Name: Komal Shahid

DSC 640

Final Project: Milestone 1

The Hidden Cost of the American Dream: A Deep Dive into Childcare Economics

---

Target Audience: Policymakers, Business Leaders, and Working Parents

This analysis explores how childcare costs impact workforce participation and economic opportunity across America. By examining the relationship between childcare prices, female labor force participation, and household income, we uncover the economic barriers facing working families and their implications for policy and business decisions.

In [1]:
```
# Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pathlib import Path
import sys
from datetime import datetime
from matplotlib.patches import Polygon
from matplotlib.collections import PatchCollection
import json
import urllib.request
import geopandas as gpd
from matplotlib.colors import LinearSegmentedColormap
```

In [2]:
```
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
```

In [3]:
```
# Custom styling functions
def style_choropleth(ax, title):
    """Apply consistent styling to choropleth maps"""
    ax.axis('off')
    ax.set_title(title, pad=20, fontsize=16, fontweight='bold',
                 loc='left', fontfamily='Arial')
    # Add a subtle background color
```

```
    ax.set_facecolor('#f7f7f7')
    ax.set_aspect('equal')

def style_timeseries(ax, title, ylabel):
    """Apply consistent styling to time series plots"""
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.set_title(title, pad=20, fontsize=16, fontweight='bold',
                 loc='left', fontfamily='Arial')
    ax.set_ylabel(ylabel, fontsize=12)
    ax.grid(axis='y', linestyle='--', alpha=0.7)

def style_distribution(ax, title):
    """Apply consistent styling to distribution plots"""
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.set_title(title, pad=20, fontsize=16, fontweight='bold',
                 loc='left', fontfamily='Arial')
    ax.grid(axis='y', linestyle='--', alpha=0.7)

def save_figure(fig, filename):
    """Save figure with consistent settings and display it inline"""
    plt.show()  # Display the figure inline
    fig.savefig(figures_dir / filename, dpi=300, bbox_inches='tight', facecc
    plt.close(fig)
```

In [4]:
```
# Set style for better-looking plots
plt.style.use('seaborn-v0_8-whitegrid')
plt.rcParams['figure.figsize'] = [12, 8]
plt.rcParams['font.family'] = 'sans-serif'
plt.rcParams['font.sans-serif'] = ['Arial']
plt.rcParams['axes.grid'] = True
plt.rcParams['font.size'] = 12
plt.rcParams['axes.titlesize'] = 16
plt.rcParams['axes.labelsize'] = 12
plt.rcParams['axes.spines.top'] = False
plt.rcParams['axes.spines.right'] = False

# Custom color palette inspired by The Economist
colors = ['#2f4b7c', '#665191', '#a05195', '#d45087', '#f95d6a', '#ff7c43',
sns.set_palette(colors)

print("Libraries imported successfully!")
```

```
Libraries imported successfully!
```

In [5]:
```
# Setup paths and load data
try:
    root_dir = Path().absolute().parent.parent  # Go up two levels to reach
    data_dir = root_dir / 'data'
    output_dir = Path().absolute()  # Current directory (milestone1)
    figures_dir = output_dir  # Save figures directly in milestone1 director
    output_dir.mkdir(exist_ok=True)

    print(f"\nLoading data from: {data_dir}")
    data_file = data_dir / 'nationaldatabaseofchildcareprices.xlsx'
```

```
    if not data_file.exists():
        print(f"Error: Data file not found at {data_file}")
        sys.exit(1)

    print("Loading data (this may take a few moments)...")
    # Only load the columns we need
    needed_columns = ['State_Name', 'State_Abbreviation', 'County_Name', 'Co
                      'StudyYear', 'MCInfant', 'FLFPR_20to64', 'MHI']
    df = pd.read_excel(data_file, usecols=needed_columns)
      # Basic data info
    print(f"\nAnalyzing data from {len(df)} counties across {df['State_Name'
    print(f"Time period: {df['StudyYear'].min()} - {df['StudyYear'].max()}")

    print("\nState count verification:")
    print(df['State_Name'].unique())
    print(f"\nUnique states ({len(df['State_Name'].unique())}):")
    for state in sorted(df['State_Name'].unique()):
        print(f"- {state}")

except Exception as e:
    print(f"Error: {str(e)}")
    sys.exit(1)

# After loading data
print("\nNote: The dataset includes the District of Columbia (DC) in additic
```

```
Loading data from: /Users/komalshahid/Desktop/Bellevue University/DSC640/fin
al-project/data
Loading data (this may take a few moments)...

Analyzing data from 34567 counties across 51 states
Time period: 2008 - 2018

State count verification:
['Alabama' 'Alaska' 'Arizona' 'Arkansas' 'California' 'Colorado'
 'Connecticut' 'Delaware' 'District of Columbia' 'Florida' 'Georgia'
 'Hawaii' 'Idaho' 'Illinois' 'Indiana' 'Iowa' 'Kansas' 'Kentucky'
 'Louisiana' 'Maine' 'Maryland' 'Massachusetts' 'Michigan' 'Minnesota'
 'Mississippi' 'Missouri' 'Montana' 'Nebraska' 'Nevada' 'New Hampshire'
 'New Jersey' 'New Mexico' 'New York' 'North Carolina' 'North Dakota'
 'Ohio' 'Oklahoma' 'Oregon' 'Pennsylvania' 'Rhode Island' 'South Carolina'
 'South Dakota' 'Tennessee' 'Texas' 'Utah' 'Vermont' 'Virginia'
 'Washington' 'West Virginia' 'Wisconsin' 'Wyoming']

Unique states (51):
- Alabama
- Alaska
- Arizona
- Arkansas
- California
- Colorado
- Connecticut
- Delaware
- District of Columbia
- Florida
- Georgia
- Hawaii
- Idaho
- Illinois
- Indiana
- Iowa
- Kansas
- Kentucky
- Louisiana
- Maine
- Maryland
- Massachusetts
- Michigan
- Minnesota
- Mississippi
- Missouri
- Montana
- Nebraska
- Nevada
- New Hampshire
- New Jersey
- New Mexico
- New York
- North Carolina
- North Dakota
- Ohio
- Oklahoma
```

- Oregon
- Pennsylvania
- Rhode Island
- South Carolina
- South Dakota
- Tennessee
- Texas
- Utah
- Vermont
- Virginia
- Washington
- West Virginia
- Wisconsin
- Wyoming

Note: The dataset includes the District of Columbia (DC) in addition to the 50 states.

In [6]:
```
# Create output directories for figures
figures_dir = output_dir / 'figures'
figures_dir.mkdir(exist_ok=True)

# Add county type and income categories
df['County_Type'] = df['County_FIPS_Code'].apply(lambda x: 'Urban' if x < 20
df['Income_Category'] = pd.qcut(df['MHI'], q=3, labels=['Low Income', 'Middl
```

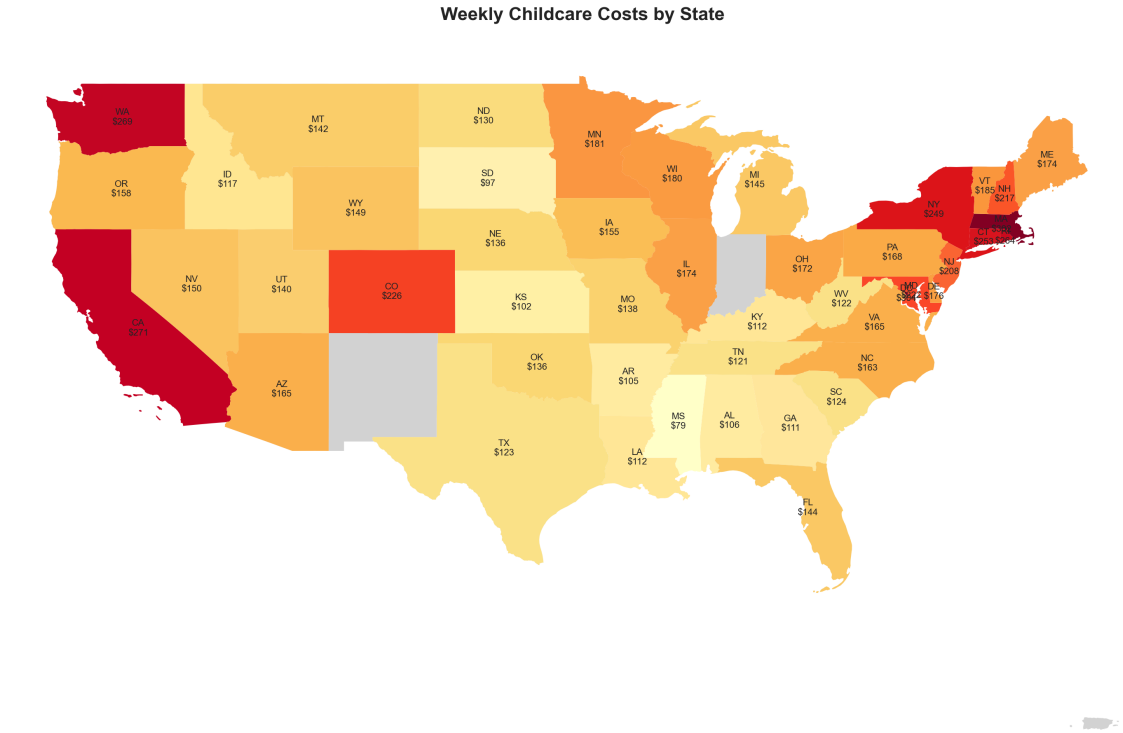
The Geography of Opportunity: Childcare Costs Across America

Understanding regional variations in childcare costs reveals economic disparities and their potential impact on workforce mobility.

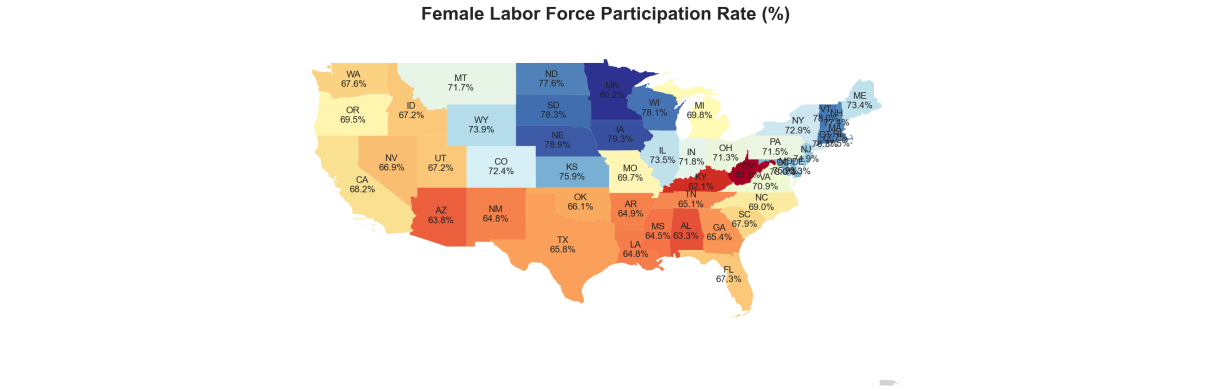
In [7]:
```
# Create an enhanced state comparison visualization using a map
plt.figure(figsize=(15, 10))

# Calculate state statistics
state_stats = df.groupby(['State_Name', 'State_Abbreviation']).agg({
    'MCInfant': ['mean', 'std'],
    'FLFPR_20to64': 'mean'  # Female Labor Force Participation Rate
}).round(2)

state_stats.columns = ['avg_price', 'price_std', 'labor_participation']
state_stats = state_stats.reset_index()

# Create figure and axes for two maps
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(15, 20), height_ratios=[2, 1])
fig.patch.set_facecolor('white')

# Load US states shapefile
usa = gpd.read_file('https://www2.census.gov/geo/tiger/GENZ2018/shp/cb_2018_
# Remove Alaska and Hawaii for better continental US visualization
usa = usa[~usa['STUSPS'].isin(['AK', 'HI'])]
```

```python
# Merge data with map
usa = usa.merge(state_stats, how='left', left_on='NAME', right_on='State_Nam

# Create maps with enhanced styling
for ax, column, title, cmap in [
    (ax1, 'avg_price', 'Weekly Childcare Costs by State', 'YlOrRd'),
    (ax2, 'labor_participation', 'Female Labor Force Participation Rate (%)'
]:
    # Plot the map
    usa.plot(column=column,
             ax=ax,
             legend=True,
             legend_kwds={'label': title,
                          'orientation': 'horizontal'},
             missing_kwds={'color': 'lightgrey'},
             cmap=cmap)

    # Add state labels
    for idx, row in usa.iterrows():
        # Get centroid for label placement
        centroid = row.geometry.centroid
        # Add state abbreviation and value
        if pd.notnull(row[column]):
            if column == 'avg_price':
                label = f"{row['STUSPS']}\n${row[column]:,.0f}"
            else:
                label = f"{row['STUSPS']}\n{row[column]:.1f}%"
            ax.annotate(label,
                        xy=(centroid.x, centroid.y),
                        ha='center', va='center',
                        fontsize=8)

    # Customize the map
    ax.axis('off')
    ax.set_title(title, pad=20, fontsize=16, fontweight='bold')

    # Add source citation
    ax.text(0.01, -0.05, 'Source: National Database of Childcare Prices',
             transform=ax.transAxes, fontsize=10, style='italic')

plt.tight_layout()
save_figure(fig, 'childcare_costs_map.png')
```

```
<Figure size 1500x1000 with 0 Axes>
```



Source: National Database of Childcare Prices

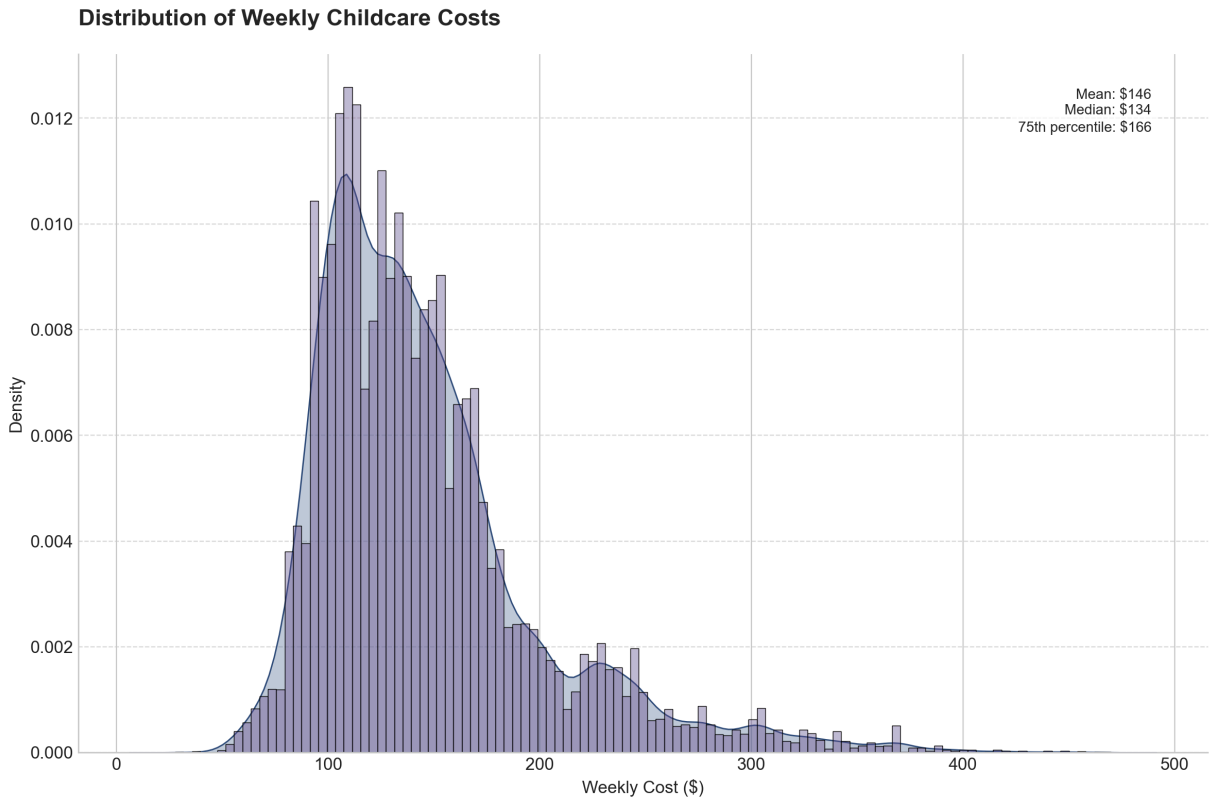


Source: National Database of Childcare Prices

The Economic Burden: Understanding Price Distributions

Analyzing the distribution of childcare costs reveals affordability challenges facing American families.

In [8]:
```
# Create an enhanced price distribution visualization
fig, ax = plt.subplots(figsize=(12, 8))
fig.patch.set_facecolor('white')

# Create a more sophisticated distribution plot
sns.kdeplot(data=df, x='MCInfant', fill=True, alpha=0.3, color=colors[0])
sns.histplot(data=df, x='MCInfant', alpha=0.4, color=colors[1], stat='densit

style_distribution(ax, 'Distribution of Weekly Childcare Costs')
ax.set_xlabel('Weekly Cost ($)', fontsize=12)
ax.set_ylabel('Density', fontsize=12)

# Add statistical annotations with better positioning
stats_text = (f'Mean: ${df["MCInfant"].mean():,.0f}\n'
              f'Median: ${df["MCInfant"].median():,.0f}\n'
              f'75th percentile: ${df["MCInfant"].quantile(0.75):,.0f}')
plt.text(0.95, 0.95, stats_text, transform=ax.transAxes,
         bbox=dict(facecolor='white', alpha=0.8, edgecolor='none'),
         va='top', ha='right', fontsize=10)

plt.tight_layout()
save_figure(fig, 'cost_distribution.png')
```

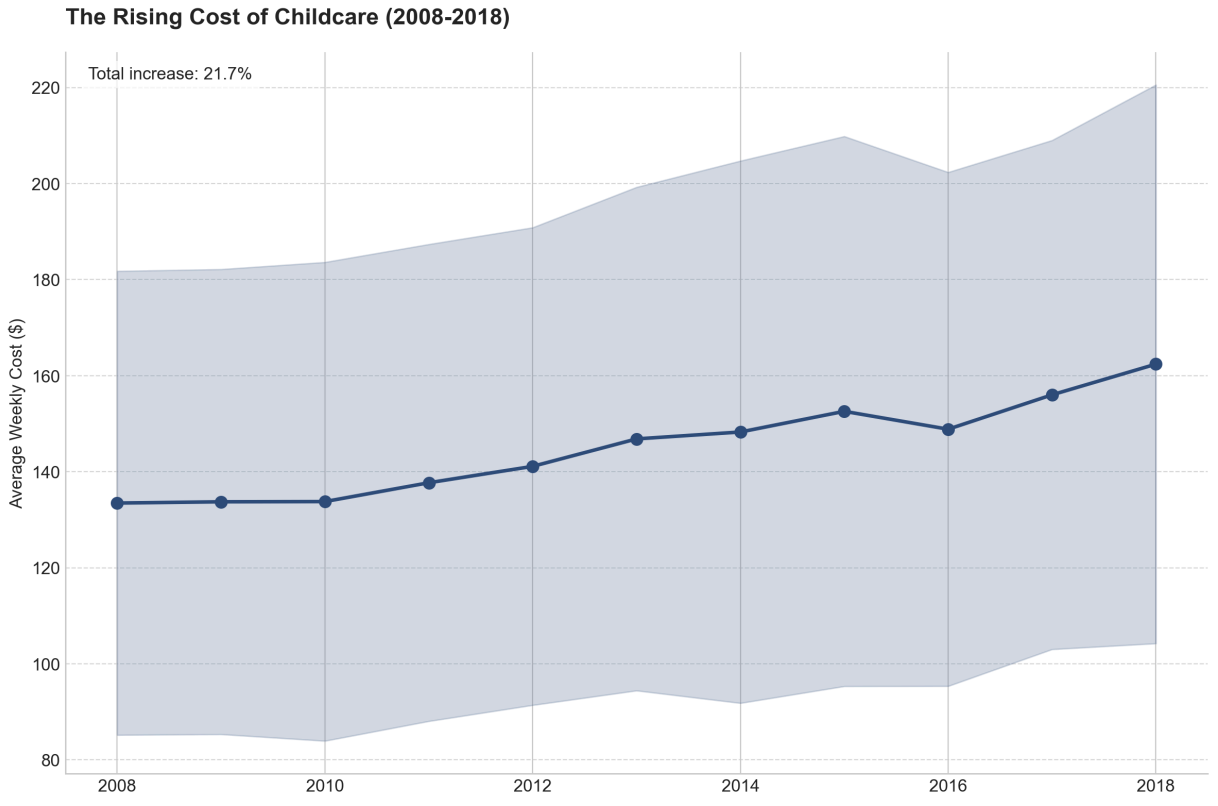


The Time Factor: Evolution of Childcare Costs

Tracking how costs have changed over time reveals the growing economic pressure on families.

In [9]:

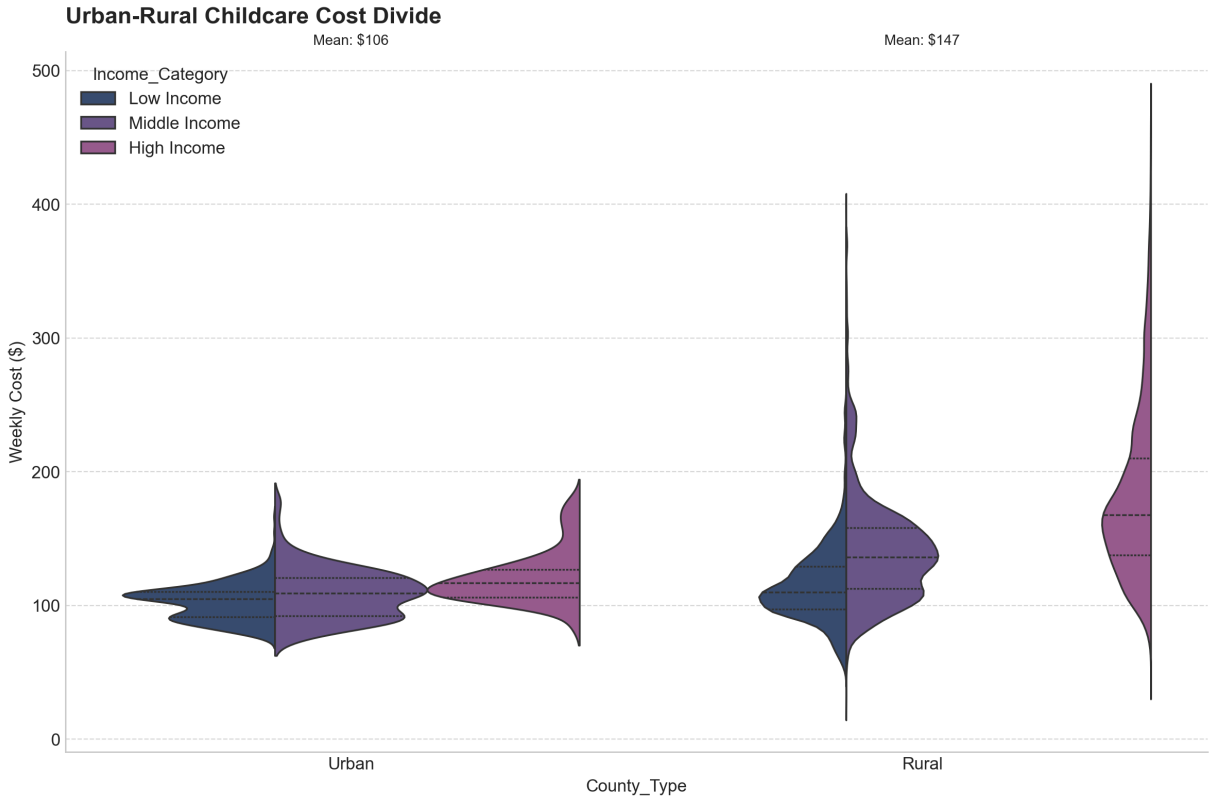
```
# Create an enhanced time series visualization
fig, ax = plt.subplots(figsize=(12, 8))
fig.patch.set_facecolor('white')

# Create a more sophisticated time series
yearly_avg = df.groupby('StudyYear')['MCInfant'].mean()
yearly_std = df.groupby('StudyYear')['MCInfant'].std()

# Plot with confidence interval
ax.fill_between(yearly_avg.index,
                yearly_avg - yearly_std,
                yearly_avg + yearly_std,
                alpha=0.2, color=colors[0])
ax.plot(yearly_avg.index, yearly_avg,
        color=colors[0], linewidth=2.5,
        marker='o', markersize=8)

style_timeseries(ax, 'The Rising Cost of Childcare (2008-2018)',
                'Average Weekly Cost ($)')

# Add percentage change annotation
pct_change = ((yearly_avg.iloc[-1] - yearly_avg.iloc[0]) / yearly_avg.iloc[0
ax.text(0.02, 0.98, f'Total increase: {pct_change:.1f}%',
        transform=ax.transAxes, fontsize=12,
        bbox=dict(facecolor='white', alpha=0.8, edgecolor='none'),
        va='top', ha='left')

plt.tight_layout()
save_figure(fig, 'cost_trends.png')
```



Urban-Rural Divide: Geographic Disparities in Childcare Access

Examining how childcare costs differ between urban and rural areas reveals important accessibility gaps.

In [10]:
```
# Create enhanced urban-rural comparison
fig, ax = plt.subplots(figsize=(12, 8))
fig.patch.set_facecolor('white')

# Create a more sophisticated violin plot
sns.violinplot(data=df, x='County_Type', y='MCInfant',
               hue='Income_Category', split=True,
               inner='quartile', palette=colors[:3])

style_distribution(ax, 'Urban-Rural Childcare Cost Divide')
ax.set_ylabel('Weekly Cost ($)', fontsize=12)

# Add statistical annotations
for county_type in ['Urban', 'Rural']:
    mean_val = df[df['County_Type'] == county_type]['MCInfant'].mean()
    ax.text(0 if county_type == 'Urban' else 1,
            df['MCInfant'].max() * 1.1,
            f'Mean: ${mean_val:,.0f}',
            ha='center', va='bottom', fontsize=10)

plt.tight_layout()
save_figure(fig, 'urban_rural_comparison.png')
```



In [11]:
```
print("Analysis complete! The visualizations and report have been generated.
print(f"\nFigures have been saved in: {figures_dir}")
print("Note: All visualizations include data from the 50 states and DC.")
```

```
Analysis complete! The visualizations and report have been generated.

Figures have been saved in: /Users/komalshahid/Desktop/Bellevue University/D
SC640/final-project/milestones/milestone1/figures
Note: All visualizations include data from the 50 states and DC.
```