

Final Project

Dylan Xia Kevin Yao Yuqing Wen

2024-12-01

- Partner 1 : Dylan Xia; dizhexia
- Partner 2 : Kevin Yao; qiyin
- Partner 3 : Yuqing Wen, wyuqing

```
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
from shapely.geometry import shape
import requests
import altair as alt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
alt.renderers.enable("png")
```

```
RendererRegistry.enable('png')
```

Step 1: Plot data based on Unemployment Rate

Unemployment Rate General Trend (2011~2020)

```
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Load the unemployment data
data_path = "state_yearly_unemployment_rate_with_state_name.csv"
unemployment_data = pd.read_csv(data_path)
```

```

# Step 2: Calculate the average unemployment rate by year
# Assuming the dataset contains columns: 'year', 'unemployment_rate'
yearly_avg_unemployment = (
    unemployment_data.groupby('year')['unemployment_rate']
    .mean()
    .reset_index()
)

# Convert unemployment rate to percentage
yearly_avg_unemployment['unemployment_rate'] =
    yearly_avg_unemployment['unemployment_rate'] * 100

# Step 3: Plot the line chart
plt.figure(figsize=(12, 6))

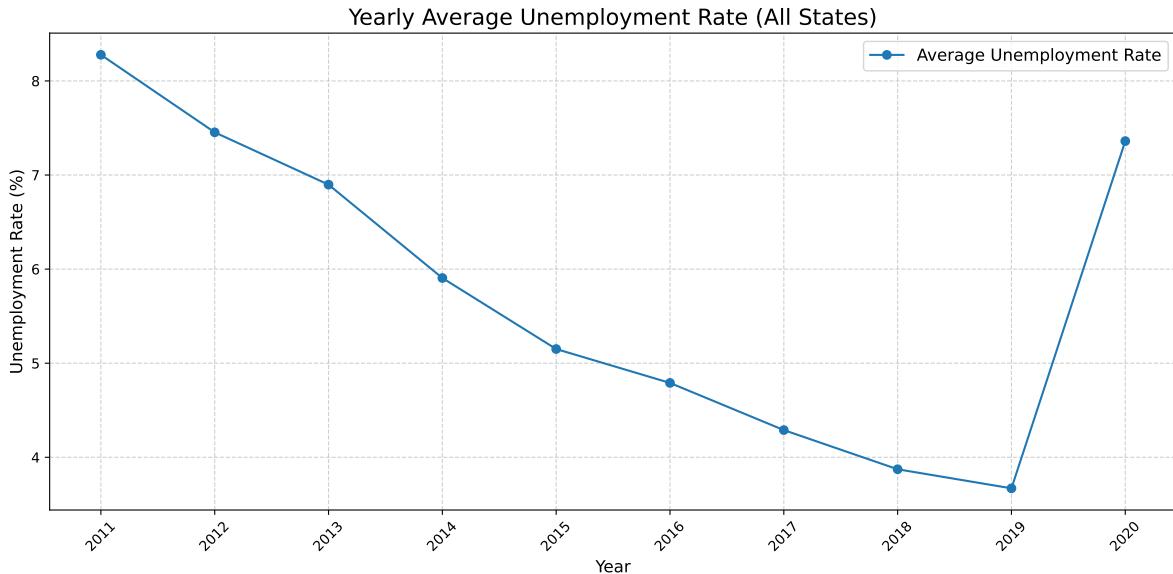
plt.plot(
    yearly_avg_unemployment['year'],
    yearly_avg_unemployment['unemployment_rate'],
    marker='o',
    linestyle='--',
    label='Average Unemployment Rate'
)

# Add labels and title
plt.xlabel('Year', fontsize=12)
plt.ylabel('Unemployment Rate (%)', fontsize=12)
plt.title('Yearly Average Unemployment Rate (All States)', fontsize=16)

# Customize grid and legend
plt.grid(visible=True, linestyle='--', alpha=0.6)
plt.legend(fontsize=12)
plt.xticks(yearly_avg_unemployment['year'], rotation=45)
plt.tight_layout()

# Show the plot
plt.show()

```



Unemployment Rate (2011~2015) & Unemployment Rate (2016~2020), Map by State

```

import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
from shapely.geometry import shape
import requests

# Step 1: Load the unemployment data
data_path = "state_yearly_unemployment_rate_with_state_name.csv"
unemployment_data = pd.read_csv(data_path)

# Calculate average unemployment rates for 2011-2015
unemployment_avg_2011_2015 = (
    unemployment_data[(unemployment_data['year'] >= 2011) &
    (unemployment_data['year'] <= 2015)]
    .groupby('state_name')['unemployment_rate']
    .mean()
    .reset_index()
)
unemployment_avg_2011_2015['unemployment_rate'] =
    unemployment_avg_2011_2015['unemployment_rate'] * 100

```

```

# Calculate average unemployment rates for 2016–2019
unemployment_avg_2016_2019 = (
    unemployment_data[(unemployment_data['year'] >= 2016) &
    (unemployment_data['year'] <= 2019)]
    .groupby('state_name')['unemployment_rate']
    .mean()
    .reset_index()
)
unemployment_avg_2016_2019['unemployment_rate'] =
    ↪ unemployment_avg_2016_2019['unemployment_rate'] * 100

# Find global vmin and vmax for consistent coloring
all_data = pd.concat([unemployment_avg_2011_2015,
    ↪ unemployment_avg_2016_2019])
vmin = all_data['unemployment_rate'].min()
vmax = all_data['unemployment_rate'].max()

# Load GeoJSON data
shape_url = 'https://data.ojp.usdoj.gov/resource/5fdt-n5ne.json'
response = requests.get(shape_url)

if response.status_code == 200:
    geo_data = response.json()
else:
    raise ValueError(f"Failed to fetch GeoJSON data. HTTP Status Code:
        ↪ {response.status_code}")

geometries = [shape(feature["the_geom"]) for feature in geo_data]
properties = [{key: feature[key] for key in feature if key != "the_geom"} for
    ↪ feature in geo_data]
shape_data = gpd.GeoDataFrame(properties, geometry=geometries)

# Function to plot a map with specified data
def plot_map(data, title):
    data['state_name'] = data['state_name'].str.strip()
    merged_data = shape_data.merge(data, left_on='state',
    ↪ right_on='state_name', how='left')
    merged_data['unemployment_rate'] =
    ↪ merged_data['unemployment_rate'].fillna(0)

    # Plot map

```

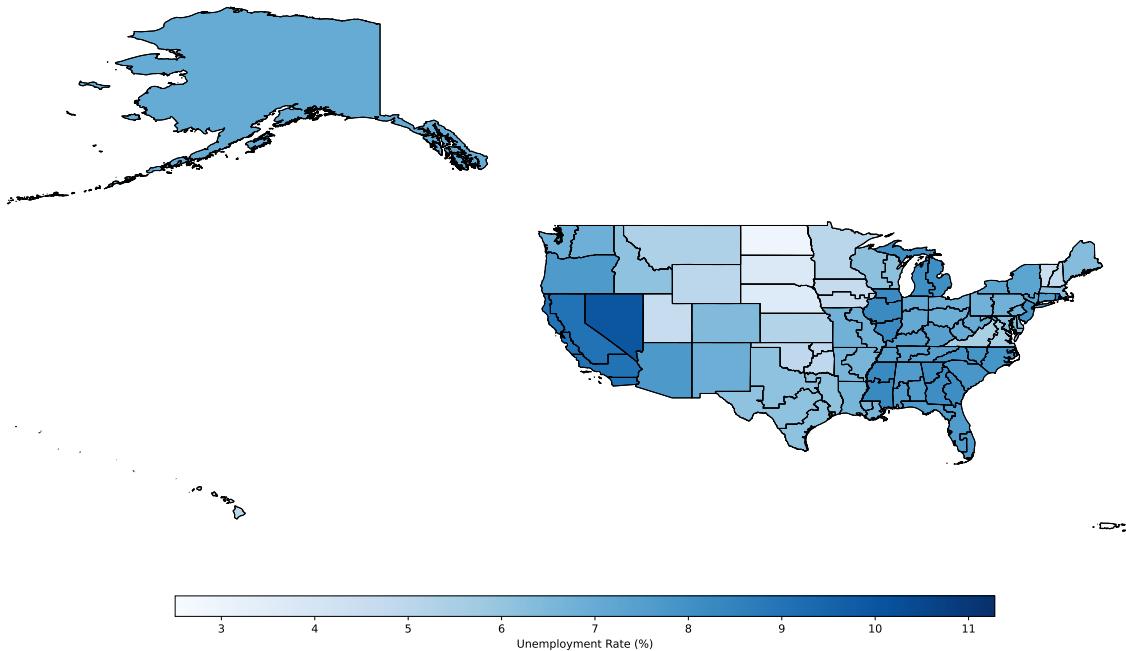
```

fig, ax = plt.subplots(1, 1, figsize=(15, 10))
merged_data.plot(
    column='unemployment_rate',
    cmap='Blues',
    linewidth=0.8,
    ax=ax,
    edgecolor='black',
    legend=True,
    legend_kwds={
        'shrink': 0.7,
        'orientation': "horizontal",
        'pad': 0.05,
        'aspect': 40,
        'label': "Unemployment Rate (%)"
    },
    vmin=vmin, # Set global min
    vmax=vmax # Set global max
)
merged_data.boundary.plot(ax=ax, linewidth=0.8, color="black")
ax.set_xlim([-180, -60]) # Extends to cover Alaska and Hawaii
ax.set_ylim([15, 72]) # Adjusted for both Hawaii and Alaska latitudes
ax.set_title(title, fontsize=16)
ax.set_axis_off()
plt.tight_layout()
plt.show()

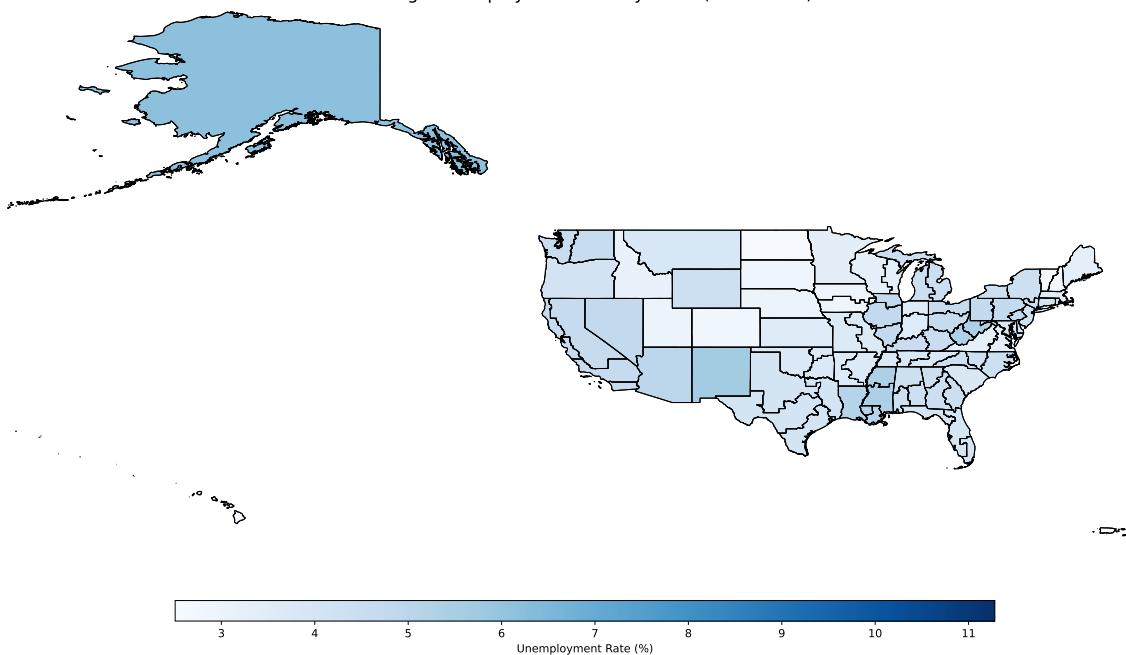
# Plot maps for 2011-2015 and 2016-2019
plot_map(unemployment_avg_2011_2015, 'Average Unemployment Rate by State
↪ (2011-2015)')
plot_map(unemployment_avg_2016_2019, 'Average Unemployment Rate by State
↪ (2016-2019)')

```

Average Unemployment Rate by State (2011-2015)



Average Unemployment Rate by State (2016-2019)



Unemployment Rate (Difference), Map by State

```
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
from shapely.geometry import shape
import requests

# Step 1: Load the unemployment data
data_path = "state_yearly_unemployment_rate_with_state_name.csv"
unemployment_data = pd.read_csv(data_path)

# Step 2: Calculate average unemployment rates for 2011-2015 and 2016-2019 by
# state
unemployment_avg_2011_2015 = (
    unemployment_data[(unemployment_data['year'] >= 2011) &
    (unemployment_data['year'] <= 2015)]
    .groupby('state_name')['unemployment_rate']
    .mean()
    .reset_index()
)
unemployment_avg_2016_2019 = (
    unemployment_data[(unemployment_data['year'] >= 2016) &
    (unemployment_data['year'] <= 2019)]
    .groupby('state_name')['unemployment_rate']
    .mean()
    .reset_index()
)

# Convert to percentage
unemployment_avg_2011_2015['unemployment_rate'] =
    unemployment_avg_2011_2015['unemployment_rate'] * 100
unemployment_avg_2016_2019['unemployment_rate'] =
    unemployment_avg_2016_2019['unemployment_rate'] * 100

# Step 3: Calculate the difference between 2016-2019 and 2011-2015
unemployment_diff = pd.merge(
    unemployment_avg_2016_2019,
    unemployment_avg_2011_2015,
    on='state_name',
    suffixes=('_2016_2019', '_2011_2015')
```

```

)
unemployment_diff['rate_difference'] = (
    unemployment_diff['unemployment_rate_2016_2019'] -
    unemployment_diff['unemployment_rate_2011_2015']
)

# Step 4: Load the JSON data from the API endpoint for state boundaries
shape_url = 'https://data.ojp.usdoj.gov/resource/5fdt-n5ne.json'
response = requests.get(shape_url)

if response.status_code == 200:
    geo_data = response.json()
else:
    raise ValueError(f"Failed to fetch GeoJSON data. HTTP Status Code:
        {response.status_code}")

# Step 5: Convert JSON data to a GeoDataFrame
geometries = [shape(feature["the_geom"]) for feature in geo_data]
properties = [{key: feature[key] for key in feature if key != "the_geom"} for
    feature in geo_data]
shape_data = gpd.GeoDataFrame(properties, geometry=geometries)

# Step 6: Merge shape data with unemployment difference data
unemployment_diff['state_name'] = unemployment_diff['state_name'].str.strip()
merged_data = shape_data.merge(unemployment_diff, left_on='state',
    right_on='state_name', how='left')

# Fill missing values for states with no data
merged_data['rate_difference'] = merged_data['rate_difference'].fillna(0)

# Step 7: Plot the unemployment difference map
fig, ax = plt.subplots(1, 1, figsize=(15, 10))

# Plot unemployment difference using a diverging colormap
merged_data.plot(
    column='rate_difference',
    cmap='RdYlGn_r', # Red-White-Green reversed color map
    linewidth=0.8,
    ax=ax,
    edgecolor='black',
    legend=True,
    legend_kwds={
```

```

        'shrink': 0.7,
        'orientation': "horizontal",
        'pad': 0.05,
        'aspect': 40,
        'label': "Unemployment Rate Difference (%)"
    }
)

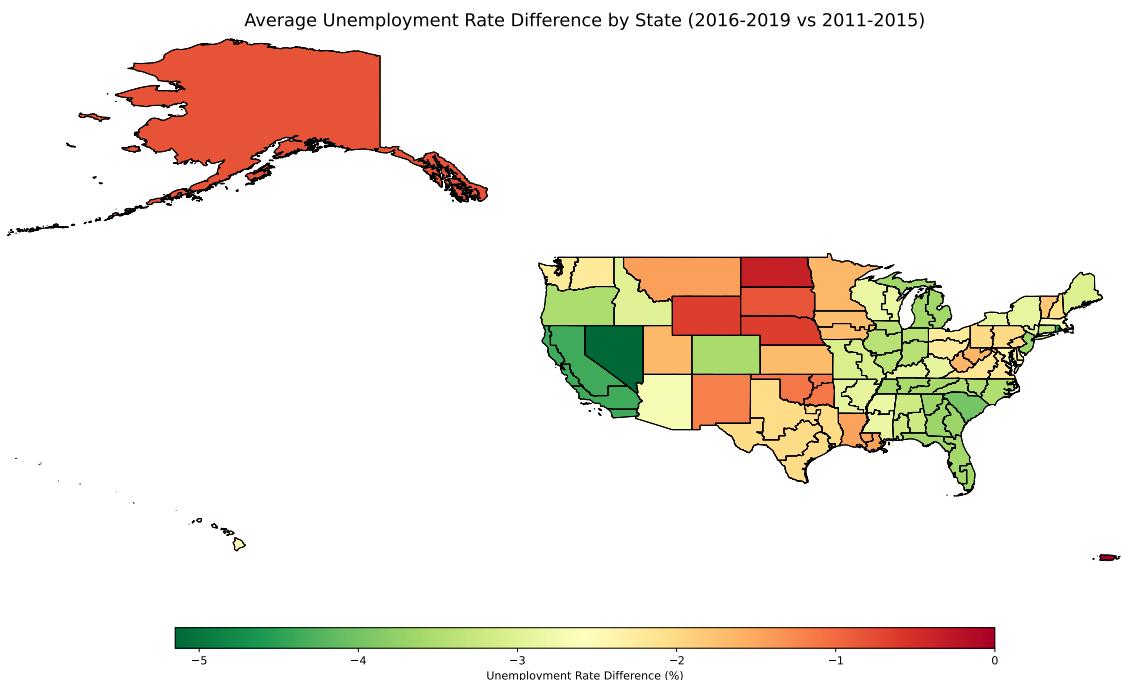
# Add state boundaries
merged_data.boundary.plot(ax=ax, linewidth=0.8, color="black")

# Adjust map extent to include Alaska and Hawaii
ax.set_xlim([-180, -60]) # Extends to cover Alaska and Hawaii
ax.set_ylim([15, 72]) # Adjusted for both Hawaii and Alaska latitudes

# Customize the plot further for better visualization
ax.set_title('Average Unemployment Rate Difference by State (2016-2019 vs
    ↵ 2011-2015)', fontsize=16)
ax.set_axis_off()
plt.tight_layout()

plt.show()

```



Step 2: Plot data based on CPI

CPI Rate General Trend (2011~2020)

```
# Load the CPI data from the Excel file
file_path = 'CPI_urban_2011-2020_2.0.xlsx'
cpi_data = pd.ExcelFile(file_path)

# Parse the relevant sheet
cpi_sheet = cpi_data.parse('BLS Data Series')

# Step 1: Clean the CPI data
# Skip the initial rows to extract relevant data
cpi_cleaned = cpi_sheet.iloc[3:]

# Rename columns for clarity
cpi_cleaned.columns = ['Series ID', 'Area Description'] + [f'Annual_{year}' for year in range(2011, 2021)]

# Keep only relevant columns (Area Description and CPI values)
cpi_cleaned = cpi_cleaned[['Area Description'] + [f'Annual_{year}' for year in range(2011, 2021)]] 

# Drop rows with missing Area Description (e.g., aggregated regions or
# unrelated data)
cpi_cleaned = cpi_cleaned.dropna(subset=['Area Description'])

# Step 2: Calculate the average CPI for all regions for each year
average_cpi_by_year = cpi_cleaned[[f'Annual_{year}' for year in range(2011, 2021)]].mean()

# Prepare data for plotting
years = list(range(2011, 2021))
average_cpi = average_cpi_by_year.values

# Step 3: Plot the line chart
plt.figure(figsize=(10, 6))
plt.plot(years, average_cpi, marker='o', linestyle='-', color='blue',
         label='Average CPI (All Regions)')

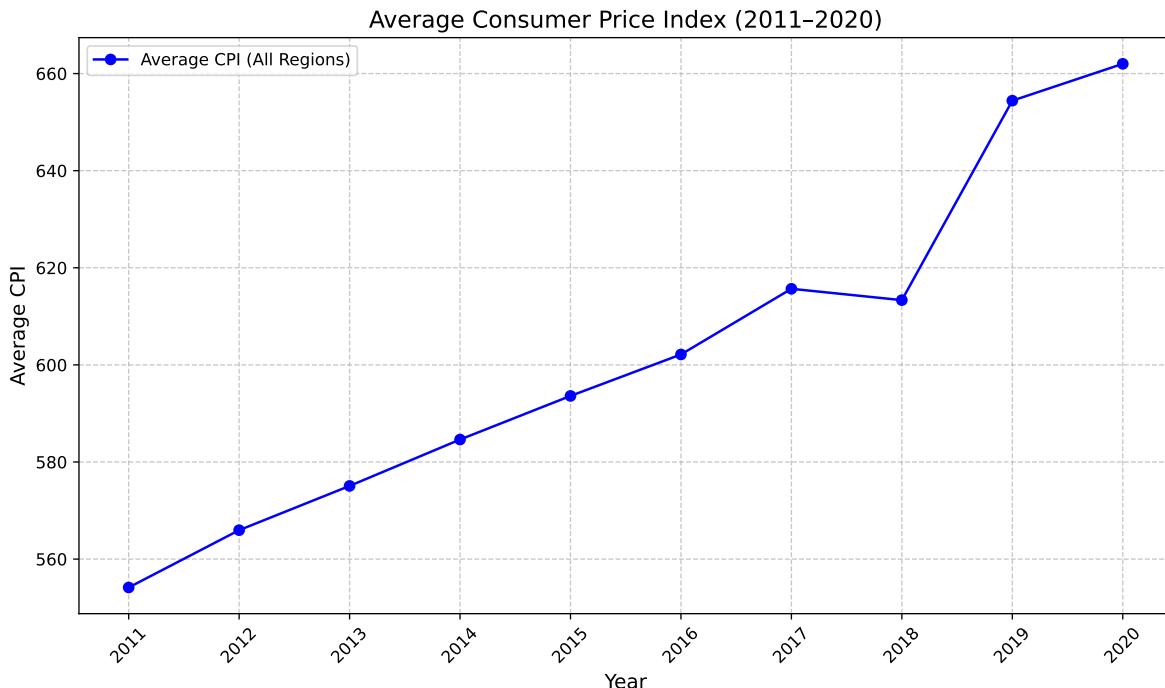
# Add labels, title, and legend
```

```

plt.xlabel('Year', fontsize=12)
plt.ylabel('Average CPI', fontsize=12)
plt.title('Average Consumer Price Index (2011-2020)', fontsize=14)
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend()
plt.xticks(years, rotation=45)
plt.tight_layout()

# Show the plot
plt.show()

```



CPI Rate Scatter (Self-Regression)

```

# Load the CPI data
file_path = 'CPI_urban_2011-2020_2.0.xlsx' # Update this to your local file
# path
cpi_data = pd.ExcelFile(file_path)
cpi_sheet = cpi_data.parse('BLS Data Series')

```

```

# Clean the data
cpi_cleaned = cpi_sheet.iloc[3:]
cpi_cleaned.columns = ['Series ID', 'Area Description'] + [f'Annual_{year}' for year in range(2011, 2021)]
cpi_cleaned = cpi_cleaned[['Area Description'] + [f'Annual_{year}' for year in range(2011, 2021)]]
cpi_cleaned = cpi_cleaned.dropna(subset=['Area Description'])

# Convert annual columns to numeric
for year in range(2011, 2021):
    cpi_cleaned[f'Annual_{year}'] = pd.to_numeric(cpi_cleaned[f'Annual_{year}'], errors='coerce')

# Calculate average CPI for 2011–2015 and 2016–2019
cpi_cleaned['CPI_2011_2015'] = cpi_cleaned[[f'Annual_{year}' for year in range(2011, 2016)]].mean(axis=1)
cpi_cleaned['CPI_2016_2019'] = cpi_cleaned[[f'Annual_{year}' for year in range(2016, 2020)]].mean(axis=1)

# Drop rows where averages could not be computed
cpi_cleaned = cpi_cleaned.dropna(subset=['CPI_2011_2015', 'CPI_2016_2019'])

# Calculate percentage change
cpi_cleaned['CPI_Percent_Change'] = ((cpi_cleaned['CPI_2016_2019'] - cpi_cleaned['CPI_2011_2015']) / cpi_cleaned['CPI_2011_2015']) * 100

# Identify top 5 and bottom 5 areas
top5_changes = cpi_cleaned.nlargest(5, 'CPI_Percent_Change')
bottom5_changes = cpi_cleaned.nsmallest(5, 'CPI_Percent_Change')

# Add jittered labels for visualization
top5_changes['Label'] = top5_changes['Area Description']
top5_changes['Jitter_X'] = top5_changes['CPI_2011_2015'] + 10
top5_changes['Jitter_Y'] = top5_changes['CPI_2016_2019'] + 20

bottom5_changes['Label'] = bottom5_changes['Area Description']
bottom5_changes['Jitter_X'] = bottom5_changes['CPI_2011_2015'] + 10
bottom5_changes['Jitter_Y'] = bottom5_changes['CPI_2016_2019'] - 20

# Create diagonal y=x line
diagonal_data = pd.DataFrame({
    'x': [cpi_cleaned['CPI_2011_2015'].min(),
          cpi_cleaned['CPI_2011_2015'].max()],

```

```

'y': [cpi_cleaned['CPI_2011_2015'].min(),
      cpi_cleaned['CPI_2011_2015'].max()]
})

diagonal_line = alt.Chart(diagonal_data).mark_line(strokeDash=[5, 5],
    color='gray').encode(
    x='x',
    y='y'
)

# Scatter plot for all areas
scatter = alt.Chart(cpi_cleaned).mark_circle(size=60, color='blue').encode(
    x=alt.X('CPI_2011_2015', title='Average CPI (2011-2015)'),
    y=alt.Y('CPI_2016_2019', title='Average CPI (2016-2019)'),
    tooltip=['Area Description', 'CPI_2011_2015', 'CPI_2016_2019',
    'CPI_Percent_Change']
)

# Highlight top 5 areas
top5 = alt.Chart(top5_changes).mark_circle(size=100, color='red').encode(
    x='CPI_2011_2015',
    y='CPI_2016_2019',
    tooltip=['Area Description', 'CPI_2011_2015', 'CPI_2016_2019',
    'CPI_Percent_Change']
)

top5_labels = alt.Chart(top5_changes).mark_text(align='left', fontSize=12,
    color='red').encode(
    x='Jitter_X',
    y='Jitter_Y',
    text='Label'
)

# Highlight bottom 5 areas
bottom5 = alt.Chart(bottom5_changes).mark_circle(size=100,
    color='green').encode(
    x='CPI_2011_2015',
    y='CPI_2016_2019',
    tooltip=['Area Description', 'CPI_2011_2015', 'CPI_2016_2019',
    'CPI_Percent_Change']
)

```

```

bottom5_labels = alt.Chart(bottom5_changes).mark_text(align='left',
    ←  fontSize=12, color='green').encode(
    x='Jitter_X',
    y='Jitter_Y',
    text='Label'
)

# Tables for top 5 and bottom 5 areas
top5_table = alt.Chart(top5_changes).mark_text(align='left', fontSize=12,
    ←  color='red').encode(
    y=alt.Y('row_number:0', axis=None),
    text='Label:N'
).transform_window(
    row_number='row_number()'
).properties(
    title='Top 5 Areas by Percentage Change',
    width=300
)

bottom5_table = alt.Chart(bottom5_changes).mark_text(align='left',
    ←  fontSize=12, color='green').encode(
    y=alt.Y('row_number:0', axis=None),
    text='Label:N'
).transform_window(
    row_number='row_number()'
).properties(
    title='Bottom 5 Areas by Percentage Change',
    width=300
)

# Combine scatter plot with updated tables
scatter_chart = (scatter + top5 + top5_labels + bottom5 + bottom5_labels +
    ←  diagonal_line).properties(
    title='Scatter Plot and Highlights of CPI (2011-2015 vs. 2016-2019)',
    width=800,
    height=500
)

final_chart = alt.hconcat(
    scatter_chart,
    alt.vconcat(top5_table, bottom5_table).properties(title='Top and Bottom 5
    ←  Areas (Colored)')
)

```