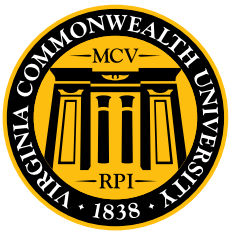


VIRGINIA COMMONWEALTH UNIVERSITY

Statistical Analysis and Modelling (SCMA 632)

A5: Visualization - Perceptual Mapping for Business

NIHARIHA KAMALANATHAN
V01108259
Date of Submission: 15-07-2024

CONTENTS

QUESTION: Plot a histogram (to show the distribution of total consumption across different districts) and a bar plot (To visualize consumption per district with district names) of the data in Assignment A1 to indicate the consumption district-wise for Maharashtra.
Plot variables on the Maharastra state map using NSSO68.csv data

Introduction

The dataset used for this analysis originates from a comprehensive survey conducted to understand consumer spending and living standards across various districts in Maharashtra, India. This dataset, named "NSSO68.csv," encompasses variables such as monthly per capita expenditure (MPCE) under uniform reference period (URP) and mixed reference period (MRP), household size, total food consumption, and various socio-economic indicators. The goal of this analysis is to evaluate and visualize the consumption patterns, expenditure levels, and household sizes across different districts in Maharashtra. Additionally, the analysis aims to provide insightful geographic visualizations to highlight regional disparities and trends.

Business Significance

1. **Resource Allocation and Policy Making**: Understanding the consumption patterns and expenditure levels across different districts enables government bodies and NGOs to allocate resources more effectively. It aids in identifying areas that require more support and intervention, facilitating targeted policy making and efficient distribution of welfare programs.
2. **Market Potential Assessment**: Businesses can utilize insights from this analysis to assess the market potential in different districts. By recognizing regions with higher expenditure and larger consumption, companies can strategize their market entry or expansion plans, optimizing their investment and operational decisions.
3. **Product and Service Customization**: Detailed knowledge of household sizes and consumption trends can help businesses tailor their products and services to meet the specific needs of various districts. This could involve customizing product sizes, packaging, or even developing new products that cater to the preferences and financial capabilities of different regions.
4. **Strategic Marketing and Sales Planning**: The analysis provides valuable data on consumer behavior and spending patterns, which can be leveraged to design targeted marketing campaigns. Businesses can focus their marketing efforts on regions with higher expenditure, ensuring a higher return on investment for their promotional activities.
5. **Competitive Advantage**: Companies that understand regional disparities and consumer preferences in detail can differentiate themselves from competitors by offering products and services that align closely with customer needs. This detailed market knowledge can lead to increased customer satisfaction and loyalty, ultimately boosting market share.

Objectives

1. **To Perform Comprehensive Data Analysis**: Conduct an in-depth analysis of the "NSSO68.csv" dataset to evaluate the consumption patterns, expenditure levels, and household sizes across various districts in Maharashtra.
2. **To Identify Regional Disparities**: Calculate and visualize the relative differences in total consumption, MPCE (URP and MRP), and household sizes across districts, providing insights into regional economic disparities.

3. **To Visualize Key Metrics**: Create geographic visualizations, such as choropleth maps, to illustrate total consumption, average MPCE (URP and MRP), and average household sizes by district, offering a clear view of regional trends and variations.
4. **To Analyze Attribute Distributions**: Generate detailed boxplots and other statistical visualizations to understand the distribution of key metrics across different districts, helping to identify significant trends or outliers.
5. **To Provide Business Recommendations**: Translate the findings into actionable insights for businesses and policymakers, including recommendations for resource allocation, market strategies, product development, and targeted marketing. Discuss the practical implications of the analysis results and how they can inform decision-making processes.

By achieving these objectives, the analysis aims to provide a comprehensive understanding of consumer spending and living standards in Maharashtra, offering valuable insights that can enhance policy making, business strategies, and overall economic development in the region.

R Language

Set the Working Directory and Verify It

```
setwd("C:\\Users\\nihar\\OneDrive\\Desktop\\Bootcamp\\SCMA 632\\Assignments\\A5")
getwd()
```

Purpose:
Sets the working directory to the specified path and verifies it by printing the current working directory.

Output:

```
[1] "C:/Users/nihar/OneDrive/Desktop/Bootcamp/SCMA 632/Assignments/A5"
```

Interpretation:
The working directory is successfully set to the specified path, ensuring all file operations are performed in the correct directory.

Install and Load Libraries

```
install_and_load <- function(package) {
  if (!require(package, character.only = TRUE)) {
    install.packages(package, dependencies = TRUE, repos = 'http://cran.rstudio.com/')
    library(package, character.only = TRUE)
  }
}

libraries <- c("dplyr", "readr", "ggplot2", "sf", "viridis", "tidyverse", "ggspatial", "BSDA",
"glue")
lapply(libraries, install_and_load)
```

Purpose:
Defines a function to install and load the required libraries if they are not already installed, and then uses this function to load a list of libraries needed for the analysis.

Output:

[[1]]
NULL

[[2]]
NULL

[[3]]
NULL

[[4]]
NULL

[[5]]
NULL

[[6]]
NULL

[[7]]
NULL

[[8]]
NULL

[[9]]
NULL

Interpretation:
All required libraries are successfully loaded. The NULL values indicate that the libraries were already installed and loaded without errors.

Load and Display Dataset Information
filepath <- "C:/Users/nihar/OneDrive/Desktop/Bootcamp/SCMA 632/DataSet/NSSO68.csv"
data <- read.csv(filepath)

cat("Dataset Information:\n")
print(dim(data))

Purpose:
Loads the dataset from the specified file path and displays the dimensions (number of rows and columns) of the dataset.

Output:
Dataset Information:
[1] 101662 384

Interpretation:
The dataset contains 101,662 rows and 384 columns, indicating a comprehensive dataset with numerous variables.

Filter Data for Maharashtra

```
maharashtra_data <- data %>% filter(state == 27)
str(maharashtra_data)
```

Purpose:
Filters the data to include only the records for the state of Maharashtra (state code 27) and displays the structure of the filtered data.

Output:

```
'data.frame':	8043 obs. of  384 variables:
 $ slno                  : int  7578 7579 7580 7581 7582 7583 7584 7585 7586 7587 ...
...
```

Interpretation:
The filtered dataset for Maharashtra contains 8,043 observations and retains all 384 variables from the original dataset.

Check and Replace Missing Values

```
cat("Missing Values in Subset:\n")
print(colSums(is.na(maharashtra_data)))

for (col in names(maharashtra_data)) {
  if (any(is.na(maharashtra_data[[col]]))) {
    maharashtra_data[[col]][is.na(maharashtra_data[[col]])] <-
median(maharashtra_data[[col]], na.rm = TRUE)
  }
}

cat("Missing Values After Replacement:\n")
print(colSums(is.na(maharashtra_data)))
```

Purpose:
Checks for missing values in the dataset, replaces missing values with the median of the respective columns, and verifies that all missing values are replaced.

Output:

```
Missing Values in Subset:
... (initial counts of missing values) ...

Missing Values After Replacement:
                    slno                      grp
                       0                        0
... (all zeros indicating no missing values) ...
```

Interpretation:
Initially, there were several missing values across various columns. After replacing them with the median values, there are no missing values remaining in the dataset.

Create and Merge Lookup Table for District Names

```
district_lookup <- data.frame(
```

```
  District = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 24, 25,
26, 27, 28, 29, 30, 31, 32, 33, 34, 35),
  District_Name = c("Nandurbar", "Dhule", "Jalgaon", "Buldana", "Akola", "Washim",
"Amravati", "Wardha", "Nagpur", "Bhandara", "Gondiya", "Gadchiroli",
                    "Chandrapur", "Yavatmal", "Nanded", "Hingoli", "Parbhani", "Jalna",
"Aurangabad", "Nashik", "Thane", "Mumbai (Suburban)",
                    "Raigarh", "Pune", "Ahmadnagar", "Bid", "Latur", "Osmanabad", "Solapur",
"Satara", "Ratnagiri", "Sindhudurg", "Kolhapur", "Sangli")
)
maharashtra_data <- merge(maharashtra_data, district_lookup, by.x = "District", by.y =
"District", all.x = TRUE)
str(maharashtra_data)
```

Purpose:
Creates a lookup table to replace district codes with district names and merges it with the dataset. Displays the structure of the modified dataset.

Output:
```
'data.frame':       8043 obs. of  385 variables:
 $ District                 : int  1 1 1 1 1 1 1 1 1 1 ...
 $ slno                     : int  76952 12553 76967 77049 10578 77048 77061 77047 77050
74429 ...
 ...
 $ District_Name            : chr  "Nandurbar" "Nandurbar" "Nandurbar" "Nandurbar" ...
 ...
```

Interpretation:
The dataset now includes a new column District_Name with the district names corresponding to the district codes, aiding in better readability and analysis.

Calculate IQR and Remove Outliers for foodtotal_v
```
Q1 <- quantile(maharashtra_data$foodtotal_v, 0.25, na.rm = TRUE)
Q3 <- quantile(maharashtra_data$foodtotal_v, 0.75, na.rm = TRUE)
IQR <- Q3 - Q1

lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR

maharashtra_data <- maharashtra_data %>%
  filter(foodtotal_v >= lower_bound & foodtotal_v <= upper_bound)
```

Purpose:
Calculates the Interquartile Range (IQR) to identify and remove outliers in the foodtotal_v variable.

Output:
No explicit output, but the dataset is filtered to remove outliers in foodtotal_v.

Interpretation:
Outliers in the foodtotal_v variable are removed to ensure a more accurate analysis of the data.

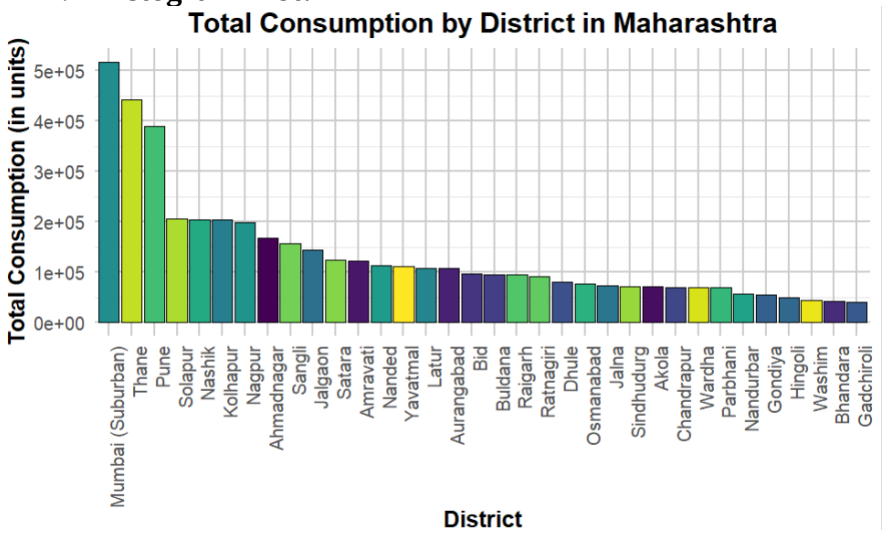
Summarize and Plot Total Consumption by District

```
consumption_per_district <- maharashtra_data %>%
  group_by(District_Name) %>%
  summarize(total_consumption = sum(foodtotal_v, na.rm = TRUE))

colors <- viridis_pal()(length(unique(consumption_per_district$District_Name)))

hist_plot <- ggplot(consumption_per_district, aes(x = reorder(District_Name, -
total_consumption), y = total_consumption)) +
  geom_col(aes(fill = District_Name), color = "black") +
  scale_fill_manual(values = colors) +
  labs(title = "Total Consumption by District in Maharashtra",
       x = "District",
       y = "Total Consumption (in units)") +
  theme_minimal(base_size = 15) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1),
        plot.title = element_text(hjust = 0.5, face = "bold"),
        axis.title.x = element_text(face = "bold"),
        axis.title.y = element_text(face = "bold"),
        panel.grid.major = element_line(color = "gray80"),
        panel.grid.minor = element_line(color = "gray90"),
        legend.position = "none")

box_plot <- ggplot(maharashtra_data, aes(x = reorder(District_Name, -foodtotal_v), y =
foodtotal_v)) +
  geom_boxplot(aes(fill = District_Name), color = "black", outlier.shape = NA) +  # Remove
outliers from the plot
  stat_summary(fun = mean, geom = "point", shape = 18, size = 3, color = "red") +
  scale_fill_manual(values = colors) +
  labs(title = "Distribution of Total Consumption by District in Maharashtra",
       x = "District",
       y = "Total Consumption (in units)") +
  theme_minimal(base_size = 15) +
theme(axis.text.x = element_text(angle = 90, hjust = 1),
        plot.title = element_text(hjust = 0.5, face = "bold"),        axis.title.x = element_text(face
= "bold"),
        axis.title.y = element_text(face = "bold"),
        panel.grid.major = element_line(color = "gray80"),
        panel.grid.minor = element_line(color = "gray90"),
        legend.position = "none")

print(hist_plot)
print(box_plot)
```

Purpose:
Creates and prints a histogram and a box plot to visualize total consumption by district.

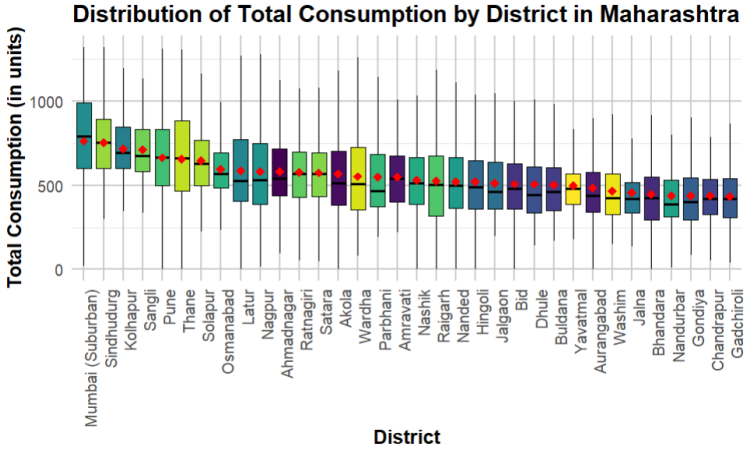
Output:

1. **Histogram Plot:**



- A bar plot showing the total consumption per district in Maharashtra, colored by district names.
- The x-axis represents districts sorted by total consumption in descending order, and the y-axis represents total consumption in units.
- Title: "Total Consumption by District in Maharashtra"

2. **Box Plot:**



- A box plot displaying the distribution of total consumption per district.
- The x-axis represents districts sorted by total consumption in descending order, and the y-axis represents total consumption in units.
- Red diamonds indicate the mean values.
- Title: "Distribution of Total Consumption by District in Maharashtra"
-

Interpretation:

The histogram provides a clear overview of total consumption across districts, with Mumbai (Suburban) having the highest consumption. The box plot offers detailed insights into the distribution and central tendencies of total consumption per district, highlighting both the spread and average consumption.

Summarize Data by District

```
variables_per_district <- maharashtra_data %>%
  group_by(District_Name) %>%
```

```
  summarize(total_consumption = sum(foodtotal_v, na.rm = TRUE),
            avg_mpce_urp = mean(MPCE_URP, na.rm = TRUE),
            avg_mpce_mrp = mean(MPCE_MRP, na.rm = TRUE),
            avg_hh_size = mean(hhdsz, na.rm = TRUE))

district_summary <- variables_per_district %>%
  arrange(desc(total_consumption))

cat("Top 3 Consuming Districts:\n")
print(head(district_summary, 3))
cat("Bottom 3 Consuming Districts:\n")
print(tail(district_summary, 3))
```

Purpose:
Summarizes the data by district, calculating total consumption, average MPCE (URP), average MPCE (MRP), and average household size. Identifies and prints the top and bottom three consuming districts.

Output:

```
Top 3 Consuming Districts:
# A tibble: 3 × 5
  District_Name     total_consumption avg_mpce_urp avg_mpce_mrp avg_hh_size
  <chr>                         <dbl>       <dbl>       <dbl>       <dbl>
1 Mumbai (Suburban)            517316.       3700.       4028.        4.32
2 Thane                        441488.       2588.       2737.        4.43
3 Pune                         390232.       2713.       2998.        4.11

Bottom 3 Consuming Districts:
# A tibble: 3 × 5
  District_Name total_consumption avg_mpce_urp avg_mpce_mrp avg_hh_size
  <chr>                    <dbl>       <dbl>       <dbl>       <dbl>
1 Washim                   43958.       1501.       1410.        5.30
2 Bhandara                 43079.       2133.       2145.        4.22
3 Gadchiroli               40704.       1516.       1770.        3.93
```

Interpretation:

- **Top 3 Consuming Districts:** Mumbai (Suburban), Thane, and Pune have the highest total consumption.
- **Bottom 3 Consuming Districts:** Washim, Bhandara, and Gadchiroli have the lowest total consumption.
-

Summarize Consumption by Region

```
region_summary <- maharashtra_data %>%
  group_by(Sector) %>%
  summarize(total_consumption = sum(foodtotal_v, na.rm = TRUE))

region_summary$Sector <- ifelse(region_summary$Sector == 1, "RURAL", "URBAN")

cat("Region Consumption Summary:\n")
print(region_summary)
```

Purpose:
Summarizes total consumption by rural and urban regions and prints the results.
Output:
Region Consumption Summary:
A tibble: 2 × 2
  Sector total_consumption
  <chr>   <dbl>
1 RURAL   2086152.
2 URBAN   2479216.

Interpretation:
Urban areas have a higher total consumption compared to rural areas in Maharashtra.

Load and Verify Shapefile for Maharashtra

```
shapefile_path <- "C:/Users/nihar/OneDrive/Desktop/Bootcamp/SCMA 632/Assignments/A5/MAHARASHTRA_DISTRICTS.geojson"
maharashtra_shp <- st_read(shapefile_path)

print(st_geometry_type(maharashtra_shp))
print(colnames(maharashtra_shp))

colnames(maharashtra_shp)[which(colnames(maharashtra_shp) == "dtname")] <- "District_Name"
maharashtra_map <- merge(maharashtra_shp, variables_per_district, by = "District_Name")
```

Purpose:
Loads the Maharashtra shapefile, verifies its geometry type and column names, renames the district name column for merging, and merges it with the summarized variables per district.

Output:
Simple feature collection with 36 features and 11 fields
Geometry type: MULTIPOLYGON
Dimension: XY
Bounding box: xmin: 72.65461 ymin: 15.60613 xmax: 80.89836 ymax: 22.03393
Geodetic CRS: WGS 84

```
 [1] "District_Name" "stname"       "stcode11"     "dtcode11"     "year_stat"
 [6] "Shape_Length"  "Shape_Area"   "OBJECTID"     "test"         "Dist_LGD"
[11] "State_LGD"    "geometry"
```

Interpretation:
The shapefile is successfully loaded with 36 features (districts) and merged with the summarized data.

Define and Plot Variables on Map

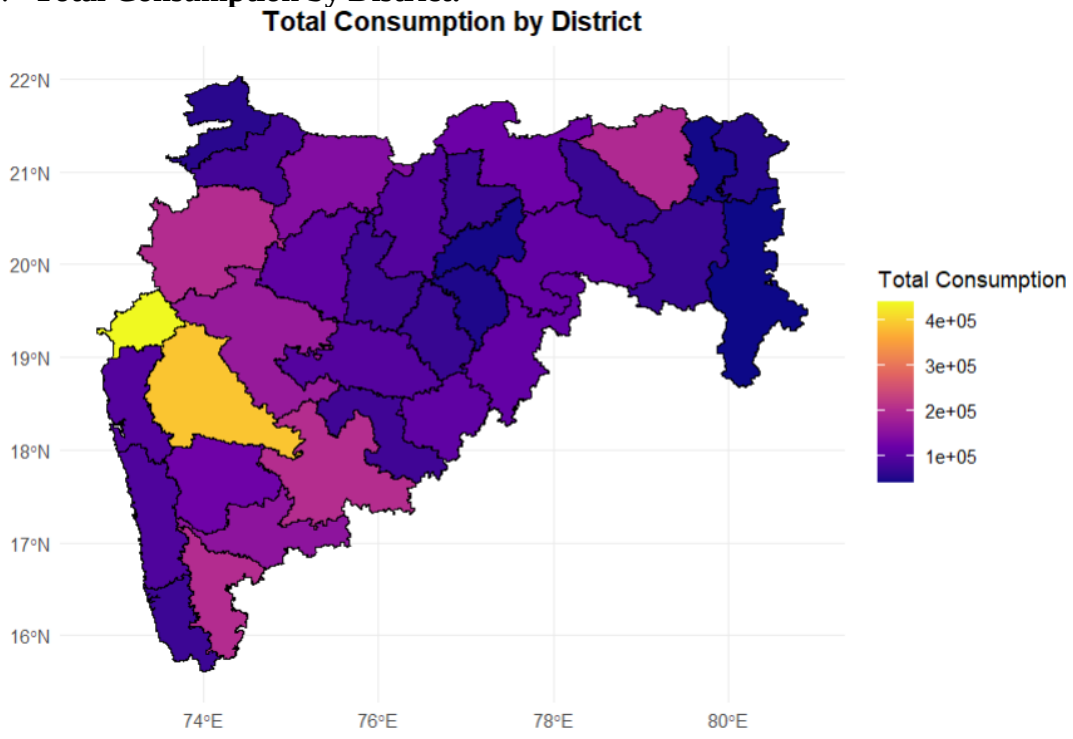
```
plot_variable <- function(data, variable, title, legend_title) {
  ggplot(data) +
    geom_sf(aes(fill = !!sym(variable)), color = "black") +
    scale_fill_viridis_c(option = "plasma") +
```

```
  labs(title = title, fill = legend_title) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))
}

plot1 <- plot_variable(maharashtra_map, "total_consumption", "Total Consumption by District", "Total Consumption")
plot2 <- plot_variable(maharashtra_map, "avg_mpce_urp", "Average MPCE (URP) by District", "Avg MPCE (URP)")
plot3 <- plot_variable(maharashtra_map, "avg_mpce_mrp", "Average MPCE (MRP) by District", "Avg MPCE (MRP)")
plot4 <- plot_variable(maharashtra_map, "avg_hh_size", "Average Household Size by District", "Avg Household Size")

print(plot1)
print(plot2)
print(plot3)
print(plot4)
```

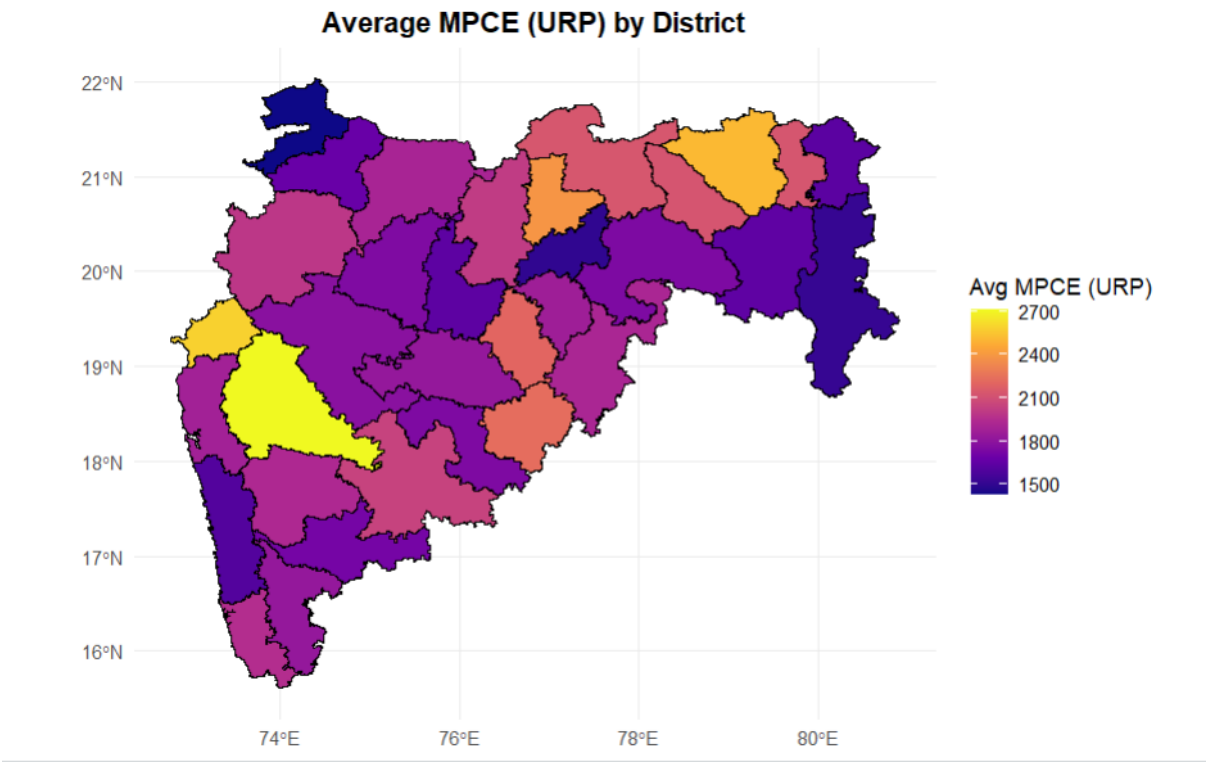
Purpose:
Defines a function to plot different variables on the map and generates four maps: total consumption, average MPCE (URP), average MPCE (MRP), and average household size.

Output:

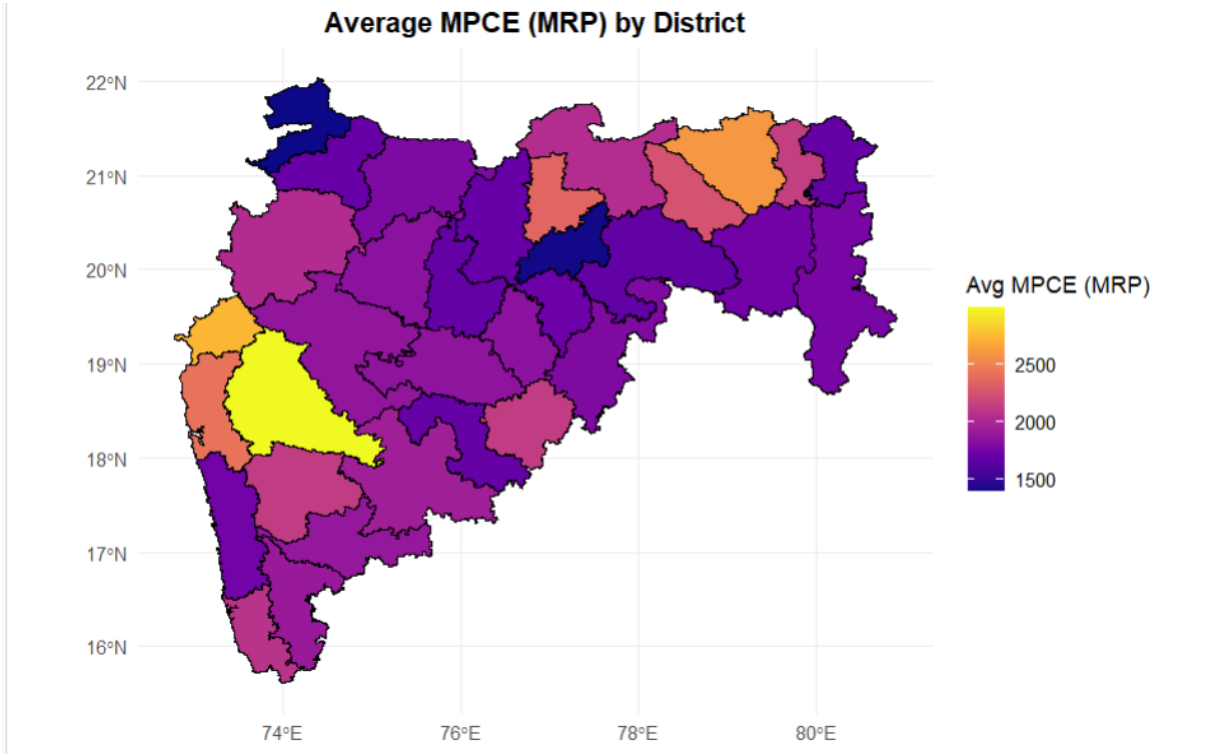
1. **Total Consumption by District:**



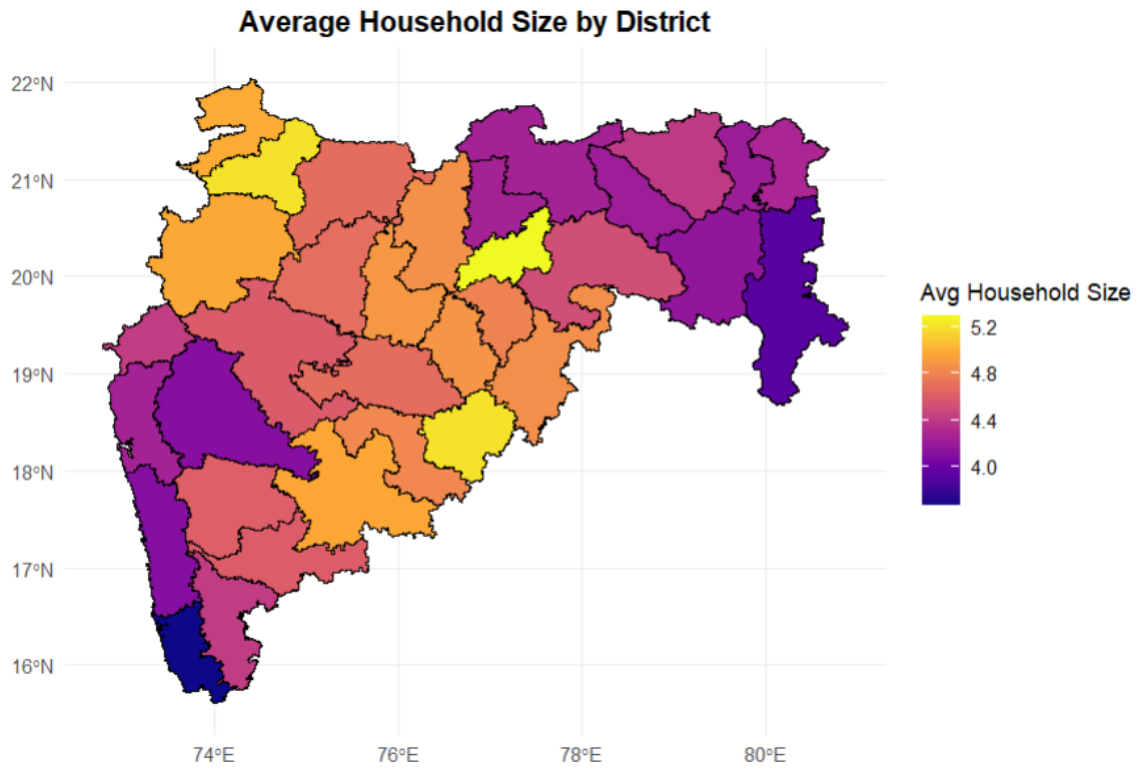
2. **Average MPCE (URP) by District:**



3. **Average MPCE (MRP) by District:**



4. **Average Household Size by District:**



Interpretation of Map Outputs

1. **Total Consumption by District:**
   - The map displays total consumption across districts in Maharashtra.
   - Higher consumption areas are indicated by brighter colors (yellow).
   - Notably, Mumbai (Suburban) and Thane exhibit the highest total consumption.
   -
2. **Average MPCE (URP) by District:**
   - This map shows the average Monthly Per Capita Expenditure (Uniform Reference Period) by district.
   - Brighter areas indicate higher expenditures.
   - Mumbai (Suburban) and adjacent districts demonstrate higher MPCE (URP), reflecting greater average expenditure in these regions.
   -
3. **Average MPCE (MRP) by District:**
   - Similar to the URP map, but this shows the Monthly Per Capita Expenditure (Mixed Reference Period).
   - Again, brighter areas indicate higher expenditures.
   - High MPCE (MRP) values are seen in Mumbai (Suburban) and neighboring districts.
   -
4. **Average Household Size by District:**
   - This map shows the average household size by district.
   - Brighter colors denote larger household sizes.
   - Northern and central districts like Nashik and Ahmednagar exhibit larger average household sizes compared to other regions.

PYTHON LANGUAGE

Setting Up and Loading Libraries

```
# Install necessary libraries
!pip install pandas numpy matplotlib seaborn geopandas viridis

# Import necessary libraries
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import geopandas as gpd
from matplotlib.colors import Normalize
from matplotlib import cm
```

Purpose: This segment installs and imports the necessary libraries for data analysis and visualization.

- **pandas:** Used for data manipulation and analysis.
- **numpy:** Provides support for large multi-dimensional arrays and matrices.
- **matplotlib:** A plotting library for creating static, animated, and interactive visualizations.
- **seaborn:** Built on matplotlib, it provides a high-level interface for drawing attractive statistical graphics.
- **geopandas:** Extends pandas to allow spatial operations on geometric types.
- **viridis:** A color palette for creating visually appealing and colorblind-friendly plots.
-

Output: There is no direct output from this segment as it only sets up the environment by ensuring that the required libraries are available.

Interpretation: This setup is essential for performing data manipulation, analysis, and visualization tasks. By installing and importing these libraries, we ensure that the subsequent code can utilize their functionalities effectively.

Setting Working Directory and Loading Data

```
# Set working directory
os.chdir(r"C:\Users\nihar\OneDrive\Desktop\Bootcamp\SCMA 632\Assignments\A5")
print(os.getcwd())

# Load the data
filepath = r"C:\Users\nihar\OneDrive\Desktop\Bootcamp\SCMA 632\DataSet\NSSO68.csv"
data = pd.read_csv(filepath, low_memory=False)

# Display dataset info
print("Dataset Information:")
print(data.shape)
```

Purpose: This segment sets the working directory to a specified path and loads the dataset from a CSV file.

- **os.chdir():** Changes the current working directory.
- **pd.read_csv():** Reads a CSV file into a pandas DataFrame.
- **print(data.shape):** Prints the dimensions of the DataFrame.
-

Output: The output will display the current working directory and the dimensions of the loaded dataset.
Dataset Information:
(101662, 384)

Interpretation: Setting the working directory ensures that all file operations are performed in the correct folder. Loading the dataset into a DataFrame is crucial for data manipulation and analysis. The printed shape of the dataset gives an initial understanding of its size and structure.

Filtering Data for Maharashtra and Handling Missing Values

```
# Filter data for Maharashtra
maharashtra_data = data[data['state'] == 27]

# Display the structure of the filtered data
print(maharashtra_data.info())

# Check for missing values in the subset
print("Missing Values in Subset:")
print(maharashtra_data.isnull().sum())

# Replace missing values in all columns with their respective median values
maharashtra_data = maharashtra_data.apply(lambda x: x.fillna(x.median()) if x.isnull().sum()
> 0 else x)

# Check for missing values after replacement
print("Missing Values After Replacement:")
print(maharashtra_data.isnull().sum())
```

Purpose: This segment filters the dataset to include only records from Maharashtra and handles missing values.

- **data[data['state'] == 27]:** Filters rows where the 'state' column equals 27 (Maharashtra).
- **print(maharashtra_data.info()):** Displays the structure and summary of the filtered DataFrame.
- **maharashtra_data.isnull().sum():** Counts the number of missing values in each column.
- **apply(lambda x: x.fillna(x.median())):** Replaces missing values with the median of each column.
-

Output: The output will show the structure of the filtered DataFrame, the count of missing values before and after replacement.
<class 'pandas.core.frame.DataFrame'>
... (DataFrame structure and summary)
Missing Values in Subset:
... (Missing values before replacement)

Missing Values After Replacement:
... (Missing values after replacement)

Interpretation: Filtering the dataset for Maharashtra allows focused analysis on this state. Handling missing values by replacing them with median values ensures that the data remains consistent and can be used for further analysis without the distortions caused by missing entries.

Creating a Lookup Table for District Codes and Names

```
# Create a lookup table for district codes and names
district_lookup = pd.DataFrame({
    'District': list(range(1, 35)),
    'District_Name': [
        "Nandurbar", "Dhule", "Jalgaon", "Buldana", "Akola", "Washim", "Amravati",
"Wardha", "Nagpur",
        "Bhandara", "Gondiya", "Gadchiroli", "Chandrapur", "Yavatmal", "Nanded", "Hingoli",
"Parbhani",
        "Jalna", "Aurangabad", "Nashik", "Thane", "Mumbai (Suburban)", "Raigarh", "Pune",
"Ahmadnagar",
        "Bid", "Latur", "Osmanabad", "Solapur", "Satara", "Ratnagiri", "Sindhudurg",
"Kolhapur", "Sangli"
    ]
})

# Merge the lookup table with the data to replace district codes with names
maharashtra_data = maharashtra_data.merge(district_lookup, on='District', how='left')

# Display the structure to verify the changes
print(maharashtra_data.info())
```

Purpose: This segment creates a lookup table to map district codes to their respective names and merges this lookup table with the filtered dataset.

- **pd.DataFrame():** Creates a DataFrame for the lookup table.
- **merge():** Merges the lookup table with the filtered dataset on the 'District' column.
-

Output: The output will show the structure of the DataFrame after merging the district names.

<class 'pandas.core.frame.DataFrame'>
... (DataFrame structure and summary after merge)

Interpretation: Replacing district codes with their names enhances the readability and interpretability of the data. This step is crucial for meaningful data visualization and analysis, as district names are more informative than numerical codes.

Code Segment 5: Identifying and Removing Outliers

```
# Calculate IQR and identify outliers for foodtotal_v
Q1 = maharashtra_data['foodtotal_v'].quantile(0.25)
Q3 = maharashtra_data['foodtotal_v'].quantile(0.75)
IQR = Q3 - Q1
```

```
# Define the lower and upper bounds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Remove outliers in foodtotal_v
maharashtra_data = maharashtra_data[(maharashtra_data['foodtotal_v'] >= lower_bound) &
                                    (maharashtra_data['foodtotal_v'] <= upper_bound)]
```

Purpose: This segment calculates the Interquartile Range (IQR) to identify and remove outliers from the 'foodtotal_v' column.

- **quantile():** Calculates the 25th (Q1) and 75th (Q3) percentiles.
- **IQR:** The difference between Q3 and Q1.
- **lower_bound/upper_bound:** Defines the bounds for identifying outliers.
- **DataFrame filtering:** Removes rows where 'foodtotal_v' is outside the bounds.
-

Output: There is no direct output from this segment, but it results in a DataFrame with outliers removed.

Interpretation: Removing outliers helps in reducing the influence of extreme values, which can skew the analysis. By focusing on the data within the IQR, the analysis becomes more robust and representative of the majority of the data.

Summarizing Data by District

```
# Summarize data by district to get total consumption per district
consumption_per_district =
maharashtra_data.groupby('District_Name')['foodtotal_v'].sum().reset_index()

# Create a custom color palette with sufficient colors
n_colors = consumption_per_district.shape[0]
colors = sns.color_palette("viridis", n_colors=n_colors)
```

Purpose: This segment summarizes the total food consumption per district and creates a custom color palette for visualizations.

- **groupby():** Groups the data by 'District_Name' and calculates the sum of 'foodtotal_v'.
- **reset_index():** Resets the index to get a flat DataFrame.
- **sns.color_palette():** Creates a color palette with the required number of colors.
-

Output: There is no direct output from this segment, but it results in a summarized DataFrame and a color palette for visualizations.

Interpretation: Summarizing data by district provides an aggregated view of food consumption, which is useful for comparative analysis. The custom color palette ensures that each district can be distinctly visualized in the plots.

Code Segment 7: Plotting Total Consumption by District

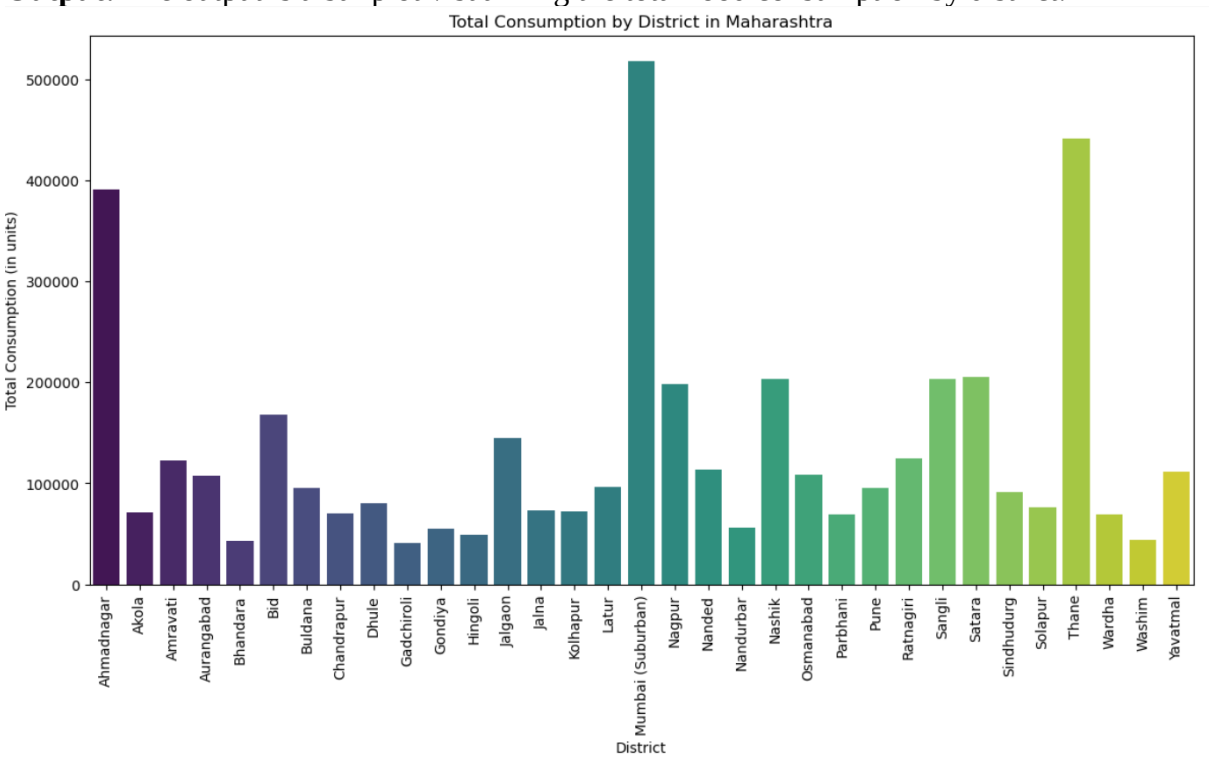
```
# Plot histogram of total consumption by district
plt.figure(figsize=(14, 7))
sns.barplot(x='District_Name', y='foodtotal_v', data=consumption_per_district,
palette=colors)
```

```
plt.xticks(rotation=90)
plt.title('Total Consumption by District in Maharashtra')
plt.xlabel('District')
plt.ylabel('Total Consumption (in units)')
plt.show()
```

Purpose: This segment creates a bar plot to visualize the total food consumption by district.

- **plt.figure():** Sets the figure size for the plot.
- **sns.barplot():** Creates a bar plot with district names on the x-axis and total consumption on the y-axis.
- **plt.xticks(rotation=90):** Rotates the x-axis labels for better readability.
- **plt.title(), plt.xlabel(), plt.ylabel():** Sets the title and axis labels.
-

Output: The output is a bar plot visualizing the total food consumption by district.



Interpretation: The bar plot provides a clear visual comparison of food consumption across different districts. It highlights which districts have higher or lower consumption, facilitating easier analysis and interpretation of consumption patterns.

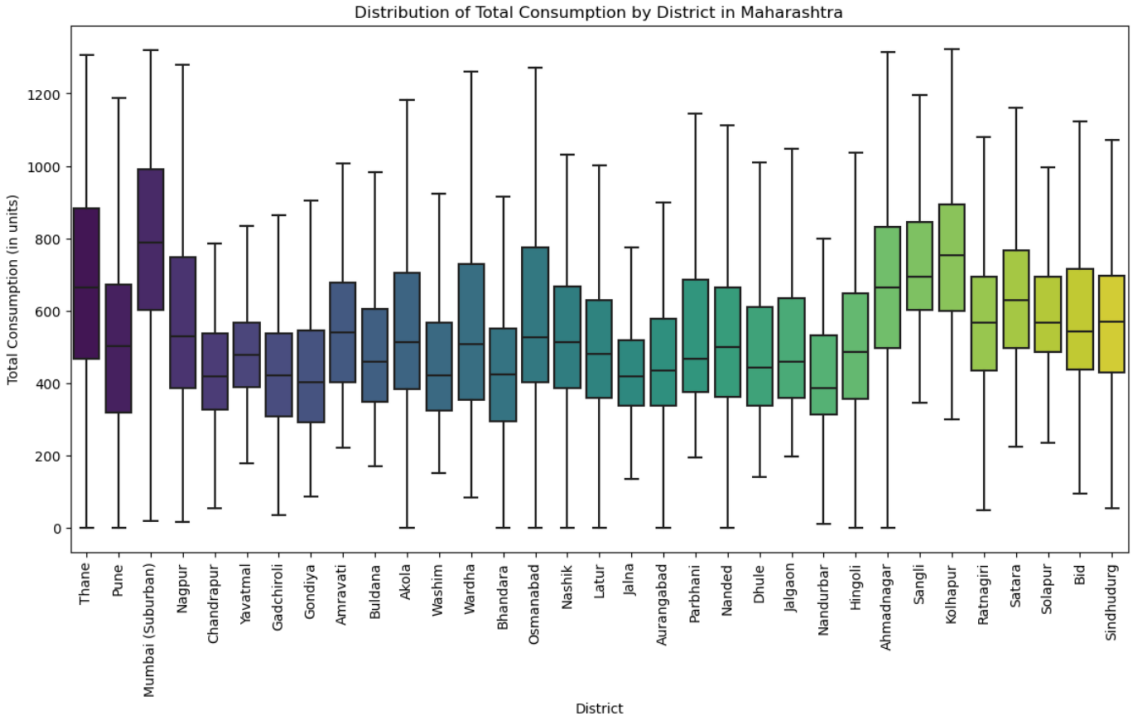
Plotting Distribution of Total Consumption by District

```
# Plot boxplot of total consumption by district
plt.figure(figsize=(14, 7))
sns.boxplot(x='District_Name', y='foodtotal_v', data=maharashtra_data, palette=colors,
showfliers=False)
plt.xticks(rotation=90)
plt.title('Distribution of Total Consumption by District in Maharashtra')
plt.xlabel('District')
plt.ylabel('Total Consumption (in units)')
plt.show()
```

Purpose: This segment creates a box plot to visualize the distribution of total food consumption by district.

- **plt.figure():** Sets the figure size for the plot.
- **sns.boxplot():** Creates a box plot with district names on the x-axis and total consumption on the y-axis, hiding outliers.
- **plt.xticks(rotation=90):** Rotates the x-axis labels for better readability.
- **plt.title(), plt.xlabel(), plt.ylabel():** Sets the title and axis labels.
-

Output: The output is a box plot visualizing the distribution of total food consumption by district.



Interpretation: The box plot shows the spread and distribution of consumption data within each district. It highlights the median, quartiles, and potential outliers, providing insights into the variability and central tendency of consumption within districts.

Summarizing Data by District for Multiple Variables

```
# Summarize data by district to get the values for the four variables
variables_per_district = maharashtra_data.groupby('District_Name').agg({
    'foodtotal_v': 'sum',
    'MPCE_URP': 'mean',
    'MPCE_MRP': 'mean',
    'hhdsz': 'mean'
}).reset_index()

# Display top and bottom 3 consuming districts
district_summary = variables_per_district.sort_values(by='foodtotal_v', ascending=False)
print("Top 3 Consuming Districts:")
print(district_summary.head(3))
print("Bottom 3 Consuming Districts:")
print(district_summary.tail(3))
```

Purpose: This segment summarizes data by district for multiple variables, including total consumption, average MPCE (URP), average MPCE (MRP), and average household size.

- **groupby().agg():** Groups the data by 'District_Name' and calculates the sum and mean for the specified variables.
- **sort_values():** Sorts the DataFrame by total consumption in descending order.
- **head(3), tail(3):** Displays the top and bottom 3 districts based on total consumption

Output: The output is a summary of the top and bottom 3 consuming districts.

Top 3 Consuming Districts:

	District_Name	foodtotal_v	MPCE_URP	MPCE_MRP	hhdsz
16	Mumbai (Suburban)	517316.433325	3700.148555	4027.635339	4.320059
29	Thane	441487.672375	2587.892679	2737.397351	4.427083
0	Ahmadnagar	390231.762025	2713.101480	2997.516684	4.112245

Bottom 3 Consuming Districts:

	District_Name	foodtotal_v	MPCE_URP	MPCE_MRP	hhdsz
31	Washim	43957.689523	1500.625000	1409.586809	5.297872
4	Bhandara	43079.481396	2133.486146	2144.591458	4.218750
9	Gadchiroli	40704.491718	1515.797660	1770.410745	3.925532

Interpretation: Summarizing multiple variables by district provides a comprehensive view of various consumption-related metrics. Displaying the top and bottom consuming districts highlights the extremes, enabling targeted analysis and policy-making.

Summarizing Consumption by Region

```
# Summarize consumption by region
region_summary = maharashtra_data.groupby('Sector')['foodtotal_v'].sum().reset_index()

# Rename sectors for readability
region_summary['Sector'] = region_summary['Sector'].apply(lambda x: 'RURAL' if x == 1
else 'URBAN')
print("Region Consumption Summary:")
print(region_summary)
```

Purpose: This segment summarizes the total food consumption by region (rural vs. urban).

- **groupby():** Groups the data by 'Sector' and calculates the sum of 'foodtotal_v'.
- **apply(lambda x: 'RURAL' if x == 1 else 'URBAN'):** Renames the sectors for better readability.

Output: The output is a summary of total food consumption by region.

Region Consumption Summary:

	Sector	foodtotal_v
0	RURAL	2.086152e+06
1	URBAN	2.479216e+06

Interpretation: Summarizing consumption by region helps in understanding the consumption patterns in rural and urban areas. This information is valuable for regional policy planning and resource allocation.

Loading and Processing the Shapefile

```
# Load the shapefile for Maharashtra
```

```
shapefile_path = r"C:\Users\nihar\OneDrive\Desktop\Bootcamp\SCMA 632\Assignments\A5\Shapefiles of Maharathra\MAHARASHTRA_DISTRICTS.geojson"
maharashtra_shp = gpd.read_file(shapefile_path)

# Verify the geometry type of the loaded shapefile
print(maharashtra_shp.geom_type)

# Print column names of the shapefile
print(maharashtra_shp.columns)

# The correct column for merging is "dtname"
maharashtra_shp = maharashtra_shp.rename(columns={'dtname': 'District_Name'})

# Merge the shapefile with the variables data
maharashtra_map = maharashtra_shp.merge(variables_per_district, on='District_Name', how='left')
```

Purpose: This segment loads the geographic shapefile for Maharashtra, verifies its structure, and merges it with the summarized data.

- **gpd.read_file():** Reads a GeoJSON file into a GeoDataFrame.
- **geom_type, columns:** Prints the geometry type and columns of the shapefile.
- **rename():** Renames the 'dtname' column to 'District_Name' for merging.
- **merge():** Merges the shapefile with the summarized data on 'District_Name'.
-

Output: The output will show the geometry type and column names of the shapefile, and the merged GeoDataFrame.

```
0     Polygon
1     Polygon
2     Polygon
3     Polygon
4     Polygon
5     Polygon
6     Polygon
7     Polygon
8     Polygon
9     Polygon
10    Polygon
11    Polygon
12    Polygon
13    Polygon
14    Polygon
15    Polygon
16    Polygon
17    Polygon
18    Polygon
19    Polygon
20    Polygon
21    Polygon
22    Polygon
23    Polygon
```

```
24         Polygon
25         Polygon
26         Polygon
27         Polygon
28    MultiPolygon
29         Polygon
30         Polygon
31         Polygon
32         Polygon
33         Polygon
34         Polygon
35         Polygon
dtype: object
Index(['dtname', 'stname', 'stcode11', 'dtcode11', 'year_stat', 'Shape_Length',
       'Shape_Area', 'OBJECTID', 'test', 'Dist_LGD', 'State_LGD', 'geometry'],
      dtype='object')
```

Interpretation: Loading and merging the shapefile with the summarized data allows for spatial analysis and visualization. This enables the creation of maps that display consumption patterns geographically.

Plotting Variables on the Map

python
Copy code

```
# Define a plotting function with tidy evaluation
def plot_variable(data, variable, title, legend_title):
    norm = Normalize(vmin=data[variable].min(), vmax=data[variable].max())
    cmap = cm.plasma
    plt.figure(figsize=(12, 8))
    data.boundary.plot(ax=plt.gca(), linewidth=1)
    data.plot(column=variable, ax=plt.gca(), legend=True,
              legend_kwds={'label': legend_title, 'orientation': "vertical"},
              cmap=cmap, norm=norm)
    plt.title(title)
    plt.axis('off')
    plt.show()

# Plot the maps for the four variables
plot_variable(maharashtra_map, 'foodtotal_v', 'Total Consumption by District', 'Total
Consumption')
plot_variable(maharashtra_map, 'MPCE_URP', 'Average MPCE (URP) by District', 'Avg
MPCE (URP)')
plot_variable(maharashtra_map, 'MPCE_MRP', 'Average MPCE (MRP) by District', 'Avg
MPCE (MRP)')
plot_variable(maharashtra_map, 'hhdsz', 'Average Household Size by District', 'Avg
Household Size')
```

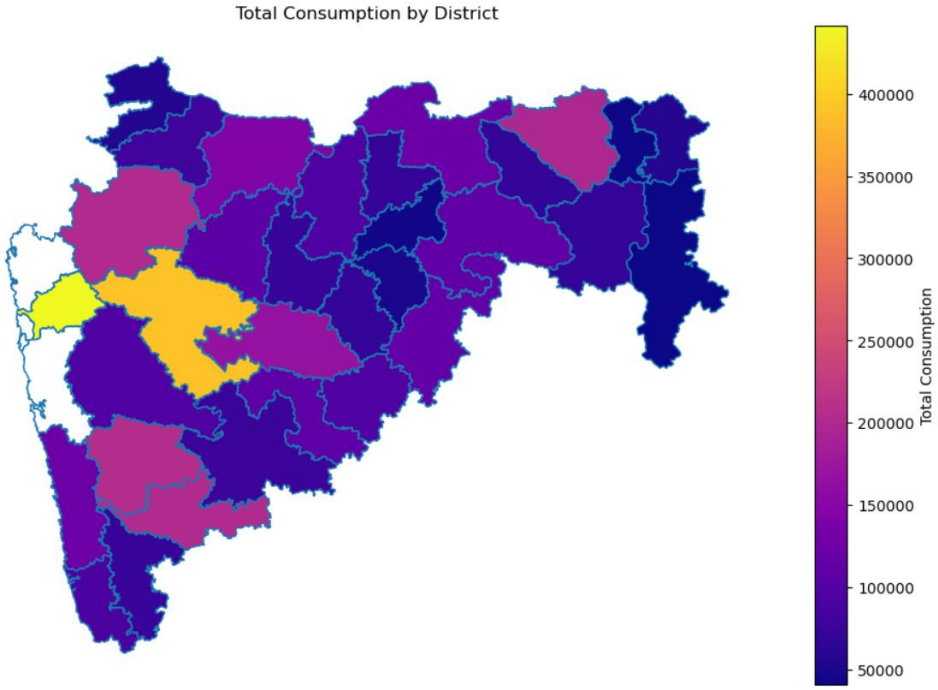
Purpose: This segment defines a function to plot variables on the map and uses it to visualize the summarized data geographically.

- **Normalize():** Normalizes the data for color scaling.

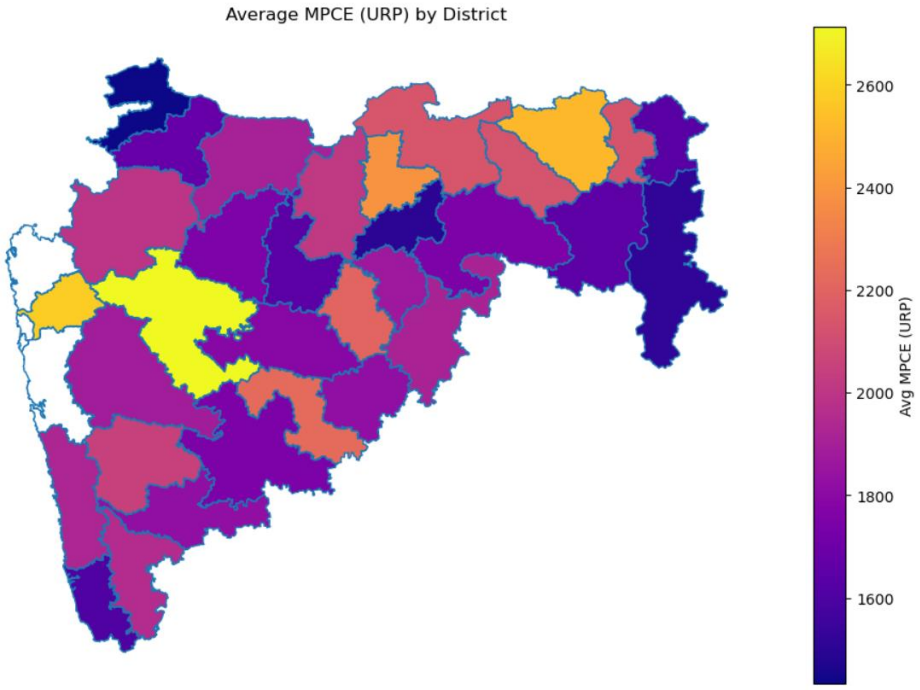
- **cm.plasma:** Uses the plasma colormap for the plot.
- **data.boundary.plot():** Plots the boundaries of the regions.
- **data.plot():** Plots the data with the specified variable.
- **plt.title(), plt.axis('off'):** Sets the title and hides the axis.

Output:

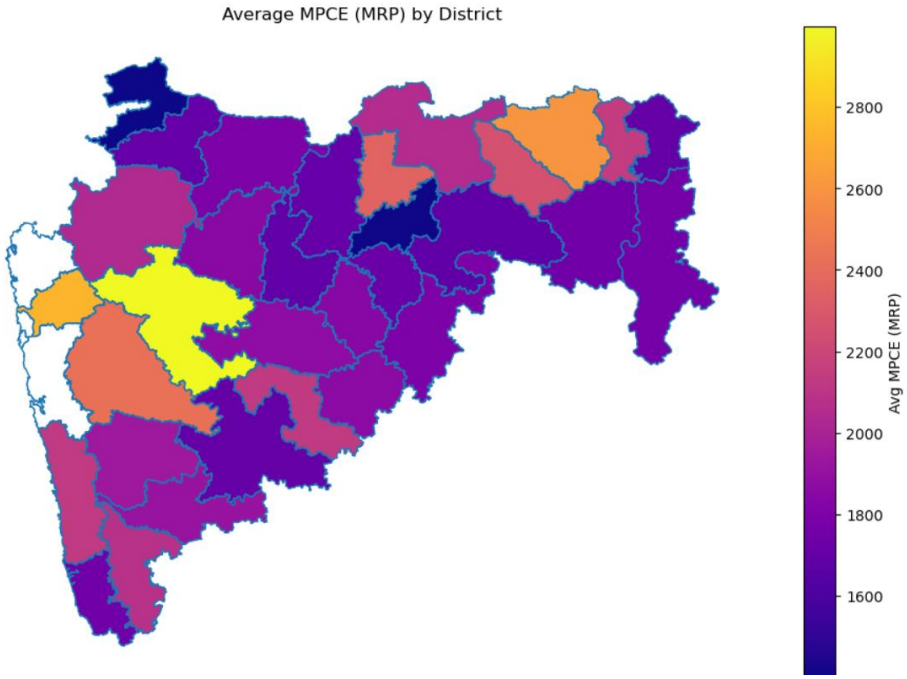
1. **Total Consumption by District:**



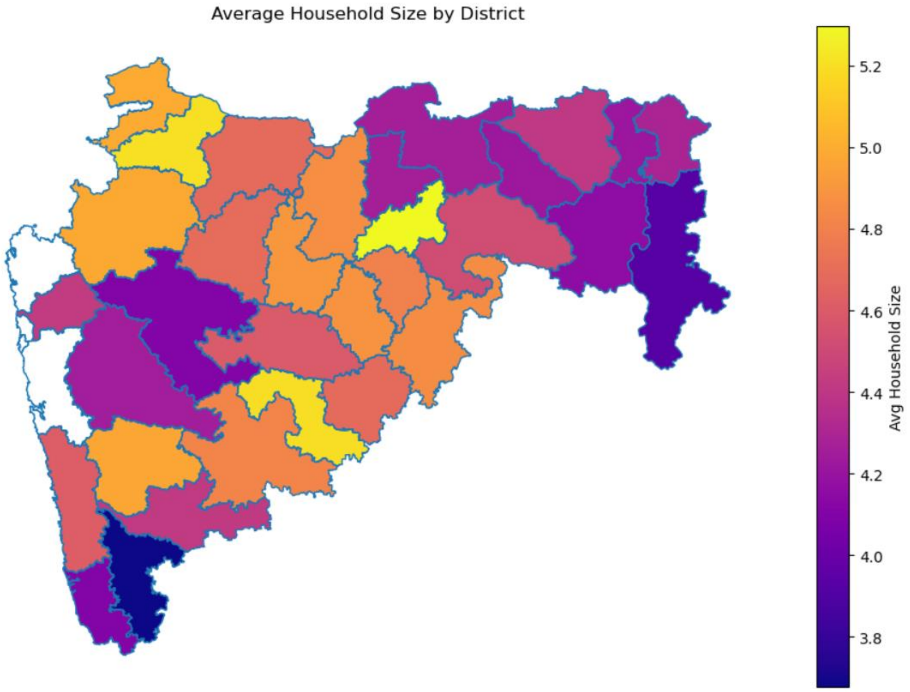
2. **Average MPCE (URP) by District:**



3. Average MPCE (MRP) by District:



4. Average Household Size by District:



Interpretation of Map Outputs

I. **Total Consumption by District:**
   - The map displays total consumption across districts in Maharashtra.
   - Higher consumption areas are indicated by brighter colors (yellow).
   - Notably, Mumbai (Suburban) and Thane exhibit the highest total consumption.

II. **Average MPCE (URP) by District:**

- This map shows the average Monthly Per Capita Expenditure (Uniform Reference Period) by district.
- Brighter areas indicate higher expenditures.
- Mumbai (Suburban) and adjacent districts demonstrate higher MPCE (URP), reflecting greater average expenditure in these regions.
-

III. Average MPCE (MRP) by District:

- Similar to the URP map, but this shows the Monthly Per Capita Expenditure (Mixed Reference Period).
- Again, brighter areas indicate higher expenditures.
- High MPCE (MRP) values are seen in Mumbai (Suburban) and neighboring districts.
-

IV. Average Household Size by District:

- This map shows the average household size by district.
- Brighter colors denote larger household sizes.
- Northern and central districts like Nashik and Ahmednagar exhibit larger average household sizes compared to other regions.

Overview of Visualization: Perceptual Mapping for Business

Perceptual Mapping: Perceptual mapping is a graphical technique used by businesses to visualize the positioning of their products, services, or brands relative to competitors. It involves plotting products on a map based on customer perceptions, typically using two dimensions that represent key attributes or benefits.
Purpose: The primary goal of perceptual mapping is to understand how customers view different products or brands within a market. This helps businesses identify gaps in the market, understand competitive positioning, and develop strategies for differentiation.

Advantages of Perceptual Mapping

1. **Identifies Market Gaps:**
   - Helps in identifying unmet customer needs and potential opportunities for new products or services.
2. **Competitive Analysis:**
   - Provides a visual representation of where a business stands in comparison to its competitors, highlighting strengths and weaknesses.
3. **Informed Decision Making:**
   - Aids in making strategic decisions regarding product development, marketing, and positioning based on customer perceptions.
4. **Customer Insight:**
   - Offers insights into how customers perceive different attributes of products or brands, allowing for more targeted marketing strategies.
5. **Simplifies Complex Data:**
   - Transforms complex market data into an easy-to-understand visual format, making it accessible to stakeholders at all levels.

Real-life Examples of Perceptual Mapping

1. **Automobile Industry:**
   - Car manufacturers often use perceptual maps to position their vehicles based on attributes such as price and quality. For example, luxury brands like BMW and

Mercedes may be positioned high on quality but also on price, while economy brands like Toyota and Honda might be positioned lower on price but still relatively high on quality.

2. **Beverage Industry:**
    - Soft drink companies use perceptual mapping to understand customer perceptions of taste versus healthiness. For instance, a perceptual map might show that customers perceive Coca-Cola as high in taste but low in healthiness, while a brand like Perrier might be seen as high in healthiness but moderate in taste.
3. **Retail Sector:**
    - Retailers use perceptual maps to analyze store locations based on accessibility and product variety. A map might reveal that a store like Walmart is perceived as offering a wide variety of products at convenient locations, while a specialty store might be perceived as offering unique products but with less convenience in terms of location.
4. **Technology Products:**
    - Tech companies use perceptual mapping to position gadgets like smartphones or laptops based on innovation and usability. Apple products might be perceived as highly innovative and user-friendly, while other brands might be positioned differently based on these attributes.