Totpal - Project 1 Code

January 18, 2026

Title: Household Hardships and Mental Health Outcomes in the United States

Author: Allyson Totpal

Date: 18 January 2026

Modified By: Allyson Totpal

Description: Analysis on food insecurity and other socioeconomic indicators that may attribute to the growing mental health crisis in the United States.

[1]:
```
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_Reported_Frequency_of_Symptoms_During_Last_7_D
↪csv')
mh.head()
```

[1]:
```
                          Indicator              Group          State  \
0  Symptoms of Depressive Disorder  National Estimate  United States
1  Symptoms of Depressive Disorder             By Age  United States
2  Symptoms of Depressive Disorder             By Age  United States
3  Symptoms of Depressive Disorder             By Age  United States
4  Symptoms of Depressive Disorder             By Age  United States

        Subgroup Phase  Time Period     Time Period Label  \
0  United States     1            1  Apr 23 - May 5, 2020
1  18 - 29 years     1            1  Apr 23 - May 5, 2020
2  30 - 39 years     1            1  Apr 23 - May 5, 2020
3  40 - 49 years     1            1  Apr 23 - May 5, 2020
4  50 - 59 years     1            1  Apr 23 - May 5, 2020

  Time Period Start Date Time Period End Date  Value  Low CI  High CI  \
0             04/23/2020           05/05/2020   23.5    22.7     24.3
1             04/23/2020           05/05/2020   32.7    30.2     35.2
```

```
2              04/23/2020           05/05/2020   25.7     24.1      27.3
3              04/23/2020           05/05/2020   24.8     23.3      26.2
4              04/23/2020           05/05/2020   23.2     21.5      25.0

  Confidence Interval Quartile Range
0         22.7 - 24.3            NaN
1         30.2 - 35.2            NaN
2         24.1 - 27.3            NaN
3         23.3 - 26.2            NaN
4         21.5 - 25.0            NaN
```

[2]:
```
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()
```

[3]:
```
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
```

[4]:
```
# 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()
```

[4]:

	State	month	food_insecurity_proxy
0	Alabama	2021-01-01	15.10
1	Alabama	2021-02-01	11.45
2	Alabama	2021-03-01	10.50
3	Alabama	2021-04-01	11.10
4	Alabama	2021-05-01	12.55

```
[6]: # 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
```

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

[10]:
```python
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
```

[10]:
```
Index(['State', 'month', 'unemployment_rate'], dtype='object')
```

[12]:
```python
# ensure date range and sort values
unemp = unemp[(unemp['month'] >= '2021-07-01') & (unemp['month'] <=␣
 ↪'2024-09-01')]
unemp.sort_values(['State', 'month']).head()
```

[12]:

	State	month	unemployment_rate
41	Alabama	2021-07-01	3.3
40	Alabama	2021-08-01	3.2
39	Alabama	2021-09-01	3.0
38	Alabama	2021-10-01	2.9
37	Alabama	2021-11-01	2.7

[16]:
```python
unemp.to_csv("state_unemployment_monthly_2021_2024.csv", index=False)
```

[18]:
```python
# 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)
```

[22]:
```python
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 standardized 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
```

[22]:
```
Index(['month', 'cpi_index'], dtype='object')
```

[24]:
```
# comput YoY inflation
cpi['inflation_yoy'] = (cpi['cpi_index'] / cpi['cpi_index'].shift(12) - 1) * 100
```

[32]:
```
# filter analysis window
cpi = cpi[(cpi['month'] >= '2021-07-01') & (cpi['month'] <= '2024-09-01')]
cpi.head()
```

[32]:

	month	cpi_index	inflation_yoy
450	2021-07-01	4.7	-20.338983
451	2021-07-01	7.4	80.487805
452	2021-07-01	5.2	-10.344828
453	2021-07-01	6.2	8.771930
454	2021-07-01	3.9	2.631579

[90]:
```
# 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",
]
```

[92]:
```
# 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']
```

[96]:
```
# 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()
```

[96]:

```
          NAME B19013_001E B25070_001E B25070_007E B25070_008E B25070_009E  \
0      Alabama       53913      589627       46263       33147       43827
1  Puerto Rico       22237      365427       18560       12371       16757
2      Arizona       69056      912033       75773       59999       83193
3     Arkansas       52528      390637       34800       19963       26297
4   California       84907     5926357      526471      394535      569147

  B25070_010E state  year S2701_C05_051E
0      132728    01  2021            9.8
1       58726    72  2021            5.7
2      217519    04  2021           10.7
3       74973    05  2021            9.2
4     1633990    06  2021            7.0
```

[98]:

```python
# 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'})
```

[104]:

```python
# 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'])
```

[106]:
```python
# 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
```

[108]:
```python
# 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')
```

[116]:
```python
# 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())
```

```
MASTER SHAPE: (241425, 24)
States: 52
Month range: 2021-07-01 00:00:00 → 2024-08-01 00:00:00
```

[118]:
```python
master.head()
```

[118]:
```
                          Indicator   Group          State Subgroup Phase  \
0  Symptoms of Depressive Disorder  By Sex  United States   Female   4.2
1  Symptoms of Depressive Disorder  By Sex  United States   Female   4.2
2  Symptoms of Depressive Disorder  By Sex  United States   Female   4.2
3  Symptoms of Depressive Disorder  By Sex  United States   Female   4.2
4  Symptoms of Depressive Disorder  By Sex  United States   Female   4.2

   Time Period      Time Period Label Time Period Start Date  \
0           71  Jul 23 - Aug 19, 2024             07/23/2024
1           71  Jul 23 - Aug 19, 2024             07/23/2024
```

```
2           71  Jul 23 - Aug 19, 2024           07/23/2024
3           71  Jul 23 - Aug 19, 2024           07/23/2024
4           71  Jul 23 - Aug 19, 2024           07/23/2024

  Time Period End Date  Value  …  start_date  WEEK      month  year  \
0           08/19/2024   14.2  …  2024-07-23    71 2024-07-01  2024
1           08/19/2024   14.2  …  2024-07-23    71 2024-07-01  2024
2           08/19/2024   14.2  …  2024-07-23    71 2024-07-01  2024
3           08/19/2024   14.2  …  2024-07-23    71 2024-07-01  2024
4           08/19/2024   14.2  …  2024-07-23    71 2024-07-01  2024

  food_insecurity_proxy  unemployment_rate median_household_income  \
0                   NaN                NaN                     NaN
1                   NaN                NaN                     NaN
2                   NaN                NaN                     NaN
3                   NaN                NaN                     NaN
4                   NaN                NaN                     NaN

  pct_housing_cost_burden  pct_uninsured  inflation_yoy
0                     NaN            NaN      19.444444
1                     NaN            NaN     -18.867925
2                     NaN            NaN       5.882353
3                     NaN            NaN     -11.764706
4                     NaN            NaN     -14.285714

[5 rows x 24 columns]
```

[134]:

```python
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)
```

```
MODEL DF SHAPE: (56250, 24)
```

[136]:

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

[138]:

```python
# 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())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Value   R-squared:                       0.119
Model:                            OLS   Adj. R-squared:                  0.119
Method:                 Least Squares   F-statistic:                     2395.
Date:                Sat, 20 Dec 2025   Prob (F-statistic):               0.00
Time:                        17:53:34   Log-Likelihood:            -1.7345e+05
No. Observations:               56250   AIC:                         3.469e+05
Df Residuals:                   56246   BIC:                         3.470e+05
Df Model:                           3
Covariance Type:                  HC3
================================================================================
=========
                            coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
---------
Intercept                19.2449      0.130    147.598      0.000      18.989
19.500
food_insecurity_proxy     0.6710      0.008     84.712      0.000       0.655
0.687
unemployment_rate         0.2548      0.023     11.040      0.000       0.210
0.300
inflation_yoy            -0.0009      0.001     -1.221      0.222      -0.002
0.001
==============================================================================
Omnibus:                      877.768   Durbin-Watson:                   0.086
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              504.964
Skew:                           0.015   Prob(JB):                    2.23e-110
Kurtosis:                       2.537   Cond. No.                         179.
==============================================================================

Notes:
[1] Standard Errors are heteroscedasticity robust (HC3)
```

[140]:

```python
# 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())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Value   R-squared:                       0.206
Model:                            OLS   Adj. R-squared:                  0.205
Method:                 Least Squares   F-statistic:                     152.0
Date:                Sat, 20 Dec 2025   Prob (F-statistic):           9.63e-16
Time:                        17:55:57   Log-Likelihood:            -1.7054e+05
No. Observations:               56250   AIC:                         3.411e+05
Df Residuals:                   56222   BIC:                         3.414e+05
Df Model:                          27
Covariance Type:              cluster
================================================================================
====================
                                      coef    std err          z      P>|z|
[0.025      0.975]
--------------------------------------------------------------------------------
--------------------
Intercept                          26.5486      0.984     26.977      0.000
24.620      28.477
C(State)[T.Alaska]                 -1.6302      0.424     -3.845      0.000
-2.461      -0.799
C(State)[T.Arizona]                -2.2004      0.254     -8.660      0.000
-2.698      -1.702
C(State)[T.Arkansas]                0.6053      0.143      4.230      0.000
0.325       0.886
C(State)[T.California]             -2.4417      0.503     -4.857      0.000
-3.427      -1.456
C(State)[T.Colorado]               -2.7425      0.234    -11.720      0.000
-3.201      -2.284
C(State)[T.Connecticut]            -4.9312      0.327    -15.103      0.000
-5.571      -4.291
C(State)[T.Delaware]               -5.1931      0.345    -15.039      0.000
-5.870      -4.516
C(State)[T.District of Columbia]   -5.0799      0.547     -9.282      0.000
-6.153      -4.007
C(State)[T.Florida]                -2.5250      0.143    -17.698      0.000
-2.805      -2.245
C(State)[T.Georgia]                -2.3052      0.152    -15.127      0.000
-2.604      -2.006
```

```
C(State)[T.Hawaii]                    -4.7291      0.186    -25.485      0.000
-5.093      -4.365
C(State)[T.Idaho]                     -2.8168      0.153    -18.395      0.000
-3.117      -2.517
C(State)[T.Illinois]                  -4.0068      0.444     -9.020      0.000
-4.877      -3.136
C(State)[T.Indiana]                   -2.3815      0.148    -16.083      0.000
-2.672      -2.091
C(State)[T.Iowa]                      -4.5296      0.127    -35.608      0.000
-4.779      -4.280
C(State)[T.Kansas]                    -2.9812      0.135    -22.109      0.000
-3.245      -2.717
C(State)[T.Kentucky]                   0.4457      0.349      1.279      0.201
-0.238       1.129
C(State)[T.Louisiana]                  1.5548      0.354      4.397      0.000
0.862       2.248
C(State)[T.Maine]                     -3.0403      0.223    -13.635      0.000
-3.477      -2.603
C(State)[T.Maryland]                  -4.8931      0.130    -37.707      0.000
-5.147      -4.639
C(State)[T.Massachusetts]             -3.4113      0.263    -12.971      0.000
-3.927      -2.896
C(State)[T.Michigan]                  -3.8796      0.363    -10.694      0.000
-4.591      -3.169
C(State)[T.Minnesota]                 -6.1846      0.253    -24.434      0.000
-6.681      -5.689
C(State)[T.Mississippi]                0.2076      0.370      0.562      0.574
-0.517       0.932
food_insecurity_proxy                  0.2770      0.047      5.942      0.000
0.186       0.368
unemployment_rate                      0.1658      0.221      0.749      0.454
-0.268       0.599
inflation_yoy                         -0.0005      0.000     -1.777      0.075
-0.001    4.73e-05
==============================================================================
Omnibus:                     2496.282   Durbin-Watson:                   0.094
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1032.793
Skew:                           0.004   Prob(JB):                    5.39e-225
Kurtosis:                       2.336   Cond. No.                         813.
==============================================================================

Notes:
[1] Standard Errors are robust to cluster correlation (cluster)
```

```
/opt/anaconda3/lib/python3.11/site-packages/statsmodels/base/model.py:1888:
ValueWarning: covariance of constraints does not have full rank. The number of
constraints is 27, but rank is 3
  warnings.warn('covariance of constraints does not have full '
```

[164]:
```
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())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Value   R-squared:                       0.263
Model:                            OLS   Adj. R-squared:                  0.262
Method:                 Least Squares   F-statistic:                     504.5
Date:                Sat, 20 Dec 2025   Prob (F-statistic):           4.76e-27
Time:                        22:44:01   Log-Likelihood:            -1.6843e+05
No. Observations:               56250   AIC:                         3.370e+05
Df Residuals:                   56198   BIC:                         3.374e+05
Df Model:                          51
Covariance Type:              cluster
================================================================================
================================
                                                  coef    std err          z
P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
--------------------------------
Intercept                                      36.4035     12.122      3.003
0.003      12.645      60.162
C(State)[T.Alaska]                              5.7043      3.544      1.610
0.107      -1.242      12.650
C(State)[T.Arizona]                             1.3041      2.576      0.506
0.613      -3.745       6.354
C(State)[T.Arkansas]                            0.0123      0.571      0.021
0.983      -1.107       1.132
C(State)[T.California]                          6.2756      4.148      1.513
0.130      -1.854      14.405
C(State)[T.Colorado]                            5.2952      3.643      1.453
0.146      -1.846      12.436
C(State)[T.Connecticut]                         3.4654      3.423      1.012
0.311      -3.244      10.175
C(State)[T.Delaware]                            0.7011      2.271      0.309
0.758      -3.750       5.152
C(State)[T.District of Columbia]                7.8132      4.431      1.763
0.078      -0.872      16.498
C(State)[T.Florida]                            -1.3025      2.767     -0.471
```

```
0.638      -6.725       4.120
C(State)[T.Georgia]                              0.4005      2.683      0.149
0.881      -4.858       5.659
C(State)[T.Hawaii]                               4.4994      3.902      1.153
0.249      -3.148      12.147
C(State)[T.Idaho]                                0.7901      1.488      0.531
0.595      -2.126       3.706
C(State)[T.Illinois]                             1.4056      2.018      0.697
0.486      -2.549       5.360
C(State)[T.Indiana]                             -0.0235      0.931     -0.025
0.980      -1.848       1.801
C(State)[T.Iowa]                                -0.7437      1.606     -0.463
0.643      -3.891       2.404
C(State)[T.Kansas]                              -0.1749      0.919     -0.190
0.849      -1.975       1.625
C(State)[T.Kentucky]                             1.2118      1.396      0.868
0.385      -1.524       3.948
C(State)[T.Louisiana]                            0.4363      0.923      0.472
0.637      -1.374       2.246
C(State)[T.Maine]                                0.4356      1.270      0.343
0.732      -2.054       2.925
C(State)[T.Maryland]                             5.2177      4.132      1.263
0.207      -2.882      13.317
C(State)[T.Massachusetts]                        7.3316      4.131      1.775
0.076      -0.766      15.429
C(State)[T.Michigan]                            -1.1841      1.656     -0.715
0.475      -4.430       2.062
C(State)[T.Minnesota]                            0.8219      2.573      0.319
0.749      -4.222       5.865
C(State)[T.Mississippi]                         -1.9903      0.952     -2.090
0.037      -3.857      -0.124
C(month)[T.Timestamp('2021-08-01 00:00:00')]    -0.0125      0.384     -0.033
0.974      -0.764       0.739
C(month)[T.Timestamp('2021-09-01 00:00:00')]     0.1684      0.488      0.345
0.730      -0.787       1.124
C(month)[T.Timestamp('2021-12-01 00:00:00')]    -0.1697      0.610     -0.278
0.781      -1.364       1.025
C(month)[T.Timestamp('2022-01-01 00:00:00')]     1.4145      0.764      1.851
0.064      -0.083       2.912
C(month)[T.Timestamp('2022-03-01 00:00:00')]     0.6458      0.934      0.691
0.489      -1.186       2.477
C(month)[T.Timestamp('2022-04-01 00:00:00')]    -0.6821      1.067     -0.639
0.523      -2.774       1.409
C(month)[T.Timestamp('2022-06-01 00:00:00')]     2.4951      1.014      2.461
0.014       0.508       4.482
C(month)[T.Timestamp('2022-07-01 00:00:00')]     1.5559      0.975      1.596
0.110      -0.355       3.467
C(month)[T.Timestamp('2022-09-01 00:00:00')]     5.3897      0.877      6.149
```

```
0.000       3.672       7.108
C(month)[T.Timestamp('2022-10-01 00:00:00')]     4.9159      0.767      6.407
0.000       3.412       6.420
C(month)[T.Timestamp('2022-11-01 00:00:00')]     3.2328      0.912      3.546
0.000       1.446       5.019
C(month)[T.Timestamp('2022-12-01 00:00:00')]     2.5362      0.781      3.249
0.001       1.006       4.066
C(month)[T.Timestamp('2023-01-01 00:00:00')]     2.0486      1.075      1.906
0.057      -0.059       4.156
C(month)[T.Timestamp('2023-02-01 00:00:00')]     2.2038      1.123      1.963
0.050       0.003       4.405
C(month)[T.Timestamp('2023-03-01 00:00:00')]     2.4323      1.219      1.996
0.046       0.044       4.821
C(month)[T.Timestamp('2023-04-01 00:00:00')]     1.9925      1.334      1.494
0.135      -0.622       4.607
C(month)[T.Timestamp('2023-06-01 00:00:00')]     2.0696      1.240      1.669
0.095      -0.361       4.501
C(month)[T.Timestamp('2023-07-01 00:00:00')]     2.2391      1.063      2.107
0.035       0.157       4.322
C(month)[T.Timestamp('2023-08-01 00:00:00')]    -0.8909      1.277     -0.698
0.485      -3.394       1.612
C(month)[T.Timestamp('2023-09-01 00:00:00')]     2.3965      0.950      2.523
0.012       0.535       4.258
C(month)[T.Timestamp('2023-10-01 00:00:00')]     3.4404      1.227      2.805
0.005       1.036       5.845
food_insecurity_proxy                            0.2647      0.050      5.337
0.000       0.167       0.362
unemployment_rate                                0.1112      0.275      0.405
0.685      -0.427       0.649
inflation_yoy                                -5.546e-15   3.33e-12     -0.002
0.999   -6.53e-12    6.52e-12
median_household_income                         -0.0003      0.000     -2.448
0.014      -0.001   -5.68e-05
pct_housing_cost_burden                          0.0971      0.101      0.958
0.338      -0.102       0.296
pct_uninsured                                    0.1671      0.439      0.381
0.703      -0.693       1.027
==============================================================================
Omnibus:                     5032.771   Durbin-Watson:                   0.101
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1587.363
Skew:                          -0.077   Prob(JB):                         0.00
Kurtosis:                       2.192   Cond. No.                     1.06e+07
==============================================================================

Notes:
[1] Standard Errors are robust to cluster correlation (cluster)
[2] The condition number is large, 1.06e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
/opt/anaconda3/lib/python3.11/site-packages/statsmodels/base/model.py:1888:
ValueWarning: covariance of constraints does not have full rank. The number of
constraints is 51, but rank is 24
  warnings.warn('covariance of constraints does not have full '
```

[156]:
```python
# 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'
```

[158]:
```python
# 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())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Value   R-squared:                       0.258
Model:                            OLS   Adj. R-squared:                  0.257
Method:                 Least Squares   F-statistic:                     2109.
Date:                Sat, 20 Dec 2025   Prob (F-statistic):           1.72e-34
Time:                        18:37:15   Log-Likelihood:            -1.6854e+05
No. Observations:               56225   AIC:                         3.372e+05
Df Residuals:                   56176   BIC:                         3.376e+05
Df Model:                          48
Covariance Type:              cluster
================================================================================
================================
                                                     coef    std err          z
P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
--------------------------------
Intercept                                         25.4092      1.343     18.913
0.000      22.776      28.042
C(State)[T.Alaska]                                -1.9239      0.611     -3.149
0.002      -3.121      -0.726
C(State)[T.Arizona]                               -2.3689      0.370     -6.404
0.000      -3.094      -1.644
C(State)[T.Arkansas]                               0.4826      0.187      2.588
0.010       0.117       0.848
C(State)[T.California]                            -2.8300      0.698     -4.052
```

```
0.000      -4.199      -1.461
C(State)[T.Colorado]                            -2.8019      0.362     -7.738
0.000      -3.512      -2.092
C(State)[T.Connecticut]                         -5.1409      0.479    -10.736
0.000      -6.079      -4.202
C(State)[T.Delaware]                            -5.4026      0.513    -10.529
0.000      -6.408      -4.397
C(State)[T.District of Columbia]                -5.4827      0.775     -7.072
0.000      -7.002      -3.963
C(State)[T.Florida]                             -2.6279      0.203    -12.933
0.000      -3.026      -2.230
C(State)[T.Georgia]                             -2.4302      0.207    -11.760
0.000      -2.835      -2.025
C(State)[T.Hawaii]                              -4.7519      0.286    -16.626
0.000      -5.312      -4.192
C(State)[T.Idaho]                               -2.7933      0.227    -12.331
0.000      -3.237      -2.349
C(State)[T.Illinois]                            -4.3167      0.638     -6.762
0.000      -5.568      -3.065
C(State)[T.Indiana]                             -2.4434      0.227    -10.749
0.000      -2.889      -1.998
C(State)[T.Iowa]                                -4.5250      0.192    -23.555
0.000      -4.902      -4.148
C(State)[T.Kansas]                              -2.9299      0.187    -15.636
0.000      -3.297      -2.563
C(State)[T.Kentucky]                             0.1581      0.467      0.338
0.735      -0.758       1.074
C(State)[T.Louisiana]                            1.2281      0.429      2.864
0.004       0.388       2.069
C(State)[T.Maine]                               -2.9833      0.322     -9.267
0.000      -3.614      -2.352
C(State)[T.Maryland]                            -4.8883      0.196    -24.930
0.000      -5.273      -4.504
C(State)[T.Massachusetts]                       -3.5051      0.406     -8.640
0.000      -4.300      -2.710
C(State)[T.Michigan]                            -4.1419      0.516     -8.028
0.000      -5.153      -3.131
C(State)[T.Minnesota]                           -6.0418      0.326    -18.526
0.000      -6.681      -5.403
C(State)[T.Mississippi]                         -0.1380      0.428     -0.322
0.747      -0.977       0.701
C(month)[T.Timestamp('2021-08-01 00:00:00')]    -0.0028      0.390     -0.007
0.994      -0.767       0.761
C(month)[T.Timestamp('2021-09-01 00:00:00')]     0.2137      0.508      0.421
0.674      -0.781       1.208
C(month)[T.Timestamp('2021-12-01 00:00:00')]    -0.0086      0.615     -0.014
0.989      -1.215       1.198
C(month)[T.Timestamp('2022-01-01 00:00:00')]    -0.0643      0.513     -0.125
```

```
0.900      -1.069       0.941
C(month)[T.Timestamp('2022-03-01 00:00:00')]    -0.8132      0.774     -1.051
0.293      -2.329       0.703
C(month)[T.Timestamp('2022-04-01 00:00:00')]    -2.1707      0.853     -2.544
0.011      -3.843      -0.498
C(month)[T.Timestamp('2022-06-01 00:00:00')]     1.0345      0.848      1.220
0.222      -0.627       2.696
C(month)[T.Timestamp('2022-07-01 00:00:00')]     0.0998      0.761      0.131
0.896      -1.392       1.592
C(month)[T.Timestamp('2022-09-01 00:00:00')]     3.9433      0.688      5.731
0.000       2.595       5.292
C(month)[T.Timestamp('2022-10-01 00:00:00')]     3.4642      0.771      4.495
0.000       1.954       4.975
C(month)[T.Timestamp('2022-11-01 00:00:00')]     1.7895      0.827      2.164
0.030       0.169       3.410
C(month)[T.Timestamp('2022-12-01 00:00:00')]     1.0820      0.713      1.518
0.129      -0.315       2.479
C(month)[T.Timestamp('2023-01-01 00:00:00')]    -0.1355      0.745     -0.182
0.856      -1.595       1.325
C(month)[T.Timestamp('2023-02-01 00:00:00')]    -0.0158      0.606     -0.026
0.979      -1.204       1.173
C(month)[T.Timestamp('2023-03-01 00:00:00')]     0.2547      0.920      0.277
0.782      -1.549       2.059
C(month)[T.Timestamp('2023-04-01 00:00:00')]    -0.1922      0.934     -0.206
0.837      -2.022       1.638
C(month)[T.Timestamp('2023-06-01 00:00:00')]    -0.1564      0.763     -0.205
0.837      -1.651       1.338
C(month)[T.Timestamp('2023-07-01 00:00:00')]    -0.0016      0.798     -0.002
0.998      -1.565       1.561
C(month)[T.Timestamp('2023-08-01 00:00:00')]    -3.1845      0.816     -3.905
0.000      -4.783      -1.586
C(month)[T.Timestamp('2023-09-01 00:00:00')]     0.0990      0.743      0.133
0.894      -1.358       1.556
C(month)[T.Timestamp('2023-10-01 00:00:00')]     1.1242      0.728      1.543
0.123      -0.303       2.552
food_insecurity_proxy_lag1                       0.3095      0.056      5.480
0.000       0.199       0.420
unemployment_rate_lag1                           0.3534      0.292      1.212
0.226      -0.218       0.925
inflation_yoy_lag1                            -7.84e-05   2.14e-05     -3.657
0.000      -0.000   -3.64e-05
==============================================================================
Omnibus:                     4724.264   Durbin-Watson:                   0.100
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1546.584
Skew:                          -0.085   Prob(JB):                         0.00
Kurtosis:                       2.205   Cond. No.                     1.20e+03
==============================================================================
```

```
Notes:
[1] Standard Errors are robust to cluster correlation (cluster)
[2] The condition number is large, 1.2e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
/opt/anaconda3/lib/python3.11/site-packages/statsmodels/base/model.py:1888:
ValueWarning: covariance of constraints does not have full rank. The number of
constraints is 48, but rank is 24
  warnings.warn('covariance of constraints does not have full '
```

[166]:
```python
import matplotlib.pyplot as plt

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

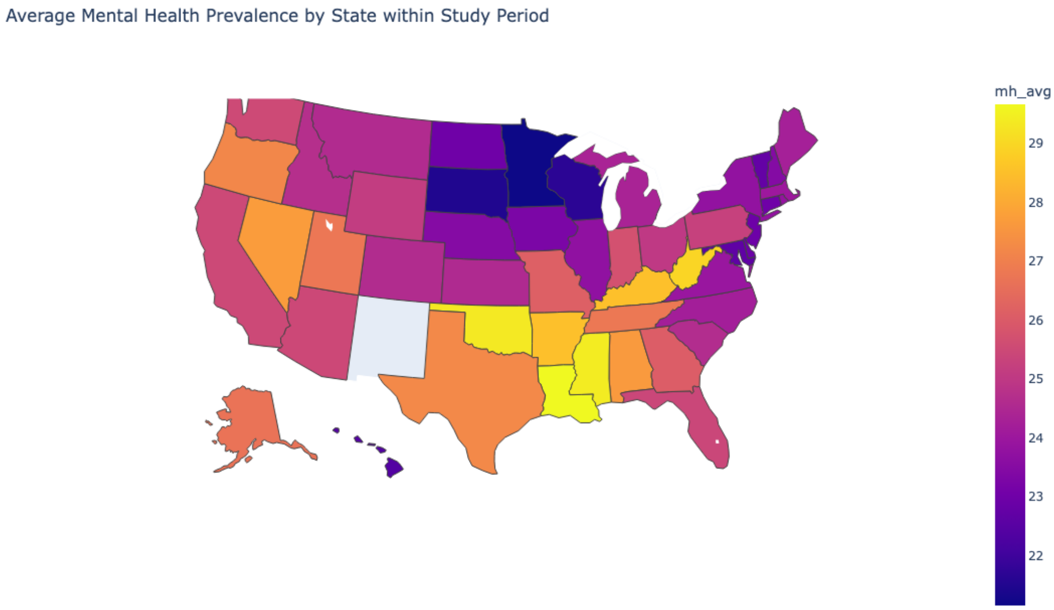
[2]:
```python
# group dataset by month and compute national averages
nat = master.groupby('month', as_index = False).agg(
    anxdep = ('Value', 'mean'),
    food_insecurity_proxy = ('food_insecurity_proxy', 'mean')
)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[2], line 2
      1 # group dataset by month and compute national averages
----> 2 nat = master.groupby('month', as_index = False).agg(
      3     anxdep = ('Value', 'mean'),
      4     food_insecurity_proxy = ('food_insecurity_proxy', 'mean')
      5 )

NameError: name 'master' is not defined
```

Visualization 2

[195]:
```python
# 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","Colorado":"CO","Connecticut":"CT",
    "Delaware":"DE","District of Columbia":"DC","Florida":"FL","Georgia":
 ↪"GA","Hawaii":"HI","Idaho":"ID","Illinois":"IL",
```

```
    "Indiana":"IN","Iowa":"IA","Kansas":"KS","Kentucky":"KY","Louisiana":
 ↪"LA","Maine":"ME","Maryland":"MD",
    "Massachusetts":"MA","Michigan":"MI","Minnesota":"MN","Mississippi":
 ↪"MS","Missouri":"MO","Montana":"MT",
    "Nebraska":"NE","Nevada":"NV","New Hampshire":"NH","New Jersey":"NJ","New␣
 ↪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()
```



Visualization 3

[206]:
```
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)
```

[214]:

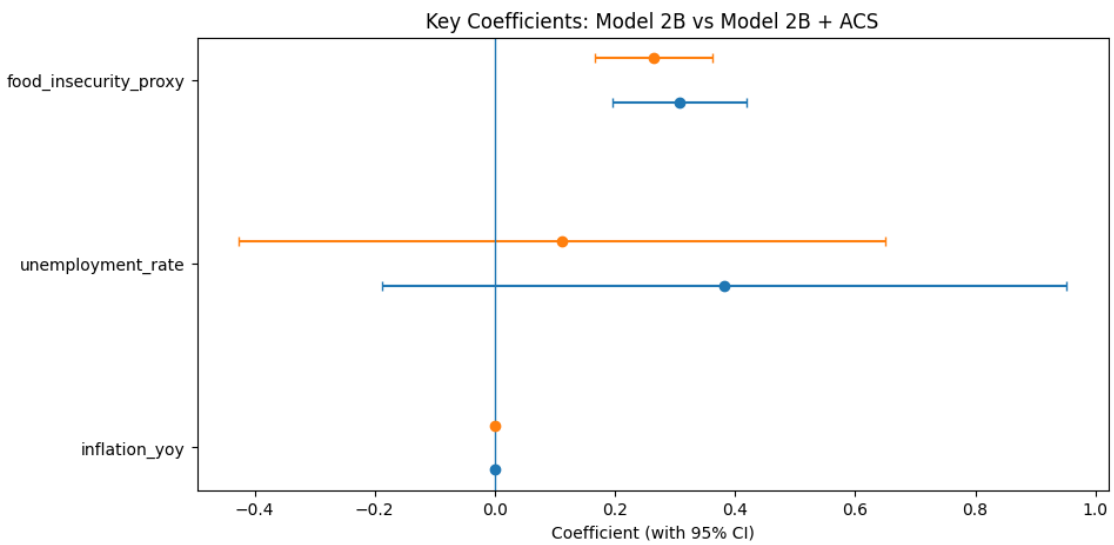
```
# define core explanatory variables of interest
terms_core = ['food_insecurity_proxy', 'unemployment_rate', 'inflation_yoy']

# 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]
```

[218]:

```
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')
```

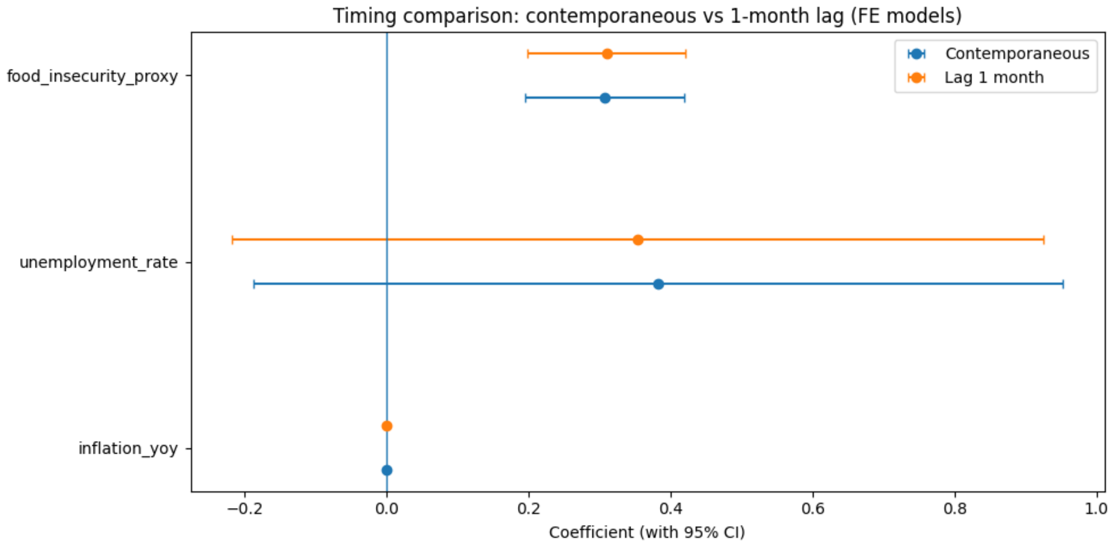
[218]: Text(0.5, 1.0, 'Key Coefficients: Model 2B vs Model 2B + ACS')



Visualization 4

[188]:
```
# each base variable is mapped to its 1-month lag counterpart
terms_lag = {
    "food_insecurity_proxy": "food_insecurity_proxy_lag1",
    "unemployment_rate": "unemployment_rate_lag1",
    "inflation_yoy": "inflation_yoy_lag1"
}

rows = []
# pre-compute confidence intervals for each model
conf_2b = model_2b.conf_int()
conf_3 = model_3.conf_int()

for base, lag in terms_lag.items():
    # contemporaneous (model 2b)
    if base in model_2b.params.index:
        rows.append({
            "term": base,
```

```
            "timing": "Contemporaneous",
            "coef": model_2b.params[base],
            "lo": conf_2b.loc[base, 0],
            "hi": conf_2b.loc[base, 1]
        })
    # lagged (model 3)
    if lag in model_3.params.index:
        rows.append({
            "term": base,
            "timing": "Lag 1 month",
            "coef": model_3.params[lag],
            "lo": conf_3.loc[lag, 0],
            "hi": conf_3.loc[lag, 1]
        })

lag_plot = pd.DataFrame(rows)
# top to bottom plotting order
term_order = list(terms_lag.keys())[::-1]

fig, ax = plt.subplots(figsize=(10, 5))

y_base = {t: i for i, t in enumerate(term_order)}
# small verticial offsets to avoid overlap between timing estimates
offsets = {"Contemporaneous": -0.12, "Lag 1 month": 0.12}

for timing in ["Contemporaneous", "Lag 1 month"]:
    sub = lag_plot[lag_plot["timing"] == timing]
    # adjust y position using time-specific offsets
    y = [y_base[t] + offsets[timing] for t in sub["term"]]
    # coefficient estimates
    x = sub["coef"].values
    # 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=timing, capsize=3)

# reference line at 0 indicates no estimated effect
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("Timing comparison: contemporaneous vs 1-month lag (FE models)")
ax.legend()
plt.tight_layout()
plt.show()
```


Timing comparison: contemporaneous vs 1-month lag (FE models)
Contemporaneous
Lag 1 month
food_insecurity_proxy
unemployment_rate
inflation_yoy
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
Coefficient (with 95% CI)

```
[ ]:
```