de-mini-state-wise-1

October 24, 2024

1. Acquiring the Dataset:

The first step involved gathering data from official sources such as government websites, which provided us with state-wise budget allocation data for different financial years (2013-14 and 2024-25 in this case). This is crucial in data engineering as working with real-world data often starts with data acquisition from trustworthy sources. For this project, the data was likely sourced in a tabular format (like Excel or CSV files), which is common in data analysis tasks.

[ ]:
```
!pip install seaborn
!pip install matplotlib
```

```
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-
packages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in
/usr/local/lib/python3.10/dist-packages (from seaborn) (1.26.4)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dist-
packages (from seaborn) (2.2.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in
/usr/local/lib/python3.10/dist-packages (from seaborn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(4.54.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(1.4.7)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in
```

```
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas>=1.2->seaborn) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-
packages (from pandas>=1.2->seaborn) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-
packages (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (4.54.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.7)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-
packages (from matplotlib) (1.26.4)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

[ ]:
```python
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving state.xlsx to state (2).xlsx
```

2. Data Preprocessing:
   Data preprocessing is an essential phase in data engineering, which includes steps such as data cleaning, transformation, integration, and reduction.
   Data Cleaning: Data is often incomplete, noisy, or inconsistent. In this project, the dataset had missing values (NaN), which were handled by either filling missing values with 0 or removing invalid data. Cleaning ensures that the dataset is consistent and reliable, avoiding errors during analysis and visualization.
   Data Transformation: Data transformation involves converting the data into a more suitable format for analysis. In this case, we set appropriate column headers, ensured correct indexing by state names, and performed type conversions where necessary. This helped us prepare the dataset for further analysis and visualizations.

Data Integration: Since the project involved comparing different financial years, integration was necessary to combine the data into a unified format. We combined the 2024-25 and 2013-14 data for better comparative analysis.
Data Reduction:

Data reduction is the process of reducing the volume but maintaining the integrity of the dataset. Although we worked with the full dataset here, we could have applied reduction techniques like aggregating data to a higher level (e.g., grouping smaller states) if needed for simplification.

```
[ ]: import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns

     # Step 1: Load the Excel file
     df = pd.read_excel('state.xlsx')

     # Step 1: Set the correct column names (set the first row as column headers)
     df.columns = df.iloc[0]  # First row becomes header
     df = df.drop(0)  # Drop the first row (now it's part of headers)

     # Step 2: Rename the first column to 'State'
     df = df.rename(columns={df.columns[0]: 'State'})

     # Step 3: Set 'State' as the index
     df = df.set_index('State')

     # Step 4: Convert all columns to numeric (ignore non-numeric errors)
     df = df.apply(pd.to_numeric, errors='coerce')

     # Step 5: Drop any rows or columns that are fully NaN
     df = df.dropna(how='all', axis=1)  # Drop columns with all NaN
     df = df.dropna(how='all', axis=0)  # Drop rows with all NaN

     # Check the cleaned data
     df.head()
```

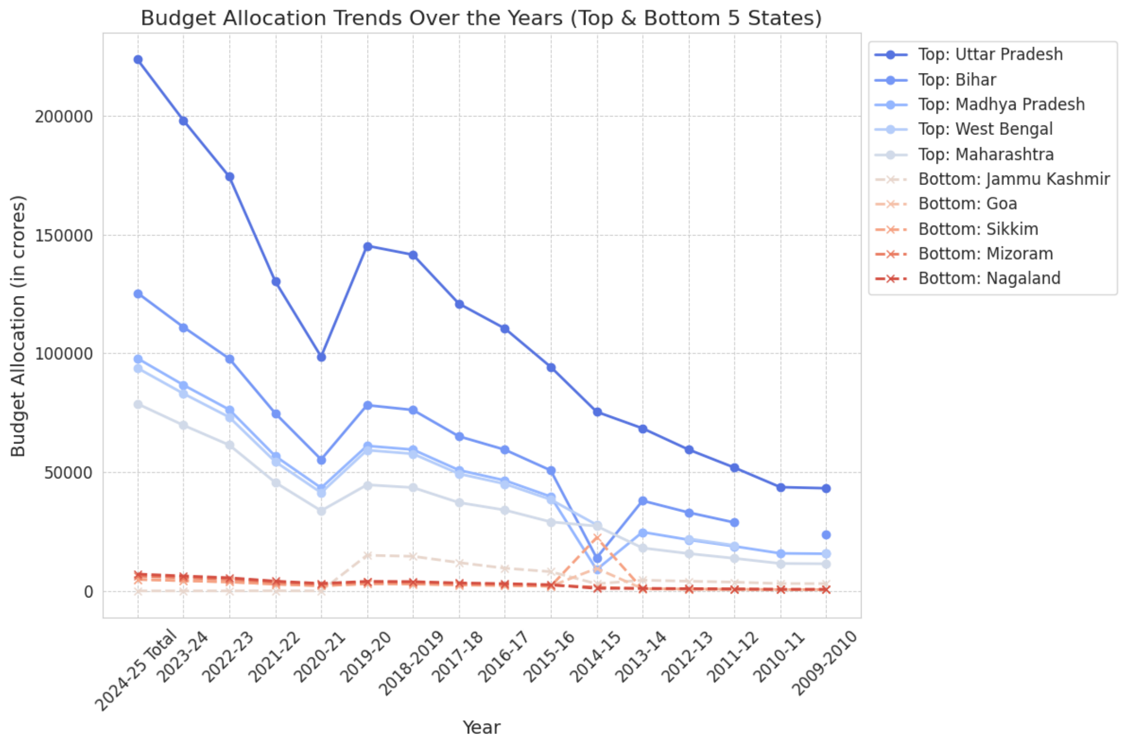
```
[ ]: 0                   2024-25 Total    2023-24   2022-23   2021-22   2020-21  \
     State
     Andhra Pradesh           50474.64   44698.99  39338.27  30356.31  22610.02
     Arunachal Pradesh        21913.50   19405.97  17081.64  13062.50   9679.78
     Assam                    39012.77   34548.57  30410.56  23266.50  17220.13
     Bihar                   125444.52  111089.98  97784.32  74789.06  55334.34
     Chhattisgarh             42492.49   37630.13  33123.02  25331.43  18798.61

     0                    2019-20  2018-2019   2017-18   2016-17   2015-16  \
     State
     Andhra Pradesh      34833.18   33929.84  29001.25  26523.99  22637.97
     Arunachal Pradesh   11085.12   10798.47   9238.79   8471.76   7231.74
```

```
Assam                26790.40   26095.30  22301.52  20388.40  17400.88
Bihar                78202.69   76172.37  65083.19  59462.65  50747.58
Chhattisgarh         24921.31   24275.49  20754.83  18995.65  16213.35

0                     2014-15    2013-14   2012-13   2011-12   2010-11  2009-2010
State
Andhra Pradesh       16838.77   24132.36  20986.60  18304.09  15403.17   15236.74
Arunachal Pradesh     9749.36    1140.38    991.84    865.16    728.05     720.18
Assam                 1256.36   12620.75  10975.63   9572.77   8055.64    7968.61
Bihar                13905.02   37977.32  33026.93  28805.50       NaN   23978.38
Chhattisgarh         41841.93    8592.52   7472.46   6517.35   5484.44    5425.19
```

3. Visualization:
   Once the data was cleaned and transformed, the next step involved generating various visualizations to gain insights. Visualization is an essential part of data analysis and engineering as it helps understand the underlying patterns and trends in data.
   Line Graph (Trend over Time):
   The line graph was used to plot budget allocations for the top states from 2013-14 to 2024-25. This visualization is highly useful in spotting trends over time and understanding how the allocation changes year over year.
   Usefulness: It clearly shows which states have seen significant increases or decreases in allocation over the years, helping policy analysts and decision-makers evaluate funding shifts.

```
[ ]: import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns

     # Increase figure size
     plt.figure(figsize=(12, 8))

     # Define color palette
     sns.set_palette('coolwarm', n_colors=10)

     # Select top 5 and bottom 5 states by 2024-25 Total allocation
     top_states = df.sort_values(by='2024-25 Total', ascending=False).head(5).index
     bottom_states = df.sort_values(by='2024-25 Total', ascending=True).head(5).index

     # Plot the trends for each of the top 5 states
     for state in top_states:
         plt.plot(df.columns, df.loc[state], label=f'Top: {state}', linewidth=2,␣
      ↪linestyle='-', marker='o')

     # Plot the trends for each of the bottom 5 states
     for state in bottom_states:
         plt.plot(df.columns, df.loc[state], label=f'Bottom: {state}', linewidth=2,␣
      ↪linestyle='--', marker='x')
```

```
# Add title and labels with custom font sizes
plt.title('Budget Allocation Trends Over the Years (Top & Bottom 5 States)',␣
  ↪fontsize=16)
plt.xlabel('Year', fontsize=14)
plt.ylabel('Budget Allocation (in crores)', fontsize=14)

# Customize ticks and rotation
plt.xticks(rotation=45, fontsize=12)
plt.yticks(fontsize=12)

# Add a grid for better readability
plt.grid(True, which='both', linestyle='--', linewidth=0.7)

# Add legend with custom positioning and font size
plt.legend(loc='upper left', bbox_to_anchor=(1, 1), fontsize=12)

# Show the plot
plt.tight_layout()
plt.show()
```



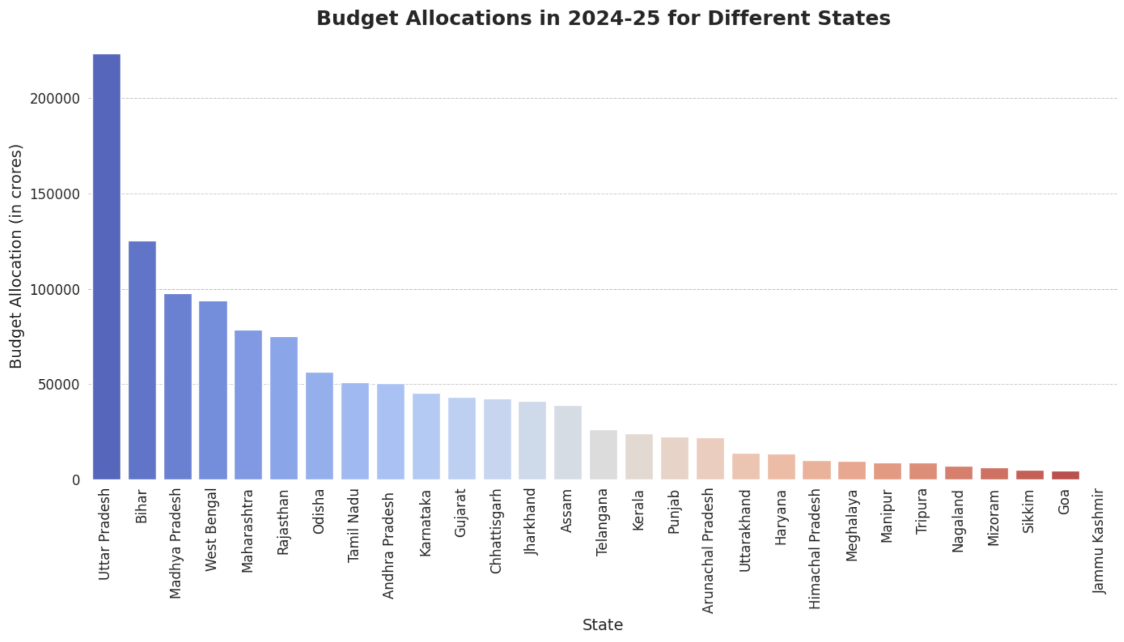
2. Bar Graph (State-wise Budget Comparison for 2024-25):
   Insights:
   Easy comparison: The bar graph clearly highlighted which states received the highest and lowest allocations in 2024-25, making it easy to compare total budget distribution across all

states.
Outlier detection: States with the highest bars stood out, and this indicates potential funding outliers—either large states with high needs or regions that may be receiving disproportionate funding.
Equity assessment: By looking at the relative size of the bars, one can assess the equity in budget distribution, especially between states of similar size or needs. If some states receive far more funding than others with similar population or economic status, this could suggest an imbalance.

[ ]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Increase figure size
plt.figure(figsize=(14, 8))

# Use a coolwarm color palette for a professional look
sns.set_palette('coolwarm')

# Sort states by the 2024-25 Total allocation for the bar chart
sorted_data = df['2024-25 Total'].sort_values(ascending=False)

# Plot the bar chart with enhanced visuals
sns.barplot(x=sorted_data.index, y=sorted_data.values, palette="coolwarm")

# Add title and axis labels with larger font sizes for better readability
plt.title('Budget Allocations in 2024-25 for Different States', fontsize=18,␣
 ↪fontweight='bold')
plt.xlabel('State', fontsize=14)
plt.ylabel('Budget Allocation (in crores)', fontsize=14)

# Rotate x-axis labels for better alignment and readability
plt.xticks(rotation=90, fontsize=12)

# Customize the y-axis ticks for clarity
plt.yticks(fontsize=12)

# Add a grid for better readability, but only on the y-axis for a cleaner look
plt.grid(axis='y', linestyle='--', linewidth=0.7)

# Remove unnecessary chart borders (despine)
sns.despine(left=True, bottom=True)

# Display the chart with tight layout to avoid overlap
plt.tight_layout()

# Show the plot
```

```
plt.show()
```

```
<ipython-input-45-8532923abfe7>:15: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=sorted_data.index, y=sorted_data.values, palette="coolwarm")
```



3. Heatmap (Correlation Analysis for State Allocations):

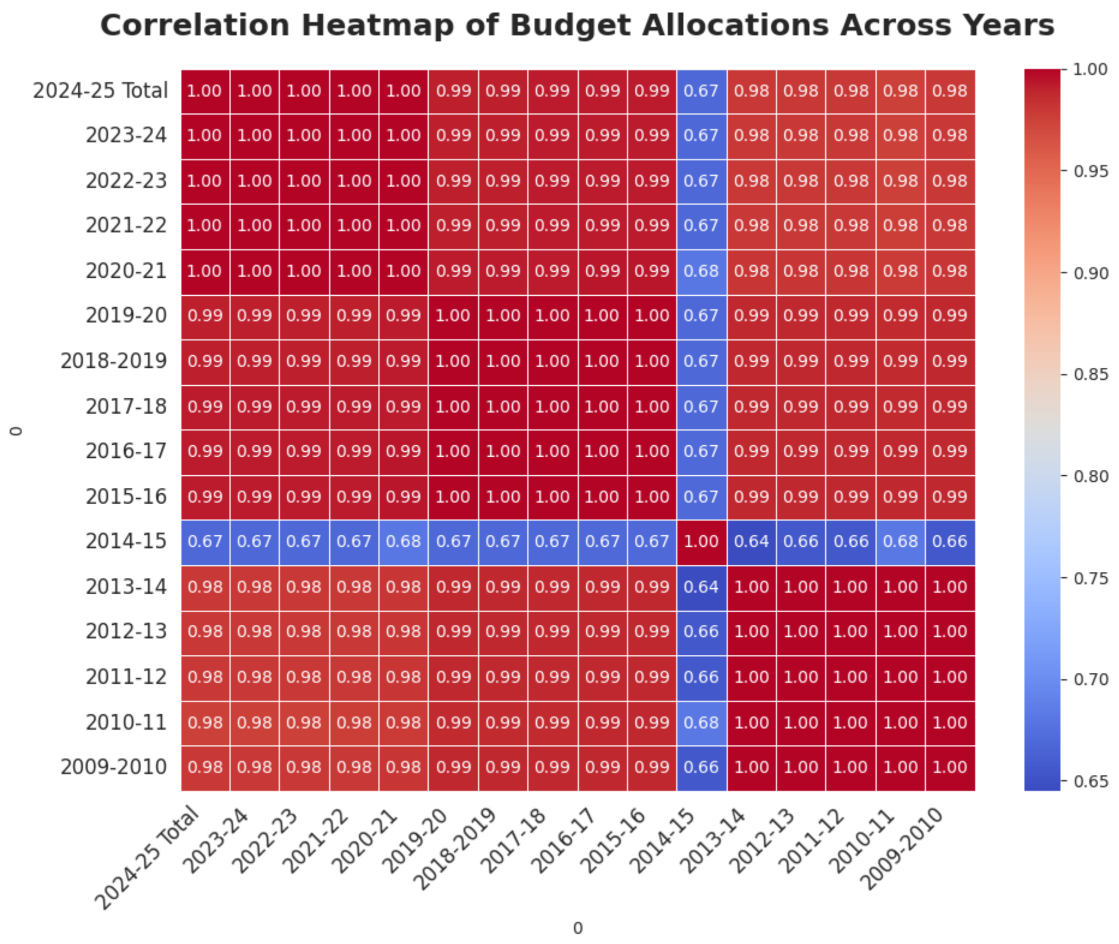
State-wise relationships: The heatmap helped visualize the correlation between budget allocations for different states. A high correlation between two states could indicate that they are being prioritized similarly, possibly because they share common characteristics like geography, economic profile, or political alignment.

Pattern discovery: Correlation analysis can reveal clusters of states that receive similar funding over time, which might suggest a regional development policy where neighboring states are supported similarly.

Budget dependencies: If two states show a strong correlation, it could also suggest that changes in the budget for one state (due to economic conditions or policies) might affect the other in similar ways, providing insights for future resource allocation strategies.

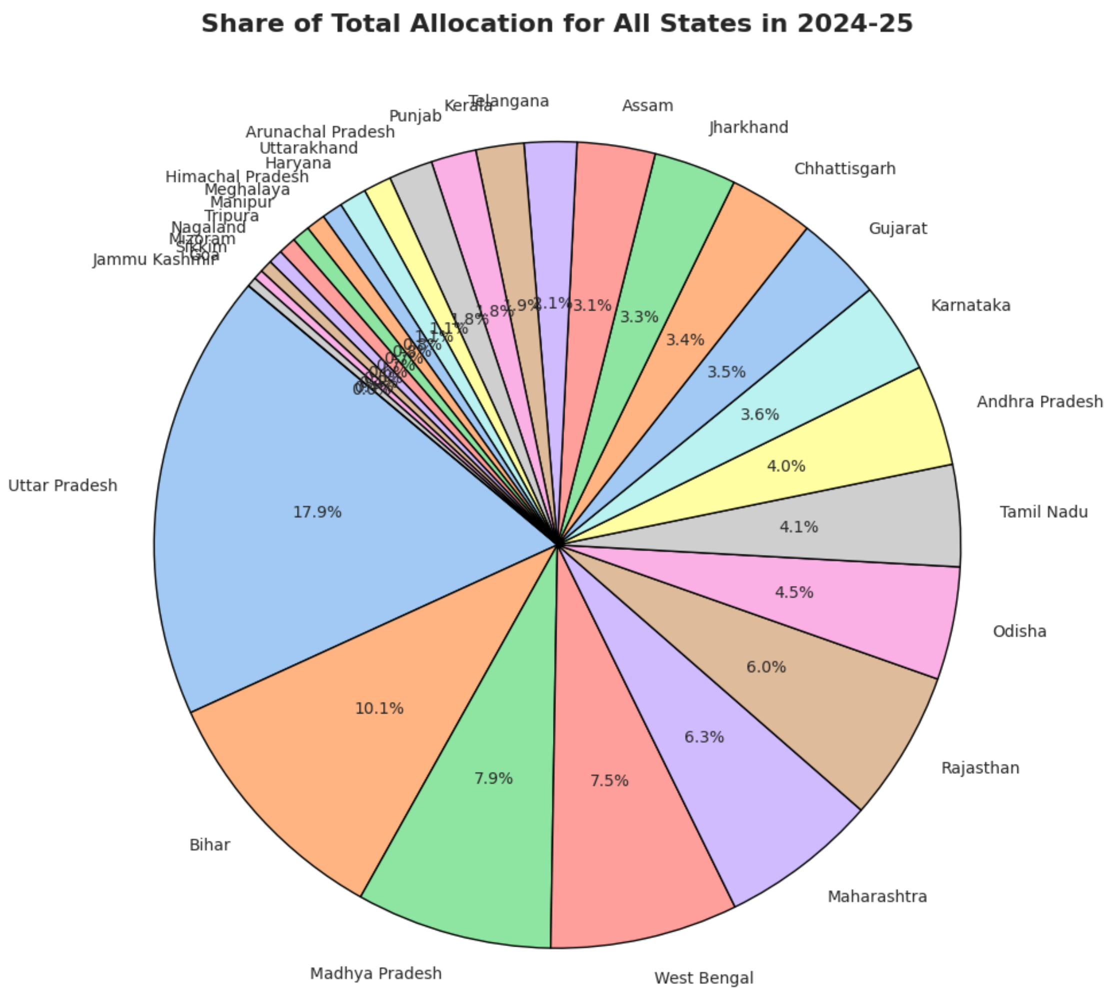
```
[ ]: import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
```

```
# Step 1: Calculate the correlation matrix
correlation_matrix = df.corr()

# Step 2: Set up the figure for the heatmap
plt.figure(figsize=(10, 8))

# Step 3: Generate the heatmap using Seaborn
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5,␣
 ↪fmt=".2f", cbar=True)

# Step 4: Customize the plot for aesthetics
plt.title('Correlation Heatmap of Budget Allocations Across Years',␣
 ↪fontsize=18, fontweight='bold', pad=20)
plt.xticks(rotation=45, ha='right', fontsize=12)
plt.yticks(fontsize=12)

# Step 5: Display the heatmap with a tight layout
plt.tight_layout()
plt.show()
```

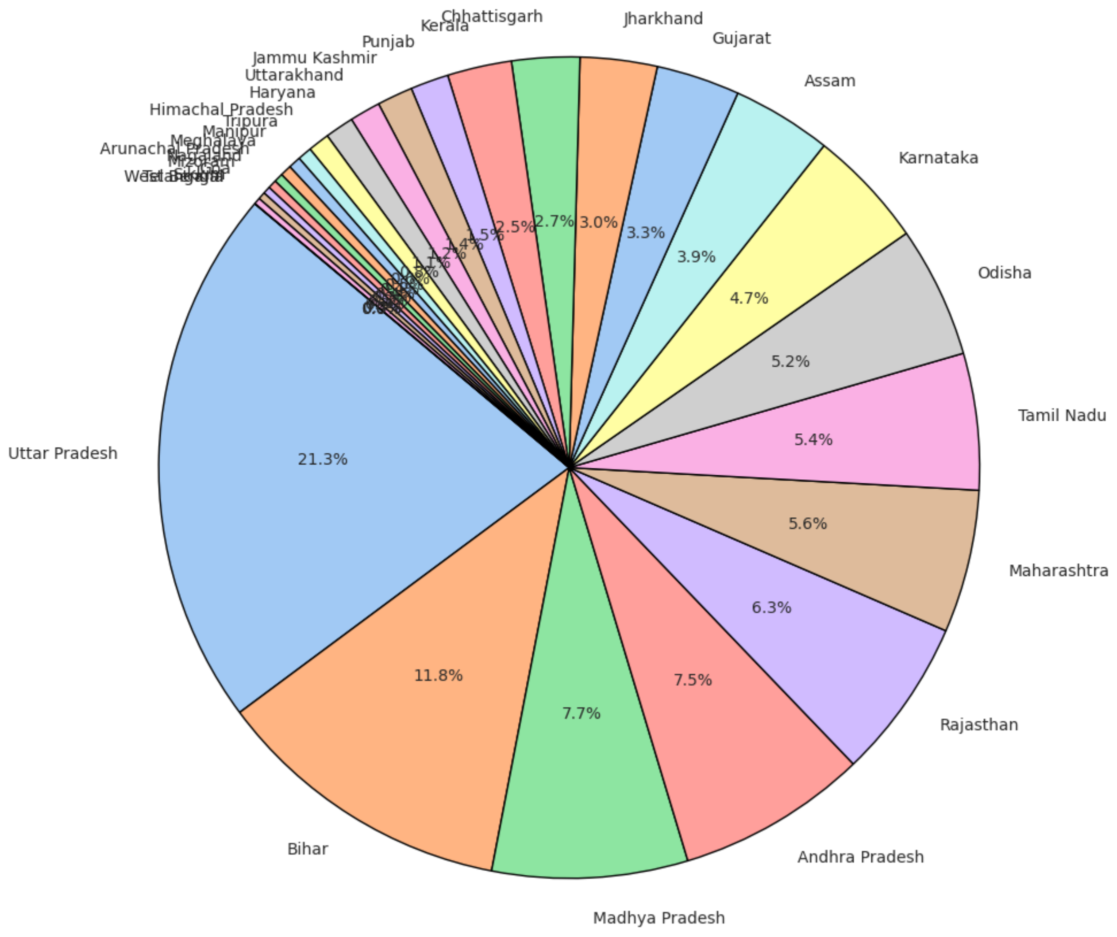


4. Pie Chart(Share of Total Budget for 2024-25 and 2013-14):
   Insights for 2024-25:
   Proportional allocation: The pie chart provided a snapshot of how the total budget was distributed across all states in 2024-25. This helps in understanding which states are receiving the largest portions of the overall budget.
   Identifying major recipients: States with the largest slices of the pie, such as those occupying more than 10-15%, are clearly the biggest recipients of funding. This could highlight regions that are prioritized for economic growth, infrastructure development, or specific government programs.
   Insights for 2013-14:
   Historical comparison: The pie chart for 2013-14 allows us to compare the share of budget allocations with 2024-25. This highlights shifts in priorities over time, showing which states gained or lost funding share over the years.
   State-wise shifts: Comparing the two years' pie charts can also help identify any states that have significantly increased or decreased their share, pointing to political, economic, or policy-driven changes.

[ ]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming 'df' is the cleaned DataFrame with budget allocations
# df = pd.read_excel('state.xlsx')  # Replace with actual file path

# Step 1: Clean the data by filling or dropping NaN values
# Fill NaN with 0 or drop NaNs based on your preference
df['2024-25 Total'] = df['2024-25 Total'].fillna(0)  # Replace NaN with 0 for␣
 ↪2024-25
df['2013-14'] = df['2013-14'].fillna(0)  # Replace NaN with 0 for 2013-14

# Step 2: Pie chart for all states' share in the 2024-25 allocation
plt.figure(figsize=(10, 10))

# All states' allocations in 2024-25
allocation_2024_25 = df['2024-25 Total'].sort_values(ascending=False)

# Plot the pie chart for 2024-25
plt.pie(allocation_2024_25, labels=allocation_2024_25.index, autopct='%1.1f%%',␣
 ↪startangle=140,
        colors=sns.color_palette('pastel'), wedgeprops={'edgecolor': 'black'})

# Add a title
plt.title('Share of Total Allocation for All States in 2024-25', fontsize=16,␣
 ↪fontweight='bold')
plt.tight_layout()
plt.show()

# Step 3: Pie chart for all states' share in the 2013-14 allocation
plt.figure(figsize=(10, 10))

# All states' allocations in 2013-14
allocation_2013_14 = df['2013-14'].sort_values(ascending=False)

# Plot the pie chart for 2013-14
plt.pie(allocation_2013_14, labels=allocation_2013_14.index, autopct='%1.1f%%',␣
↪startangle=140,
        colors=sns.color_palette('pastel'), wedgeprops={'edgecolor': 'black'})

# Add a title
plt.title('Share of Total Allocation for All States in 2013-14', fontsize=16,␣
↪fontweight='bold')
plt.tight_layout()
plt.show()
```


Share of Total Allocation for All States in 2024-25
Uttar Pradesh
17.9%
Bihar
10.1%
Madhya Pradesh
7.9%
West Bengal
7.5%
Maharashtra
6.3%
Rajasthan
6.0%
Odisha
4.5%
Tamil Nadu
4.1%
Andhra Pradesh
4.0%
Karnataka
3.6%
Gujarat
3.5%
Chhattisgarh
3.4%
Jharkhand
3.3%
Assam
3.1%
Telangana
2.1%
Kerala
1.9%
Punjab
1.8%
Arunachal Pradesh
1.8%
Uttarakhand
1.1%
Haryana
1.1%
Himachal Pradesh
Meghalaya
Manipur
Tripura
Nagaland
Mizoram
Sikkim
Goa
Jammu Kashmir
0.6%

Share of Total Allocation for All States in 2013-14


Uttar Pradesh
Bihar
Madhya Pradesh
Andhra Pradesh
Rajasthan
Maharashtra
Tamil Nadu
Odisha
Karnataka
Assam
Gujarat
Jharkhand
Chhattisgarh
Kerala
Punjab
Jammu Kashmir
Uttarakhand
Haryana
Himachal Pradesh
Tripura
Manipur
Meghalaya
Arunachal Pradesh
Nagaland
Mizoram
Sikkim
Goa
Telangana
West Bengal