reset_index(drop = True)

cpi.columns

# comput YoY inflation
cpi['inflation_yoy'] = (cpi['cpi_index'] / cpi['cpi_index'].shift(12) - 1) * 100

# filter analysis window
cpi = cpi[(cpi['month'] >= '2021-07-01') & (cpi['month'] <= '2024-09-01')]
cpi.head()

# pull acs for one year
def fetch_acs_detailed(year, var_list):
    '''Retrieves state-level ACS 1-year detailed estimates for specified year and list
of variables

    Parameters:
        year (int or str) - ACS survey year to query
        var_list (list) - list of ACS variable codes to retrieve

    Returns:
        DataFrame containing state-level ACS estimates with year metadata appended'''
    base = f'https://api.census.gov/data/{year}/acs/acs1'
    params = {'get': 'NAME,' + ','.join(var_list),
              'for': 'state:*'}

    r = requests.get(base, params = params, timeout = 60)
    r.raise_for_status()

    data = r.json()
    df = pd.DataFrame(data[1:], columns = data[0])
    df['year'] = year
    return df
```

```
acs_detailed_vars = [
    "B19013_001E",
    "B25070_001E",
    "B25070_007E",
    "B25070_008E",
    "B25070_009E",
    "B25070_010E",
]

# subject tables endpoint for percent uninsured
def fetch_acs_subject(year, var_list):
    '''Fetches state-level ACS 1-year subject table estimates for specified year and
list of variables

    Parameters:
        year (int or str) - ACS survey year to query
        var_list (list) - list of ACS variable codes to retrieve

    Returns:
        DataFrame containing state-level ACS subject table estimates with year
metadata appended'''
    base  = f'https://api.census.gov/data/{year}/acs/acs1/subject'
    params = {'get': 'NAME,' + ','.join(var_list),
              'for': 'state:*'}

    r = requests.get(base, params = params, timeout = 60)
    r.raise_for_status()
    data = r.json()
    df = pd.DataFrame(data[1:], columns = data[0])
    df['year'] = year
    return df

acs_subject_vars = ['S2701_C05_051E']

# pull all years
# no 2024 data at this time, using 2023 for 2024 data
acs_frames = []
for yr in [2021, 2022, 2023]:
    d = fetch_acs_detailed(yr, acs_detailed_vars)
    s = fetch_acs_subject(yr, acs_subject_vars)
    merged = d.merge(s, on = ['NAME', 'state', 'year'], how = 'left')
    acs_frames.append(merged)

acs = pd.concat(acs_frames, ignore_index = True)

acs.head()

# convert numeric columns
for c in acs_detailed_vars + acs_subject_vars:
    acs[c] = pd.to_numeric(acs[c], errors = 'coerce')

# build modeling fields
acs['median_household_income'] = acs['B19013_001E']

acs['pct_housing_cost_burden'] = (acs["B25070_007E"] +
                                  acs["B25070_008E"] +
                                  acs["B25070_009E"] +
                                  acs["B25070_010E"]
                                 )/ acs["B25070_001E"] * 100

acs['pct_uninsured'] = acs['S2701_C05_051E']
```

```
acs_final = acs[['NAME', 'state', 'year', 'median_household_income',
                 'pct_housing_cost_burden', 'pct_uninsured']].rename(columns =
{'NAME': 'State'})

# expand acs annual -> monthly, carry 2023 over to 2024
months = pd.DataFrame({'month': pd.date_range('2021-07-01', '2024-09-01', freq =
'MS')})
months['year'] = months['month'].dt.year

# carry forward for 2024
acs_2024 = acs_final[acs_final['year'] == 2023].copy()
acs_2024['year'] = 2024

acs_for_panel = pd.concat([acs_final, acs_2024], ignore_index = True)
acs_monthly = acs_for_panel.merge(months, on = 'year', how = 'inner').drop(columns =
['year'])

# merge all datasets
def standardize_state_month(df, state_col = 'state', month_col = 'month'):
    out = df.copy()
    out[state_col] = out[state_col].astype(str).str.strip()
    out[month_col] =
pd.to_datetime(out[month_col]).dt.to_period('M').dt.to_timestamp()
    return out

# standardize all tables
mh = standardize_state_month(mh, 'State', 'month')
food = standardize_state_month(food_state_month, 'State', 'month')
ue = standardize_state_month(unemp, 'State', 'month')
acs = standardize_state_month(acs_monthly, 'State', 'month')

# keep only needed columns
acs = acs[['State', 'month', 'median_household_income',
           'pct_housing_cost_burden', 'pct_uninsured']]

# merge starting with outcome mh
master = mh.merge(food[['State', 'month', 'food_insecurity_proxy']], on = ['State',
'month'], how = 'left'
                 ).merge(ue[['State', 'month', 'unemployment_rate']], on = ['State',
'month'], how = 'left'
                        ).merge(acs, on = ['State', 'month'], how = 'left'
                               ).merge(cpi[['month', 'inflation_yoy']], on = 'month',
how = 'left')

print("MASTER SHAPE:", master.shape)
print("States:", master["State"].nunique())
print("Month range:", master["month"].min(), "→", master["month"].max())

model_df = master.dropna(subset=[
    'Value',
    'food_insecurity_proxy',
    'unemployment_rate',
    'inflation_yoy',
    'median_household_income',
    'pct_housing_cost_burden',
    'pct_uninsured'
]).copy()

model_df.to_csv('model_state_month_dataset.csv', index=False)
print('MODEL DF SHAPE:', model_df.shape)

# sort + create numeric time index
model_df = model_df.sort_values(['State', 'month']).copy()
```

```
model_df['time_index'] = model_df.groupby('State').cumcount()

# model 1 - baseline pooled regression
import statsmodels.formula.api as smf

model_1 = smf.ols(formula = '''Value ~ food_insecurity_proxy + unemployment_rate +
inflation_yoy''',
                  data = model_df
                 ).fit(cov_type = 'HC3')

print(model_1.summary())

# model 2 state fixed effects
model_2 = smf.ols(formula = '''Value ~ food_insecurity_proxy + unemployment_rate +
inflation_yoy + C(State)''',
                  data = model_df
                 ).fit(cov_type = 'cluster', cov_kwds = {'groups': model_df['State']})

print(model_2.summary())

model_2b_acs = smf.ols(
    """Value ~ food_insecurity_proxy + unemployment_rate + inflation_yoy + C(State) +
C(month)""",
    data=model_df
).fit(
    cov_type="cluster",
    cov_kwds={"groups": model_df["State"]}
)

print(model_2b.summary())

model_2b_acs = smf.ols(
    """Value ~ food_insecurity_proxy + unemployment_rate + inflation_yoy +
median_household_income + pct_housing_cost_burden +
    pct_uninsured + C(State) + C(month)""",
    data=model_df
).fit(
    cov_type="cluster",
    cov_kwds={"groups": model_df["State"]}
)

print(model_2b_acs.summary())

# model 3 lagged predictors
# create 1-month lag
for v in ['food_insecurity_proxy', 'unemployment_rate', 'inflation_yoy']:
    model_df[f'{v}_lag1'] = model_df.groupby('State')[v].shift(1)

lag_df = model_df.dropna(subset = ['food_insecurity_proxy_lag1',
'unemployment_rate_lag1', 'inflation_yoy_lag1'

# estimate lagged FE model
model_3 = smf.ols('''Value ~ food_insecurity_proxy_lag1 + unemployment_rate_lag1 +
inflation_yoy_lag1 + C(State) + C(month)''',
                  data = lag_df
                 ).fit(cov_type = 'cluster', cov_kwds = {'groups': lag_df['State']})

print(model_3.summary())

import matplotlib.pyplot as plt

# ensure month is monthly timestamp
master['month'] = pd.to_datetime(master['month']).dt.to_period('M').dt.to_timestamp()
```

```
# group master dataset by state and compute the mean mental health
# prevalence value for each state across study period
state_avg = master.groupby('State', as_index = False).agg(
    mh_avg = ('Value', 'mean')
)

# create mapping from full state names to 2-letter abbreviations (needed for plotly)
state_to_abb = {
"Alabama":"AL","Alaska":"AK","Arizona":"AZ","Arkansas":"AR","California":"CA","Colorad
o":"CO","Connecticut":"CT",
    "Delaware":"DE","District of
Columbia":"DC","Florida":"FL","Georgia":"GA","Hawaii":"HI","Idaho":"ID","Illinois":"IL
",

"Indiana":"IN","Iowa":"IA","Kansas":"KS","Kentucky":"KY","Louisiana":"LA","Maine":"ME"
,"Maryland":"MD",

"Massachusetts":"MA","Michigan":"MI","Minnesota":"MN","Mississippi":"MS","Missouri":"M
O","Montana":"MT",
    "Nebraska":"NE","Nevada":"NV","New Hampshire":"NH","New Jersey":"NJ","New
York":"NY","North Carolina":"NC",
    "North
Dakota":"ND","Ohio":"OH","Oklahoma":"OK","Oregon":"OR","Pennsylvania":"PA","Rhode
Island":"RI",
    "South Carolina":"SC","South
Dakota":"SD","Tennessee":"TN","Texas":"TX","Utah":"UT","Vermont":"VT","Virginia":"VA",
    "Washington":"WA","West Virginia":"WV","Wisconsin":"WI","Wyoming":"WY"
}

# add state abbreviations to aggregated dataset
state_avg['state_abb'] = state_avg['State'].map(state_to_abb)

# drop rows with missing state abbreviations to avoid plotly errors
# visualize average prevalence values using US state map
fig = px.choropleth(
    state_avg.dropna(subset = ['state_abb']),
    locations = 'state_abb',
    locationmode = 'USA-states',
    color = 'mh_avg',
    scope = 'usa',
    title = 'Average Mental Health Prevalence by State within Study Period'
)

fig.show()

import numpy as np

# helper function to extract coefficients and confidence intervals
def coef_ci(result, terms):
    '''Extracts coefficient estimates and 95% confidence intervals for selected terms
from a fitted regression model.

    Parameters:
        result - statsmodels regression results object
        terms - list of coefficient names to extract

    Returns:
        DataFrame with term names, point estimates, and CI bounds'''
    # model coefficient estimates
    params = result.params
    # confidence intervals for all coefficients
    conf = result.conf_int()
    rows = []
    for t in terms:
        # only include terms that exist in model specification
        if t in params.index:
            rows.append({
                'term': t,
                'coef': params[t], # point estimate
                'lo': conf.loc[t, 0], # lower 95% CI bound
                'hi': conf.loc[t, 1] # upper 95% CI bound
            })
        return pd.DataFrame(rows)

# define core explanatory variables of interest
terms_core = ['food_insecurity_proxy', 'unemployment_rate', 'inflation_yoy']

# extract coefficients for baseline fixed-effects model (model 2b)
df_2b = coef_ci(model_2b, terms_core)
df_2b['model'] = 'Model 2b (State + Month FE)'

# extract coefficients for extended model including ACS controls
df_2b_acs = coef_ci(model_2b_acs, terms_core)
df_2b_acs['model'] = 'Model 2b + ACS'

# combine results for comparative visualization
plot_df = pd.concat([df_2b, df_2b_acs], ignore_index = True)

# reverse list so first variable appears at top of plot
term_order = terms_core[::-1]

fig, ax = plt.subplots(figsize = (10, 5))
y_base = {t: i for i, t in enumerate(term_order)}
offsets = {'Model 2b (State + Month FE)': -0.12,
           'Model 2b + ACS': 0.12
          }

# plot coefficient estimates with 95% confidence intervals
for m in plot_df['model'].unique():
    sub = plot_df[plot_df['model'] == m]

    # adjust y positions using model-specific offsets
    y = [y_base[t] + offsets[m] for t in sub['term']]

    # coefficient estimates
    x = sub['coef'].values

    # compute asymmetric error bars from confidence intervals
    xerr = np.vstack([x - sub['lo'].values, sub['hi'].values - x])

    ax.errorbar(x, y, xerr = xerr, fmt = 'o', label = m, capsize = 3)

# add reference line
ax.axvline(0, linewidth = 1)

ax.set_yticks([y_base[t] for t in term_order])
ax.set_yticklabels(term_order)

ax.set_xlabel('Coefficient (with 95% CI)')
ax.set_title('Key Coefficients: Model 2B vs Model 2B + ACS')
```