Story - 6 : What Is The State of Food Security and Nutrition in the US

Victor H Torres.

DATA 608

11/17/2024

Instructions

The United Nations Food and Agriculture Organization publication, "The State of Food Security and Nutrition in the World 2022" (https://www.fao.org/documents/card/en/c/cc0639en) might lead one to the conclusion that it's an elsewhere problem. That the people who are suffering malnutrition and starvation are "elsewhere", not in our backyard. For this assignment you will need to take a closer look here at home (the US). Are US children suffering these plights?

- You may use ANY graphical package that you find useful including PowerBI, Tableau, etc.
- You will need to locate and source data that reflects food security and nutrition by state broken down by men, women, children and by age groups.
- Your analysis should demonstrate correlations that exist between level of poverty and food insecurity, malnutrition and starvation.
- Your data and analysis should also indicate what happens to the children as they mature into adults. Will they become fully functional citizens or will they require continued support?
- Your data visualizations need to tell the story for a U.S. Senator that you were lobbying to address the issue of food insecurity in the US.
- You should provide NO MORE than 2 visualizations to address your point, and, in one paragraph, you must demonstrate that these visualizations require redress by the political audience, as the individual only has time for you in the elevator.

In [1]:
```
# libraries for this project
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from matplotlib.gridspec import GridSpec, GridSpecFromSubplotSpec
%matplotlib inline
import plotly.io as pio
pio.renderers.default = 'notebook_connected'
import seaborn as sns
import geopandas as gpd
import plotly as px
from mpl_toolkits.axes_grid1.inset_locator import inset_axes
```

In [2]:
```
# Load Data for analysis
data6 = pd.read_csv("C:/Users/vitug/OneDrive/Desktop/CUNY Masters/DATA_608/dec23pub.csv")
print(data6.head)
```

```
<bound method NDFrame.head of                 HRHHID  HRMONTH  HRYEAR4  HURESPLI  HUFINAL  FILLER  \
0       70061014511774       12     2023         1      201     NaN
1       70061014511774       12     2023         1      201     NaN
2       70061014511774       12     2023         1      201     NaN
3       70061014511774       12     2023         1      201     NaN
4       70061014511774       12     2023         1      201     NaN
...                ...      ...      ...       ...      ...     ...
126827  790456071508119       12     2023         2      201     NaN
126828  962070105399566       12     2023         1      201     NaN
126829  962070105399566       12     2023         1      201     NaN
126830  962070105399566       12     2023         1      201     NaN
126831  658406016540710       12     2023         1      218     NaN

        HETENURE  HEHOUSUT  HETELHHD  HETELAVL  ...  HRFS30D1  HRFS30D2  \
0              2         1         1        -1  ...         1         1
1              2         1         1        -1  ...         1         1
2              2         1         1        -1  ...         1         1
3              2         1         1        -1  ...         1         1
4              2         1         1        -1  ...         1         1
...          ...       ...       ...       ...  ...       ...       ...
126827         1         1         1        -1  ...        -1        -1
126828         1         1         1        -1  ...        -1        -1
126829         1         1         1        -1  ...        -1        -1
126830         1         1         1        -1  ...        -1        -1
126831         1         5         1        -1  ...        -1        -1

        HRFS30D3  HRFS30D4  HRFS30D5  HRFS30D6  HRFS30D7  HRFS30D8  HRFS30D9  \
0              0        -6         1         0        -6         1         0
1              0        -6         1         0        -6         1         0
2              0        -6         1         0        -6         1         0
3              0        -6         1         0        -6         1         0
4              0        -6         1         0        -6         1         0
...          ...       ...       ...       ...       ...       ...       ...
126827        -1        -1        -1        -1        -1        -1        -1
126828        -1        -1        -1        -1        -1        -1        -1
126829        -1        -1        -1        -1        -1        -1        -1
126830        -1        -1        -1        -1        -1        -1        -1
126831        -1        -1        -1        -1        -1        -1        -1

        HRFS30DE
0             -6
1             -6
2             -6
3             -6
4             -6
...          ...
126827        -1
126828        -1
126829        -1
126830        -1
126831        -1

[126832 rows x 508 columns]>
```

In [3]:

```python
#Clean DataFrame and select columns need it for this project
def clean_cps_fss(data6):
    """
    Clean and subset CPS Food Security Supplement data to key variables of interest,
    with additional calculation for a poverty indicator.

    Args:
```

```
    df: pandas DataFrame with original CPS FSS data

Returns:
    DataFrame with cleaned and renamed columns, subset to key variables,
    and an added poverty indicator column.
"""
state_fips = {
    1: 'Alabama', 2: 'Alaska', 4: 'Arizona', 5: 'Arkansas', 6: 'California',
    8: 'Colorado', 9: 'Connecticut', 10: 'Delaware', 11: 'District of Columbia',
    12: 'Florida', 13: 'Georgia', 15: 'Hawaii', 16: 'Idaho', 17: 'Illinois',
    18: 'Indiana', 19: 'Iowa', 20: 'Kansas', 21: 'Kentucky', 22: 'Louisiana',
    23: 'Maine', 24: 'Maryland', 25: 'Massachusetts', 26: 'Michigan',
    27: 'Minnesota', 28: 'Mississippi', 29: 'Missouri', 30: 'Montana',
    31: 'Nebraska', 32: 'Nevada', 33: 'New Hampshire', 34: 'New Jersey',
    35: 'New Mexico', 36: 'New York', 37: 'North Carolina', 38: 'North Dakota',
    39: 'Ohio', 40: 'Oklahoma', 41: 'Oregon', 42: 'Pennsylvania',
    44: 'Rhode Island', 45: 'South Carolina', 46: 'South Dakota',
    47: 'Tennessee', 48: 'Texas', 49: 'Utah', 50: 'Vermont', 51: 'Virginia',
    53: 'Washington', 54: 'West Virginia', 55: 'Wisconsin', 56: 'Wyoming'
}

# Key variables to keep and their readable names
columns_to_keep = {
    # Identifiers
    'HRHHID': 'household_id',
    'HRHHID2': 'household_id_2',

    # Demographic characteristics
    'PRTAGE': 'age',
    'PESEX': 'sex',
    'PEEDUCA': 'education',
    'PTDTRACE': 'race',
    'PEHSPNON': 'hispanic',
    'HEFAMINC': 'family_income',
    'HRNUMHOU': 'household_size',
    'HETENURE': 'housing_tenure',

    # Geography
    'GESTFIPS': 'state_fips',
    'GEREG': 'region',
    'GTMETSTA': 'metro_status',

    # Food Security Status
    'HRFS12M1': 'food_security_status',
    'HRFS12MC': 'child_food_security',
    'HRFS12M8': 'adult_food_security',

    # Food Spending
    'HETS8O': 'weekly_food_spending',
    'HETS8OU': 'usual_weekly_food_spending',

    # Program Participation
    'HESP1': 'received_snap',
    'HESP6': 'received_school_lunch',
    'HESP7': 'received_school_breakfast',
    'HESP8': 'received_wic',

    # Weights
    'PWSUPWGT': 'person_supplement_weight',
    'HHSUPWGT': 'household_supplement_weight'
}
```

```python
# Create subset with renamed columns
df_clean = data6[columns_to_keep.keys()].copy()
df_clean = df_clean.rename(columns=columns_to_keep)

# Value labels for categorical variables
value_labels = {
    'food_security_status': {-1: 'High Security', 1: 'Security', 2: 'Low Security', 3: 'Very Low Security', -9: 'Very Low Security'},
    'child_food_security' : {-1: 'High Security', 1: 'Security', 2: 'Low Security', 3: 'Very Low Security', -9: 'Very Low Security'},
    'adult_food_security' : {-1: 'High Security', 1: 'Security', 2: 'Low Security', 3: 'Very Low Security',4: 'Very Low Security', -9: 'Very Low Security'},
    'sex': {1: 'Male', 2: 'Female'},
    'hispanic': {1: 'Hispanic', 2: 'Non-Hispanic'},
    'housing_tenure': {1: 'Owned/Being Bought', 2: 'Rented', 3: 'Occupied without payment'},
    'region': {1: 'Northeast', 2: 'Midwest', 3: 'South', 4: 'West'},
    'metro_status': {1: 'Metropolitan', 2: 'Non-metropolitan', 3: 'Not Identified'},
    'family_income': {
        1: 'Less than $5,000', 2: '$5,000 to $7,499', 3: '$7,500 to $9,999',
        4: '$10,000 to $12,499', 5: '$12,500 to $14,999', 6: '$15,000 to $19,999',
        7: '$20,000 to $24,999', 8: '$25,000 to $29,999', 9: '$30,000 to $34,999',
        10: '$35,000 to $39,999', 11: '$40,000 to $49,999', 12: '$50,000 to $59,999',
        13: '$60,000 to $74,999', 14: '$75,000 to $99,999', 15: '$100,000 to $149,999',
        16: '$150,000 or more'
    },
    'received_snap': {1: 'Yes', 2: 'No'},
    'received_school_lunch': {1: 'Yes', 2: 'No'},
    'received_school_breakfast': {1: 'Yes', 2: 'No'},
    'received_wic': {1: 'Yes', 2: 'No'},
    'education': {
        -1: 'Not_relevant', -2: 'Dont_know', -3: 'Refused_to_answer', -9: 'No_response',
        31: 'Less_than_1st_grade', 32: '1st-4th_grade', 33: '5th-6th_grade', 34: '7th-8th_grade',
        35: '9th_grade', 36: '10th_grade', 37: '11th_grade', 38: '12th_grade,_no_diploma',
        39: 'High_school_graduate_diploma_or_GED', 40: 'Some_college_no_degree',
        41: 'Associate_degree_occupational_vocational', 42: 'Associate_degree_academic_program',
        43: "Bachelors_degree", 44: "Masters_degree", 45: 'Professional_school_degree_MD_DDS_DVM_etc',
        46: 'Doctorate_degree_PhD_EdD'
    }
}

# Apply value labels
for col, val_map in value_labels.items():
    if col in df_clean.columns:
        df_clean[col] = df_clean[col].map(val_map).fillna(df_clean[col])

# Convert weights by dividing by 10000
weight_cols = ['person_supplement_weight', 'household_supplement_weight']
for col in weight_cols:
    if col in df_clean.columns:
        df_clean[col] = df_clean[col] / 10000

# Create a state column
df_clean['state'] = df_clean['state_fips'].map(state_fips)

# Define poverty income threshold categories
poverty_income_levels = [
    'Less than $5,000', '$5,000 to $7,499', '$7,500 to $9,999', '$10,000 to $12,499',
    '$12,500 to $14,999', '$15,000 to $19,999', '$20,000 to $24,999', '$25,000 to $29,999',
    '$30,000 to $34,999', '$35,000 to $39,999'
]

# Create poverty indicator based on income and program participation
df_clean['poverty_indicator'] = df_clean['family_income'].apply(lambda x: 1 if x in poverty_income_levels else 0)
```

```
    # Add to poverty indicator if received benefits (any program participation marked 'Yes')
    program_columns = ['received_snap', 'received_school_lunch', 'received_school_breakfast', 'received_wic']
    for col in program_columns:
        df_clean['poverty_indicator'] = df_clean.apply(lambda row: 1 if row[col] == 'Yes' else row['poverty_indicator'], axis=1)

    return df_clean
```

In [4]:
```
# Load cleaned DataFrame
cleaned_data = clean_cps_fss(data6)

# Print DF
print(cleaned_data.head)
```

```
<bound method NDFrame.head of            household_id  household_id_2  age     sex  \
0        70061014511774           16011   43  Female
1        70061014511774           16011   44    Male
2        70061014511774           16011   13  Female
3        70061014511774           16011   15  Female
4        70061014511774           16011   20  Female
...                 ...             ...  ...     ...
126827  790456071508119           16111   64  Female
126828  962070105399566           17111   35    Male
126829  962070105399566           17111   31  Female
126830  962070105399566           17111    0  Female
126831  658406016540710           16111   -1      -1

                               education  race      hispanic  \
0                        Masters_degree     2  Non-Hispanic
1       High_school_graduate_diploma_or_GED     2  Non-Hispanic
2                          Not_relevant     2  Non-Hispanic
3                             9th_grade     2  Non-Hispanic
4       High_school_graduate_diploma_or_GED     2  Non-Hispanic
...                                   ...   ...           ...
126827  High_school_graduate_diploma_or_GED     1  Non-Hispanic
126828    Associate_degree_academic_program     1  Non-Hispanic
126829    Associate_degree_academic_program     1  Non-Hispanic
126830                       Not_relevant     1  Non-Hispanic
126831                       Not_relevant    -1           -1

            family_income  household_size      housing_tenure  ...  \
0       $75,000 to $99,999               5              Rented  ...
1       $75,000 to $99,999               5              Rented  ...
2       $75,000 to $99,999               5              Rented  ...
3       $75,000 to $99,999               5              Rented  ...
4       $75,000 to $99,999               5              Rented  ...
...                    ...             ...                 ...  ...
126827  $50,000 to $59,999               2  Owned/Being Bought  ...
126828  $60,000 to $74,999               3  Owned/Being Bought  ...
126829  $60,000 to $74,999               3  Owned/Being Bought  ...
126830  $60,000 to $74,999               3  Owned/Being Bought  ...
126831                  -1               2  Owned/Being Bought  ...

        weekly_food_spending usual_weekly_food_spending received_snap  \
0                        230                        280           -1
1                        230                        280           -1
2                        230                        280           -1
3                        230                        280           -1
4                        230                        280           -1
...                      ...                        ...          ...
126827                    -1                         -1           -1
126828                    -1                         -1           -1
126829                    -1                         -1           -1
126830                    -1                         -1           -1
126831                    -1                         -1           -1

       received_school_lunch received_school_breakfast received_wic  \
0                         -1                        -1          -1
1                         -1                        -1          -1
2                         -1                        -1          -1
3                         -1                        -1          -1
4                         -1                        -1          -1
...                      ...                       ...         ...
126827                    -1                        -1          -1
126828                    -1                        -1          -1
```

```
126829                   -1                       -1          -1
126830                   -1                       -1          -1
126831                   -1                       -1          -1

        person_supplement_weight  household_supplement_weight    state  \
0                      3267.0314                    3267.0314  Alabama
1                      3227.4523                    3267.0314  Alabama
2                      4634.0388                    3267.0314  Alabama
3                      5084.4886                    3267.0314  Alabama
4                      5443.0958                    3267.0314  Alabama
...                          ...                          ...      ...
126827                    0.0000                       0.0000  Wyoming
126828                    0.0000                       0.0000  Wyoming
126829                    0.0000                       0.0000  Wyoming
126830                    0.0000                       0.0000  Wyoming
126831                    0.0000                       0.0000  Wyoming

       poverty_indicator
0                      0
1                      0
2                      0
3                      0
4                      0
...                  ...
126827                 0
126828                 0
126829                 0
126830                 0
126831                 0

[126832 rows x 26 columns]>
```

In [5]:
```python
# Filter for children (age <= 18)
children_data = cleaned_data[cleaned_data['age'] <= 18]

# Count food security status
food_security_counts = children_data['child_food_security'].value_counts()
# Create a dictionary mapping state names to abbreviations
from us import states
state_to_abbrev = {state.name: state.abbr for state in states.STATES}

# Map state names to abbreviations
children_data['state_abbrev'] = children_data['state'].map(state_to_abbrev)
```

```
C:\Users\vitug\AppData\Local\Temp\ipykernel_28152\1711683070.py:11: 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-view-versus-a-copy
```

In [6]:
```python
# create a variable for visuals
aggregated_data = children_data.groupby('state_abbrev').agg(
    food_insecurity_rate=('food_security_status', lambda x: (x == 'Low food security').mean())
).reset_index()

print(aggregated_data.head())
```

```
  state_abbrev  food_insecurity_rate
0           AK                  0.0
1           AL                  0.0
2           AR                  0.0
3           AZ                  0.0
4           CA                  0.0
```

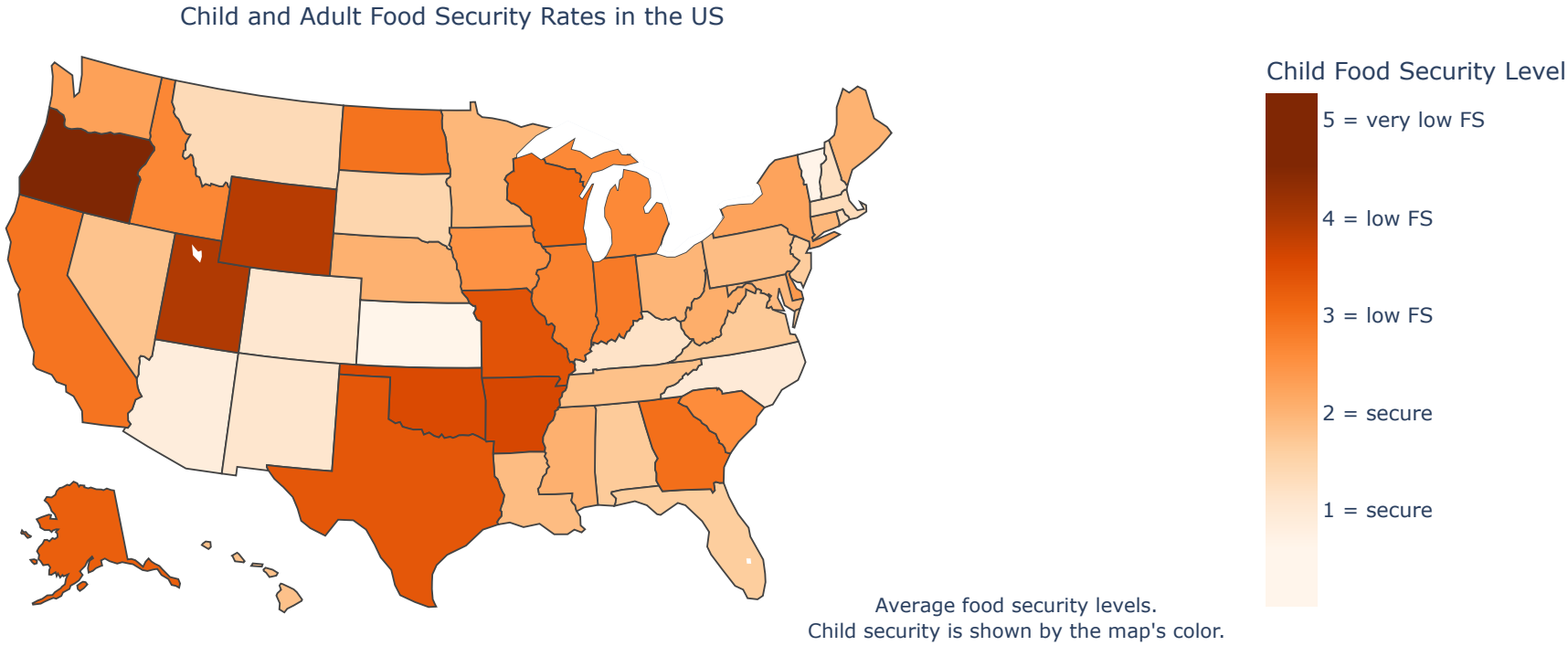
In [7]:
```
# Handle missing data for analysis
missing_states = children_data[children_data['state_abbrev'].isna()]
print(missing_states['state'].unique())
manual_mapping = {
    'District of Columbia': 'DC',
    'Puerto Rico': 'PR'  # Example for non-states
}
children_data['state_abbrev'].fillna(children_data['state'].map(manual_mapping), inplace=True)
aggregated_data = children_data.groupby('state_abbrev').agg(
    food_insecurity_rate=('food_security_status', lambda x: (x == 'Low Security').mean())
).reset_index()
print(aggregated_data.head())
```

```
['District of Columbia']
  state_abbrev  food_insecurity_rate
0           AK              0.025680
1           AL              0.029126
2           AR              0.076074
3           AZ              0.015713
4           CA              0.058996
```

```
C:\Users\vitug\AppData\Local\Temp\ipykernel_28152\3998138323.py:8: FutureWarning:

A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operat
ion inplace on the original object.



C:\Users\vitug\AppData\Local\Temp\ipykernel_28152\3998138323.py:8: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [8]:
```
# Create a heatmap to compare between child and adult food security rate
import plotly.express as px
state_food_security = children_data.groupby('state_abbrev').agg(
    avg_child_food_security=('child_food_security',lambda x: (x== 'Low Security').mean()),
    avg_adult_food_security=('adult_food_security', lambda x: (x== 'Low Security').mean())
).reset_index()

# Create a combined column to display both child and adult food security rates
state_food_security['hover_text'] = (
    "Child Food Security: " + state_food_security['avg_child_food_security'].round(2).astype(str) +
    "<br>Adult Food Security: " + state_food_security['avg_adult_food_security'].round(2).astype(str)
)

# Create the choropleth map
fig = px.choropleth(
    state_food_security,
```

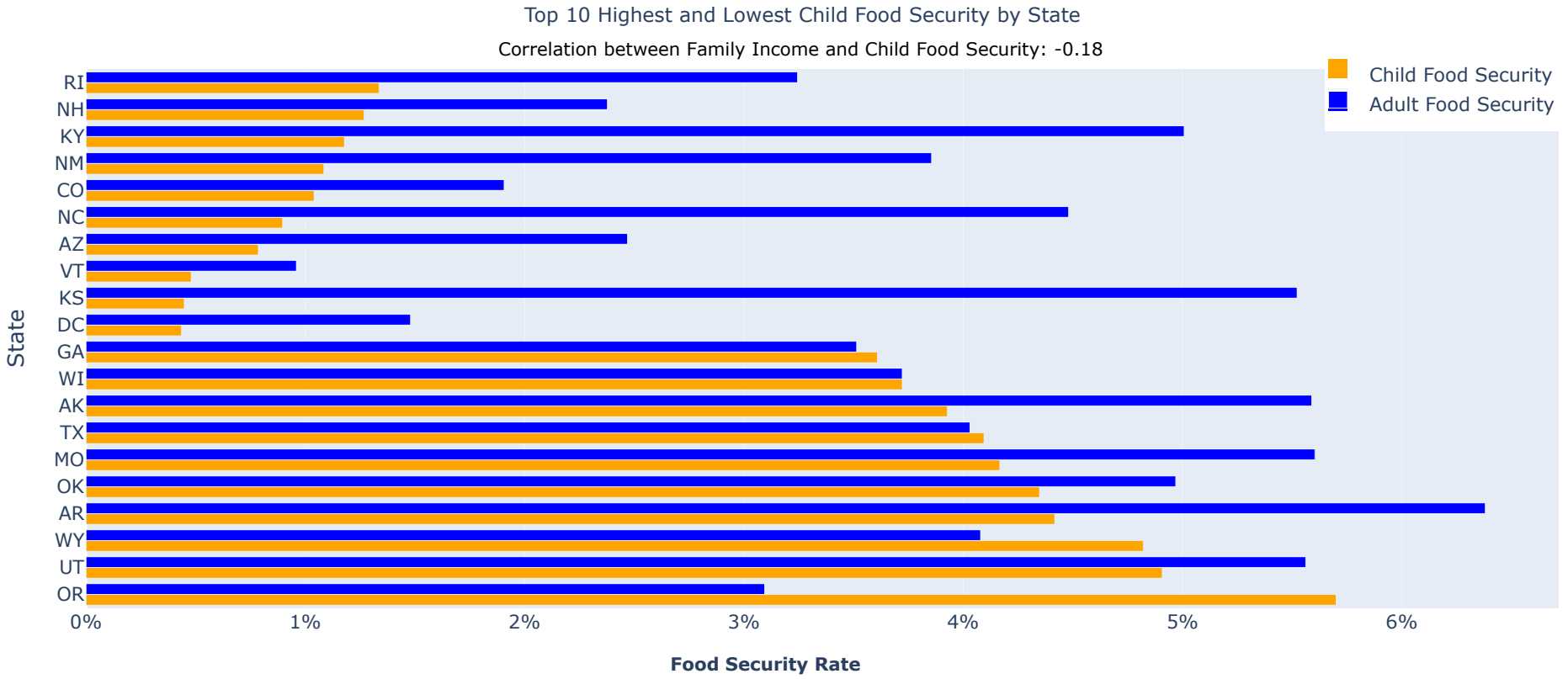
```
    locations="state_abbrev",  # State abbreviations
    locationmode="USA-states",  # Match state abbreviations to USA states
    color="avg_child_food_security",  # Color based on child food security
    hover_name="state_abbrev",  # Display state abbreviations on hover
    hover_data={'avg_child_food_security': False, 'avg_adult_food_security': False, 'hover_text': True},  # Control hover display
    color_continuous_scale="Oranges",  # Use a color scale
    title="Child Food Security Across the United States",
)

# Update map layout for better visuals
fig.update_layout(
    geo=dict(
        scope="usa",  # Focus on the USA
        projection=dict(type="albers usa"),  # USA-centric projection
        showlakes=True,  # Display lakes
        lakecolor="white",  # Set lake color
    ),
    coloraxis_colorbar=dict(
        title="Child Food Security Level",
        ticksuffix="%",
    ),
    annotations=[
        dict(
            x=0.5,
            y=1.15,
            xref="paper",
            yref="paper",
            text="Child and Adult Food Security Rates in the US",
            showarrow=False,
            font=dict(size=14),
        ),
        dict(
            x=0.05,
            y=0.05,
            xref="paper",
            yref="paper",
            text="Average food security levels.<br>Child security is shown by the map's color.",
            showarrow=False,
            font=dict(size=10),
        ),
    ]
)

# Show the map
fig.show()
```



In [9]:
```python
# Create a horizontal barplot to compare betwwen the variables and find correaltion of Child and adult food security and household income
import plotly.express as px

# convert values to numeric
def convert_income_to_numeric(income_str):
    try:
        # Remove dollar signs and commas
        income_str = income_str.replace('$', '').replace(',', '')

        # If it's a range like "$75,000 to $99,999", extract the midpoint
        if 'to' in income_str:
            low, high = income_str.split(' to ')
            low = int(low)
            high = int(high)
            return (low + high) / 2  # Take the midpoint of the range
        else:
            # If it's a single value, just convert it
            return int(income_str)
    except Exception as e:
        return None  # Handle invalid data by returning None

# Assuming the 'children_data' DataFrame has the 'family_income' column
children_data = children_data.copy()  # Make a copy of the original DataFrame

# Apply the conversion to the 'family_income' column and create a new numeric column
children_data['family_income_numeric'] = children_data['family_income'].apply(convert_income_to_numeric)

# Subset the data to get the top 10 highest and lowest states by child food security
top_10_highest = state_food_security.nlargest(10, 'avg_child_food_security')  # Top 10 highest
```

```
top_10_lowest = state_food_security.nsmallest(10, 'avg_child_food_security')  # Top 10 lowest

# Combine both subsets
subset_data = pd.concat([top_10_highest, top_10_lowest])

# Create a new dataframe for plotting
subset_data_melted = subset_data.melt(id_vars="state_abbrev",
                                      value_vars=["avg_child_food_security", "avg_adult_food_security"],
                                      var_name="food_security_type",
                                      value_name="security_rate")

# Update labels for readability
subset_data_melted["food_security_type"] = subset_data_melted["food_security_type"].replace({
    "avg_child_food_security": "Child Food Security",
    "avg_adult_food_security": "Adult Food Security"
})

# Calculate the correlation between Family Income and Child Food Security
correlation = children_data[['child_food_security', 'family_income_numeric']].copy()

# Convert 'Low Security' to 1 and 'High Security' to 0 for correlation purposes
correlation['child_food_security'] = correlation['child_food_security'].apply(lambda x: 1 if x == 'Low Security' else 0)

# Calculate correlation
correlation_matrix = correlation.corr()

# Extract correlation value
correlation_value = correlation_matrix.loc['child_food_security', 'family_income_numeric']

# Create the horizontal bar plot
fig = px.bar(
    subset_data_melted,
    x="security_rate",
    y="state_abbrev",
    color="food_security_type",
    orientation="h",  # Horizontal bars
    title="Top 10 Highest and Lowest Child Food Security by State",
    labels={"security_rate": "Food Security Rate", "state_abbrev": "State", "food_security_type": "Security Type"},
    color_discrete_map={"Child Food Security": "orange", "Adult Food Security": "blue"},
)

# Update layout for better visuals
fig.update_layout(
    barmode="group",  # Group the bars side by side
    xaxis=dict(tickformat=".0%"),  # Show percentages
    yaxis=dict(tickmode="linear", tick0=0, dtick=1),  # Ensure the y-axis has labels for each state
)

# Add a correlation annotation to the plot
fig.add_annotation(
    x=0.5,  # X position in the plot (relative to plot area)
    y=1.1,  # Y position (above the plot)
    text=f"Correlation between Family Income and Child Food Security: {correlation_value:.2f}",
    showarrow=False,
    font=dict(size=12, color="black"),
    align="center",
    xref="paper",  # Coordinate system is relative to the plot area
    yref="paper",  # Coordinate system is relative to the plot area
    borderpad=4,
    bgcolor="white",
)
```

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



Dear Senator,

After performing a deep analysis using data on children and food insecurity in the U.S, it is evident that we do have a problem on this area, and that aligns heavily with poverty levels particularly impacting children. We can clearly see in the graphs above that, there are states with higher food insecurity rates for children than adults. Poverty levels show a direct correlation with food insecurity, leaving many children without access to the nutrition needed for healthy development. As they grow, these children face heightened risks of health issues and educational setbacks. Addressing food insecurity now isn't just compassionate; it's an investment in America's future workforce and economic stability.

Sincerely,

Victor H Torres.