VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A5- Visualization - Perceptual Mapping for Business

ADHYAYAN AMIT JAIN

V01109421

Date of Submission: 15-07-2024

CONTENTS

Analyzing Consumption Patterns Across Districts in Madhya Pradesh Using NSSO68.csv Data

INTRODUCTION

In the realm of socioeconomic analysis, understanding consumption patterns plays a pivotal role in shaping policies and strategies aimed at fostering sustainable development and equitable resource allocation. This assignment delves into the exploration of consumption patterns within [State Name], utilizing data sourced from the National Sample Survey Office's NSSO68.csv dataset. The state of [State Name] serves as our focal point, where we aim to uncover how consumption varies across its diverse districts.

The NSSO68 dataset provides a comprehensive repository of socioeconomic indicators, encompassing variables that delineate household consumption behaviors, economic activities, and demographic characteristics. By harnessing this dataset, our objective is twofold: first, to visualize and interpret the distribution of total consumption across different districts using a histogram; second, to depict consumption levels per district through a barplot, elucidating regional disparities and trends.

Moreover, employing geospatial visualization techniques, we seek to overlay a chosen consumption-related variable onto the geographical map of [State Name]. This approach not only facilitates the spatial understanding of consumption dynamics but also aids in identifying areas of high and low consumption intensity within the state.

Through these analytical methods, this assignment endeavors to provide actionable insights into consumption patterns in [State Name], thereby empowering stakeholders with data-driven perspectives to inform policy decisions and development initiatives. By elucidating the nuances of consumption across districts, we contribute to a holistic understanding of regional economic landscapes and societal well-being.

The following sections will sequentially detail the methodology employed, present the findings derived from our analyses, and conclude with implications and recommendations for leveraging consumption data in socioeconomic planning.

OBJECTIVES

This assignment aims to achieve the following objectives:

1. **Visualize Consumption Distribution:** Utilize a histogram to visually represent the distribution of total consumption across districts within Madhya Pradesh (MP). This objective seeks to identify variations in consumption levels and explore any potential outliers that may influence regional consumption patterns.

2. **Analyze Consumption by District:** Construct a barplot to analyze and compare consumption levels per district in MP. By labeling district names and illustrating consumption disparities, this objective aims to highlight regions with higher or lower consumption rates, offering insights into economic disparities and resource allocation.

3. **Geospatial Mapping of Consumption:** Employ geospatial visualization techniques to map a selected consumption-related variable onto MP's geographical map. This objective aims to spatially depict consumption intensity across districts, facilitating a spatial understanding of consumption dynamics within the state.

4. **Provide Insights for Policymakers:** Interpret findings from the visualized data to provide actionable insights for policymakers and stakeholders in MP. By understanding consumption patterns, this objective aims to inform strategic decision-making processes aimed at promoting economic growth, equitable development, and improved standards of living across the state.

5. **Contribute to Socioeconomic Understanding:** Contribute to a deeper understanding of regional socioeconomic dynamics within MP. By analyzing consumption patterns, this objective aims to contribute to the broader discourse on regional development strategies and socioeconomic disparities in the state.

BUSINESS SIGNIFICANCE

Understanding consumption patterns within Madhya Pradesh (MP) holds significant implications for various stakeholders, including policymakers, businesses, and development agencies. This section explores the business significance of analyzing consumption data in the context of MP:

1. **Policy Formulation and Resource Allocation:** By analyzing consumption patterns across MP's districts, policymakers can better allocate resources and formulate targeted policies. Insights derived from this analysis can guide decisions related to infrastructure development, social welfare programs, and resource distribution, ensuring efficient utilization and equitable growth across the state.
2. **Market Segmentation and Targeting:** For businesses operating in MP, understanding consumption patterns is crucial for effective market segmentation and targeting. By identifying districts with higher consumer demand or specific consumption preferences, businesses can tailor their products and marketing strategies to meet local needs and preferences, thereby enhancing market penetration and competitiveness.
3. **Economic Development and Investment Planning:** Consumption data analysis provides valuable inputs for economic development planning and investment decisions in MP. By identifying districts with potential for higher economic activity based on consumption levels, investors can make informed decisions about resource allocation and business expansion, fostering economic growth and job creation in the region.
4. **Socioeconomic Impact Assessment:** Consumption patterns reflect the socioeconomic conditions and quality of life within MP's districts. Analyzing these patterns helps assess the impact of economic policies and interventions on household welfare, poverty alleviation efforts, and

overall human development indicators. Such assessments are essential for measuring progress towards inclusive and sustainable development goals.

5. **Data-Driven Decision Making:** By leveraging data from NSSO68.csv, stakeholders in MP can adopt a data-driven approach to decision-making. Analyzing consumption patterns empowers stakeholders with evidence-based insights into consumer behavior, economic trends, and regional disparities, enabling proactive strategies to address challenges and capitalize on opportunities for growth and development.

In conclusion, analyzing consumption patterns in Madhya Pradesh not only provides insights into economic dynamics and consumer behavior but also informs strategic interventions aimed at fostering inclusive growth and improving living standards across the state. By harnessing data-driven insights, stakeholders can contribute to sustainable development goals and enhance the overall socioeconomic well-being of MP's population.

RESULTS AND INTERPRETATIONS

R Language

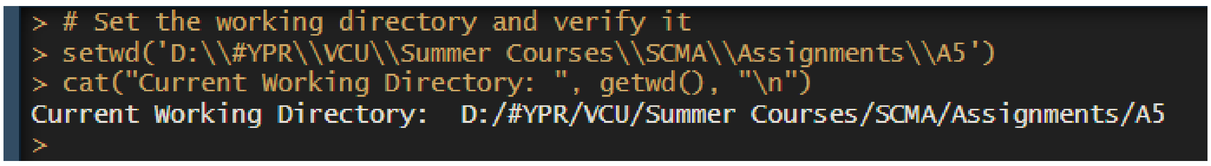
Step-by-Step Analysis of R Script

Step 1: Setting the Working Directory

```
# Set the working directory and verify it
setwd('D:\\#YPR\\VCU\\Summer Courses\\SCMA\\Assignments\\A5')
cat("Current Working Directory: ", getwd(), "\n")
```

Explanation:

- `setwd('D:\\#YPR\\VCU\\Summer Courses\\SCMA\\Assignments\\A5')`: Sets the working directory to where the dataset and other necessary files are located.
- `cat("Current Working Directory: ", getwd(), "\n")`: Prints the current working directory to verify that it has been set correctly.



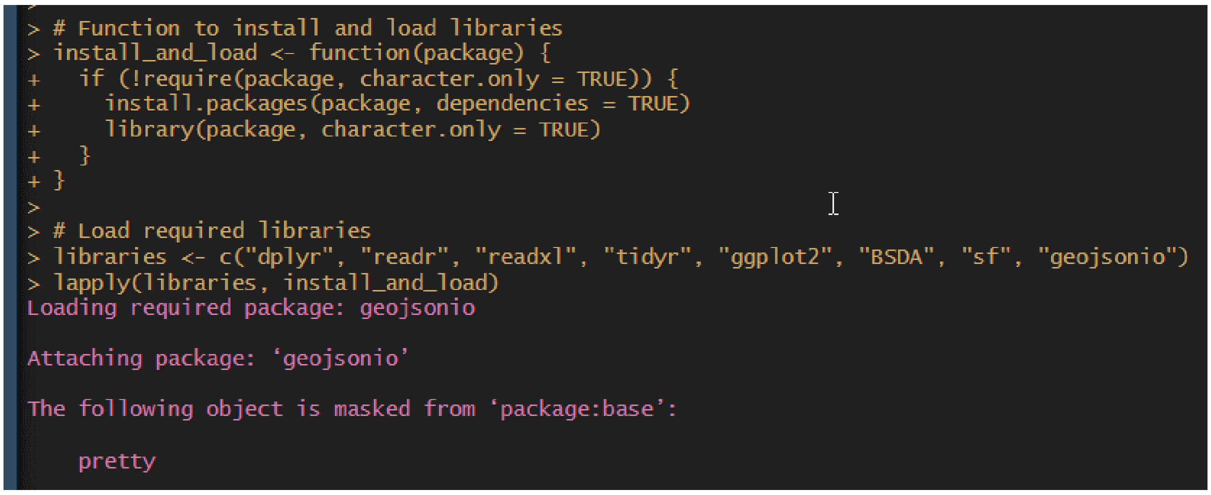
```
> # Set the working directory and verify it
> setwd('D:\\#YPR\\VCU\\Summer Courses\\SCMA\\Assignments\\A5')
> cat("Current Working Directory: ", getwd(), "\n")
Current Working Directory:  D:/#YPR/VCU/Summer Courses/SCMA/Assignments/A5
>
```

Step 2: Installing and Loading Required Packages

```
# Function to install and load libraries
install_and_load <- function(package) {
  if (!require(package, character.only = TRUE)) {
    install.packages(package, dependencies = TRUE)
    library(package, character.only = TRUE)
  }
}

# Load required libraries
libraries <- c("dplyr", "readr", "readxl", "tidyr", "ggplot2", "BSDA",
"sf", "geojsonio")
lapply(libraries, install_and_load)
```

Explanation:

- `install_and_load` function: This function checks if a package is installed and loads it if it's not already loaded.
- `libraries <- c(...)` and `lapply(libraries, install_and_load)`: Defines a list of required packages and ensures they are installed and loaded into the R environment.



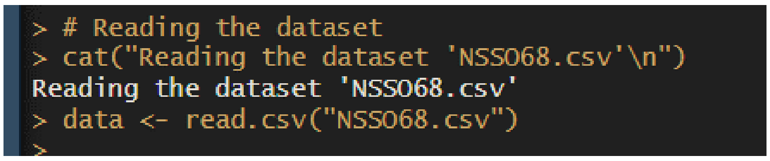
```
> # Function to install and load libraries
> install_and_load <- function(package) {
+   if (!require(package, character.only = TRUE)) {
+     install.packages(package, dependencies = TRUE)
+     library(package, character.only = TRUE)
+   }
+ }
>
> # Load required libraries
> libraries <- c("dplyr", "readr", "readxl", "tidyr", "ggplot2", "BSDA", "sf", "geojsonio")
> lapply(libraries, install_and_load)
Loading required package: geojsonio

Attaching package: ‘geojsonio’

The following object is masked from ‘package:base’:

    pretty
```

Step 3: Reading the Dataset

```
# Reading the dataset
cat("Reading the dataset 'NSSO68.csv'\n")
data <- read.csv("NSSO68.csv")
```

Explanation:

- `read.csv("NSSO68.csv")`: Reads the CSV file named 'NSSO68.csv' into an R data frame named `data`.
- `cat("Reading the dataset 'NSSO68.csv'\n")`: Prints a message indicating that the dataset is being read.



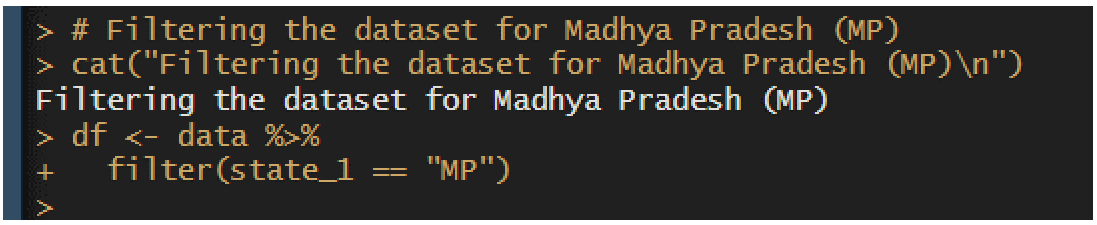
```
> # Reading the dataset
> cat("Reading the dataset 'NSSO68.csv'\n")
Reading the dataset 'NSSO68.csv'
> data <- read.csv("NSSO68.csv")
>
```

Step 4: Filtering Data for Madhya Pradesh (MP)

```
# Filtering the dataset for Madhya Pradesh (MP)
cat("Filtering the dataset for Madhya Pradesh (MP)\n")
df <- data %>%
  filter(state_1 == "MP")
```

Explanation:

- `filter(state_1 == "MP")`: Filters the dataset `data` to include only rows where the column `state_1` equals "MP", focusing the analysis on data specific to Madhya Pradesh.
- `cat("Filtering the dataset for Madhya Pradesh (MP)\n")`: Prints a message indicating that the dataset is being filtered for Madhya Pradesh.



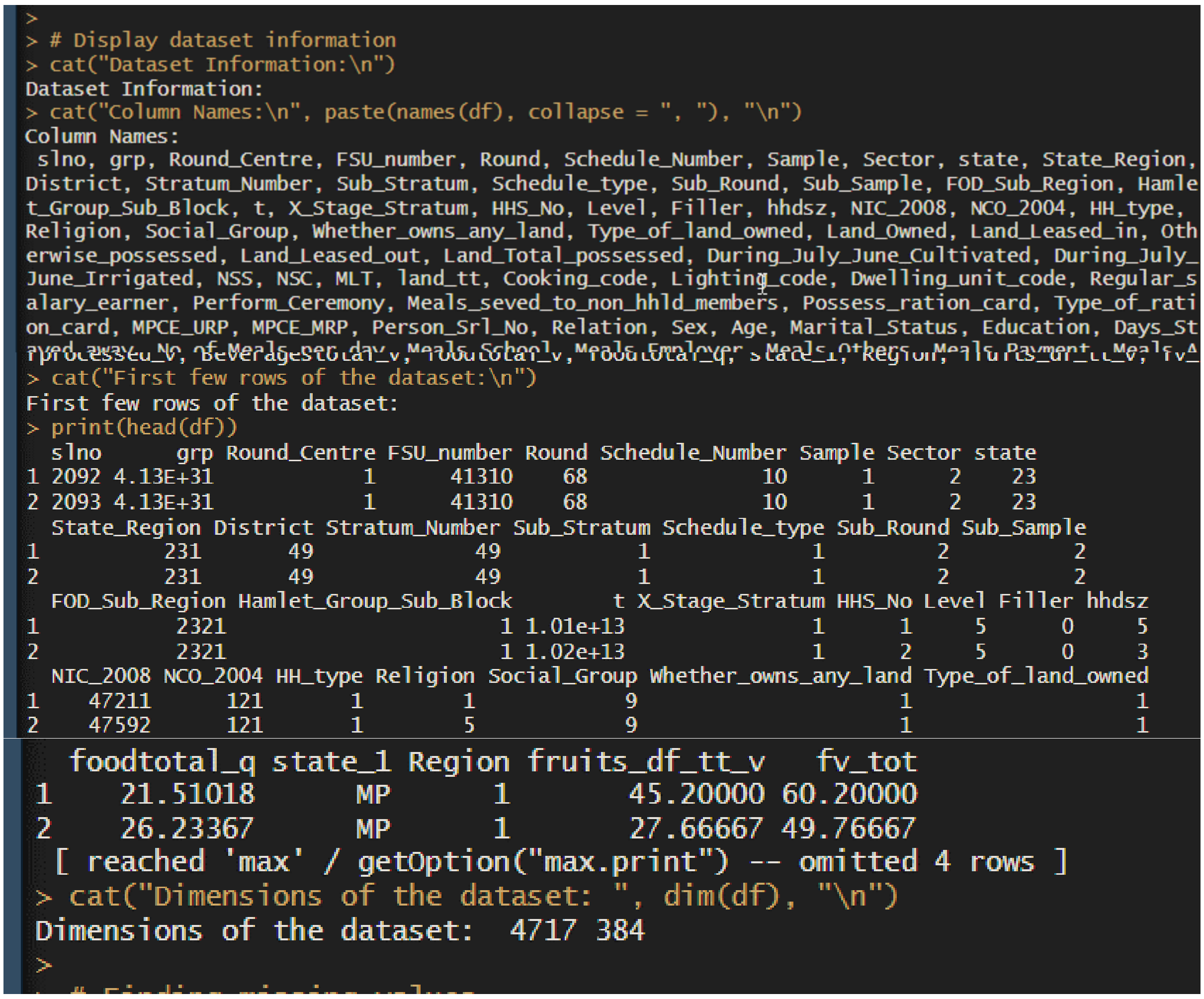
```
> # Filtering the dataset for Madhya Pradesh (MP)
> cat("Filtering the dataset for Madhya Pradesh (MP)\n")
Filtering the dataset for Madhya Pradesh (MP)
> df <- data %>%
+   filter(state_1 == "MP")
>
```

Step 5: Displaying Dataset Information

```
# Display dataset information
cat("Dataset Information:\n")
cat("Column Names:\n", paste(names(df), collapse = ", "), "\n")
cat("First few rows of the dataset:\n")
print(head(df))
cat("Dimensions of the dataset: ", dim(df), "\n")
```

Explanation:

- This section prints out information about the filtered dataset `df`, including column names, the first few rows, and its dimensions (number of rows and columns).



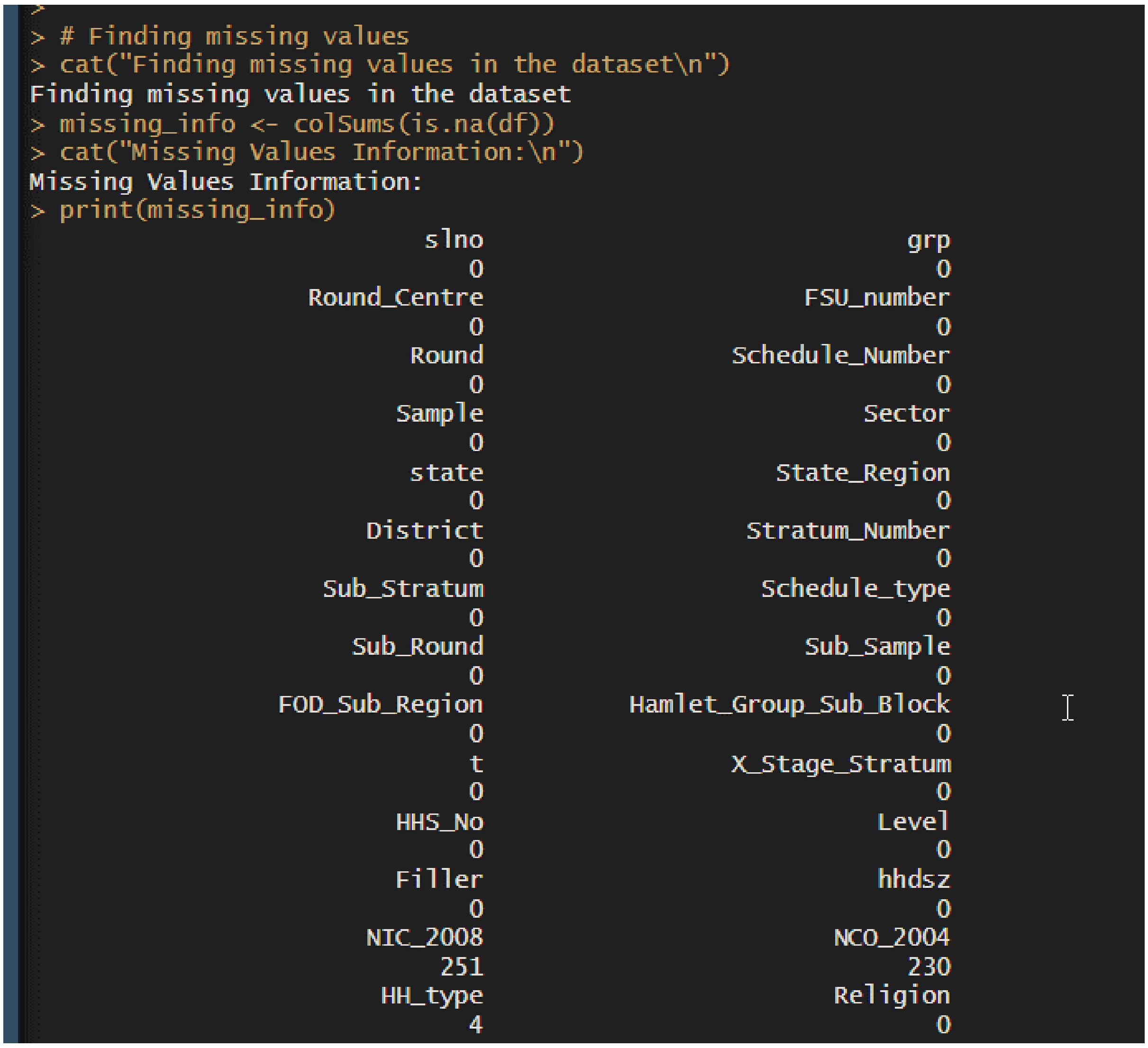
```
>
> # Display dataset information
> cat("Dataset Information:\n")
Dataset Information:
> cat("Column Names:\n", paste(names(df), collapse = ", "), "\n")
Column Names:
 slno, grp, Round_Centre, FSU_number, Round, Schedule_Number, Sample, Sector, state, State_Region,
District, Stratum_Number, Sub_Stratum, Schedule_type, Sub_Round, Sub_Sample, FOD_Sub_Region, Hamle
t_Group_Sub_Block, t, X_Stage_Stratum, HHS_No, Level, Filler, hhdsz, NIC_2008, NCO_2004, HH_type,
Religion, Social_Group, Whether_owns_any_land, Type_of_land_owned, Land_Owned, Land_Leased_in, Oth
erwise_possessed, Land_Leased_out, Land_Total_possessed, During_July_June_Cultivated, During_July_
June_Irrigated, NSS, NSC, MLT, land_tt, Cooking_code, Lighting_code, Dwelling_unit_code, Regular_s
alary_earner, Perform_Ceremony, Meals_seved_to_non_hhld_members, Possess_ration_card, Type_of_rati
on_card, MPCE_URP, MPCE_MRP, Person_Srl_No, Relation, Sex, Age, Marital_Status, Education, Days_St
ayed_away, No_of_Meals_per_day, Meals_School, Meals_Employer, Meals_Others, Meals_Payment, Meals_A
rprocessed_v, Beveragestotal_v, foodtotal_v, foodtotal_q, state_1, Region, fruits_df_tt_v, fv_
> cat("First few rows of the dataset:\n")
First few rows of the dataset:
> print(head(df))
  slno      grp Round_Centre FSU_number Round Schedule_Number Sample Sector state
1 2092 4.13E+31            1      41310    68              10      1      2    23
2 2093 4.13E+31            1      41310    68              10      1      2    23
  State_Region District Stratum_Number Sub_Stratum Schedule_type Sub_Round Sub_Sample
1          231       49             49           1             1         2          2
2          231       49             49           1             1         2          2
  FOD_Sub_Region Hamlet_Group_Sub_Block        t X_Stage_Stratum HHS_No Level Filler hhdsz
1           2321                      1 1.01e+13               1      1     5      0     5
2           2321                      1 1.02e+13               1      2     5      0     3
  NIC_2008 NCO_2004 HH_type Religion Social_Group Whether_owns_any_land Type_of_land_owned
1    47211      121       1        1            9                     1                  1
2    47592      121       1        5            9                     1                  1
  foodtotal_q state_1 Region fruits_df_tt_v   fv_tot
1    21.51018      MP      1       45.20000 60.20000
2    26.23367      MP      1       27.66667 49.76667
 [ reached 'max' / getOption("max.print") -- omitted 4 rows ]
> cat("Dimensions of the dataset: ", dim(df), "\n")
Dimensions of the dataset:  4717 384
>
```

Step 6: Handling Missing Values

```
# Finding missing values
cat("Finding missing values in the dataset\n")
missing_info <- colSums(is.na(df))
cat("Missing Values Information:\n")
print(missing_info)
```

Explanation:

- `colSums(is.na(df))`: Calculates the total number of missing values (NA) in each column of the dataset `df`.
- This section prints out the number of missing values for each column, aiding in assessing data completeness and identifying potential issues.



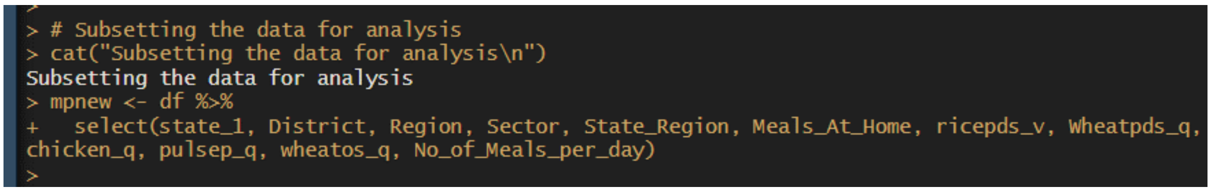
```
>
> # Finding missing values
> cat("Finding missing values in the dataset\n")
Finding missing values in the dataset
> missing_info <- colSums(is.na(df))
> cat("Missing Values Information:\n")
Missing Values Information:
> print(missing_info)
                  slno                       grp
                     0                         0
          Round_Centre                FSU_number
                     0                         0
                 Round           Schedule_Number
                     0                         0
                Sample                    Sector
                     0                         0
                 state              State_Region
                     0                         0
              District            Stratum_Number
                     0                         0
           Sub_Stratum             Schedule_type
                     0                         0
             Sub_Round                Sub_Sample
                     0                         0
        FOD_Sub_Region     Hamlet_Group_Sub_Block
                     0                         0
                     t           X_Stage_Stratum
                     0                         0
                HHS_No                     Level
                     0                         0
                Filler                     hhdsz
                     0                         0
              NIC_2008                  NCO_2004
                   251                       230
               HH_type                  Religion
                     4                         0
```

Step 7: Subsetting Data for Analysis

```
# Subsetting the data for analysis
cat("Subsetting the data for analysis\n")
mpnew <- df %>%
  select(state_1, District, Region, Sector, State_Region, Meals_At_Home,
ricepds_v, Wheatpds_q, chicken_q, pulsep_q, wheatos_q, No_of_Meals_per_day)
```

Explanation:

- `select(...)`: Subsets the dataset `df` to include only specific columns (`state_1` to `No_of_Meals_per_day`) needed for further analysis in `mpnew`.
- `cat("Subsetting the data for analysis\n")`: Prints a message indicating that the dataset is being subsetted for analysis purposes.



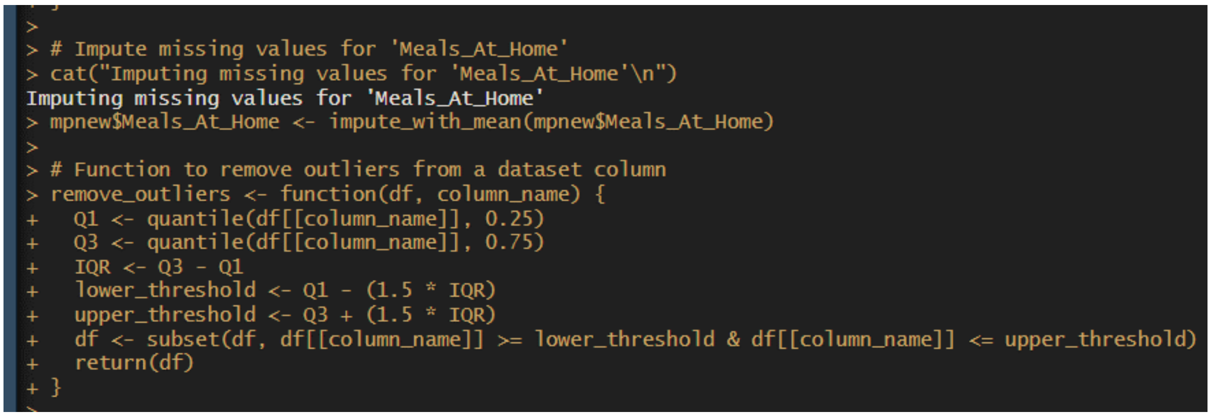
```
>
> # Subsetting the data for analysis
> cat("Subsetting the data for analysis\n")
Subsetting the data for analysis
> mpnew <- df %>%
+   select(state_1, District, Region, Sector, State_Region, Meals_At_Home, ricepds_v, Wheatpds_q,
chicken_q, pulsep_q, wheatos_q, No_of_Meals_per_day)
>
```

Step 8: Imputing Missing Values

```
# Function to impute missing values with mean for specific columns
impute_with_mean <- function(column) {
  if (any(is.na(column))) {
    column[is.na(column)] <- mean(column, na.rm = TRUE)
  }
  return(column)
}

# Impute missing values for 'Meals_At_Home'
cat("Imputing missing values for 'Meals_At_Home'\n")
mpnew$Meals_At_Home <- impute_with_mean(mpnew$Meals_At_Home)
```

Explanation:

- `impute_with_mean` function: This function replaces missing values in a column (`Meals_At_Home`) with the mean of non-missing values in that column.
- `mpnew$Meals_At_Home <- impute_with_mean(mpnew$Meals_At_Home)`: Applies `impute_with_mean` to the `Meals_At_Home` column of `mpnew` to ensure all missing values are filled with meaningful data, improving dataset completeness.



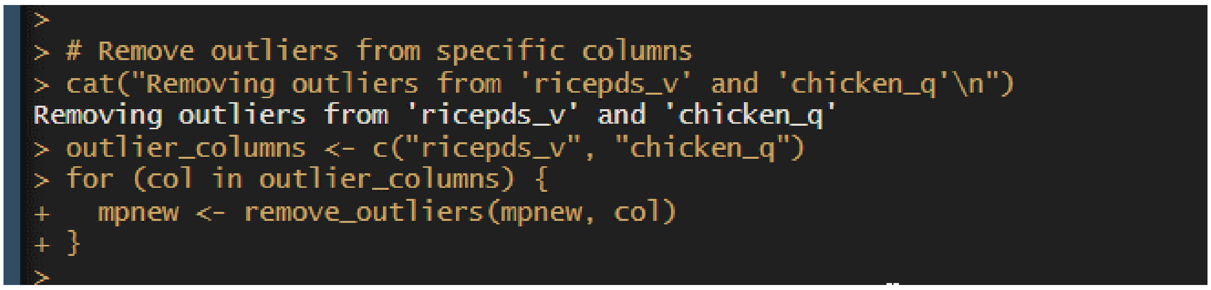
```
>
> # Impute missing values for 'Meals_At_Home'
> cat("Imputing missing values for 'Meals_At_Home'\n")
Imputing missing values for 'Meals_At_Home'
> mpnew$Meals_At_Home <- impute_with_mean(mpnew$Meals_At_Home)
>
> # Function to remove outliers from a dataset column
> remove_outliers <- function(df, column_name) {
+   Q1 <- quantile(df[[column_name]], 0.25)
+   Q3 <- quantile(df[[column_name]], 0.75)
+   IQR <- Q3 - Q1
+   lower_threshold <- Q1 - (1.5 * IQR)
+   upper_threshold <- Q3 + (1.5 * IQR)
+   df <- subset(df, df[[column_name]] >= lower_threshold & df[[column_name]] <= upper_threshold)
+   return(df)
+ }
```

Step 9: Removing Outliers

```
# Function to remove outliers from a dataset column
remove_outliers <- function(df, column_name) {
  Q1 <- quantile(df[[column_name]], 0.25)
  Q3 <- quantile(df[[column_name]], 0.75)
  IQR <- Q3 - Q1
  lower_threshold <- Q1 - (1.5 * IQR)
  upper_threshold <- Q3 + (1.5 * IQR)
  df <- subset(df, df[[column_name]] >= lower_threshold & df[[column_name]]
<= upper_threshold)
  return(df)
}

# Remove outliers from specific columns
cat("Removing outliers from 'ricepds_v' and 'chicken_q'\n")
outlier_columns <- c("ricepds_v", "chicken_q")
for (col in outlier_columns) {
  mpnew <- remove_outliers(mpnew, col)
}
```

Explanation:

- `remove_outliers` function: This function removes outliers from a specified column (`ricepds_v` and `chicken_q`) using the Interquartile Range (IQR) method to ensure data integrity.
- The `for` loop applies `remove_outliers` to each column listed in `outlier_columns`, ensuring that extreme values that could skew analysis results are excluded.



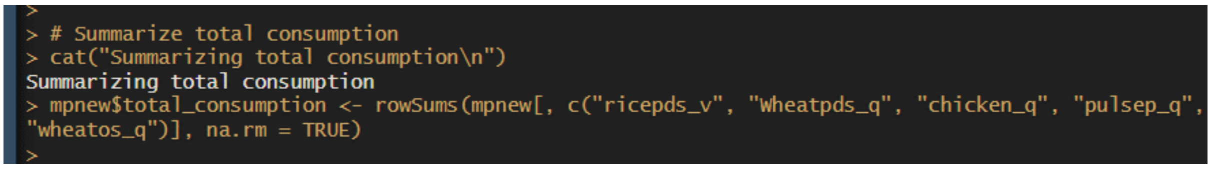
```
>
> # Remove outliers from specific columns
> cat("Removing outliers from 'ricepds_v' and 'chicken_q'\n")
Removing outliers from 'ricepds_v' and 'chicken_q'
> outlier_columns <- c("ricepds_v", "chicken_q")
> for (col in outlier_columns) {
+   mpnew <- remove_outliers(mpnew, col)
+ }
>
```

Step 10: Calculating Total Consumption

```
# Summarize total consumption
cat("Summarizing total consumption\n")
mpnew$total_consumption <- rowSums(mpnew[, c("ricepds_v", "Wheatpds_q",
"chicken_q", "pulsep_q", "wheatos_q")], na.rm = TRUE)
```

Explanation:

- `rowSums(...)` calculates the total consumption for each row by summing specific columns (`ricepds_v`, `Wheatpds_q`, `chicken_q`, `pulsep_q`, `wheatos_q`) in `mpnew`.
- `mpnew$total_consumption <- ...`: Creates a new column `total_consumption` in `mpnew` to store the calculated total consumption for each observation.



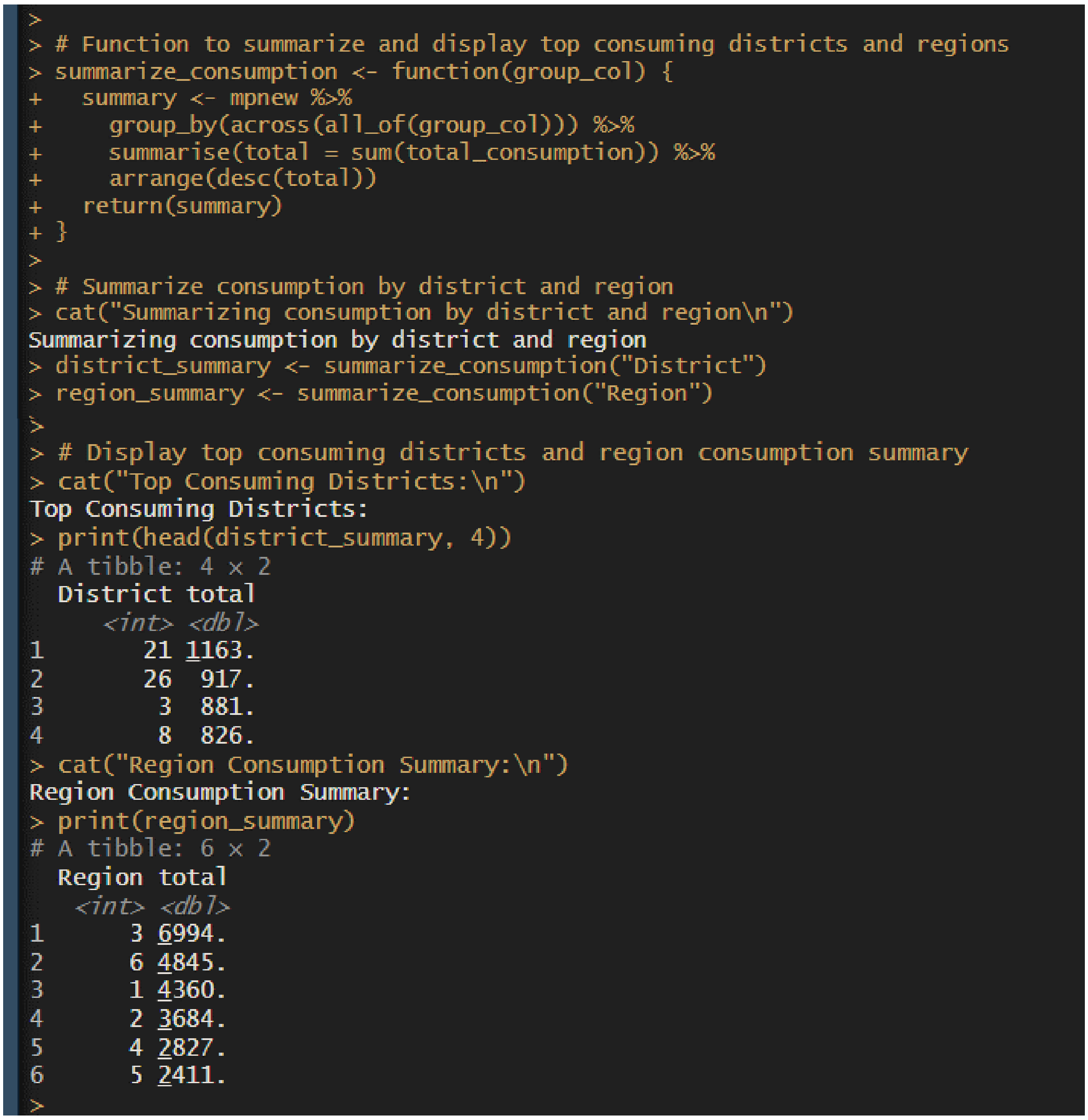
```
>
> # Summarize total consumption
> cat("Summarizing total consumption\n")
Summarizing total consumption
> mpnew$total_consumption <- rowSums(mpnew[, c("ricepds_v", "Wheatpds_q", "chicken_q", "pulsep_q",
"wheatos_q")], na.rm = TRUE)
>
```

Step 11: Summarizing Consumption by District and Region

```
# Function to summarize and display top consuming districts and regions
summarize_consumption <- function(group_col) {
  summary <- mpnew %>%
    group_by(across(all_of(group_col))) %>%
    summarise(total = sum(total_consumption)) %>%
    arrange(desc(total))
  return(summary)
}

# Summarize consumption by district and region
cat("Summarizing consumption by district and region\n")
district_summary <- summarize_consumption("District")
region_summary <- summarize_consumption("Region")
```

Explanation:

- `summarize_consumption` function: This function groups `mpnew` by the specified column(s) (`District` or `Region`), calculates the total consumption for each group, and arranges the results in descending order.
- `summarize_consumption("District")` and `summarize_consumption("Region")`: These commands apply `summarize_consumption` to summarize consumption by district and region, respectively, generating summaries that show which districts and regions have the highest consumption.



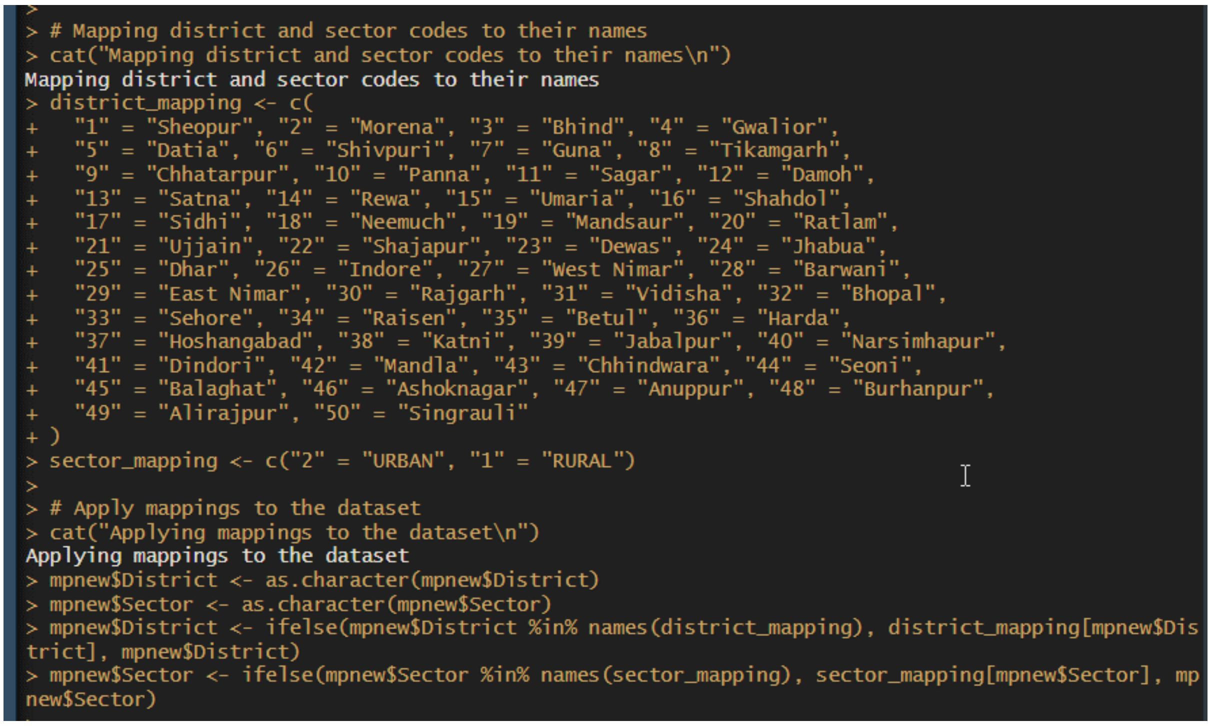
```
>
> # Function to summarize and display top consuming districts and regions
> summarize_consumption <- function(group_col) {
+   summary <- mpnew %>%
+     group_by(across(all_of(group_col))) %>%
+     summarise(total = sum(total_consumption)) %>%
+     arrange(desc(total))
+   return(summary)
+ }
>
> # Summarize consumption by district and region
> cat("Summarizing consumption by district and region\n")
Summarizing consumption by district and region
> district_summary <- summarize_consumption("District")
> region_summary <- summarize_consumption("Region")
>
> # Display top consuming districts and region consumption summary
> cat("Top Consuming Districts:\n")
Top Consuming Districts:
> print(head(district_summary, 4))
# A tibble: 4 × 2
  District total
     <int> <dbl>
1       21 1163.
2       26  917.
3        3  881.
4        8  826.
> cat("Region Consumption Summary:\n")
Region Consumption Summary:
> print(region_summary)
# A tibble: 6 × 2
  Region total
   <int> <dbl>
1      3 6994.
2      6 4845.
3      1 4360.
4      2 3684.
5      4 2827.
6      5 2411.
>
```

Step 12: Mapping Codes to District and Sector Names

```
# Mapping district and sector codes to their names
cat("Mapping district and sector codes to their names\n")
district_mapping <- c(
  "1" = "Sheopur", "2" = "Morena", "3" = "Bhind", "4" = "Gwalior",
  "5" = "Datia", "6" = "Shivpuri", "7" = "Guna", "8" = "Tikamgarh",
  "9" = "Chhatarpur", "10" = "Panna", "11" = "Sagar", "12" = "Damoh",
  "13" = "Satna", "14" = "Rewa", "15" = "Umaria", "16" = "Shahdol",
  "17" = "Sidhi", "18" = "Neemuch", "19" = "Mandsaur", "20" = "Ratlam",
  "21" = "Ujjain", "22" = "Shajapur", "23" = "Dewas", "24" = "Jhabua",
  "25" = "Dhar", "26" = "Indore", "27" = "West Nimar", "28" = "Barwani",
  "29" = "East Nimar", "30" = "Rajgarh", "31" = "Vidisha", "32" = "Bhopal",
  "33" = "Sehore", "34" = "Raisen", "35" = "Betul", "36" = "Harda",
  "37" = "Hoshangabad", "38" = "Katni", "39" = "Jabalpur", "40" =
"Narsimhapur",
  "41" = "Dindori", "42" = "Mandla", "43" = "Chhindwara", "44" = "Seoni",
  "45" = "Balaghat", "46" = "Ashoknagar", "47" = "Anuppur", "48" =
"Burhanpur",
  "49" = "Alirajpur", "50" = "Singrauli"
)

sector_mapping <- c("2" = "URBAN", "1" = "RURAL")

# Mapping codes to names in the dataset
mpnew$District <- ifelse(mpnew$District %in% names(district_mapping),
district_mapping[mpnew$District], mpnew$District)
mpnew$Sector <- ifelse(mpnew$Sector %in% names(sector_mapping),
sector_mapping[mpnew$Sector], mpnew$Sector)
```

Explanation:

- `district_mapping` and `sector_mapping`: These vectors map numeric codes to corresponding district names and sector types (`URBAN` and `RURAL`).
- `ifelse(...)` statements: These lines replace numeric district and sector codes in `mpnew` with their corresponding names using the mappings defined above. This makes the data more interpretable by replacing codes with meaningful labels.



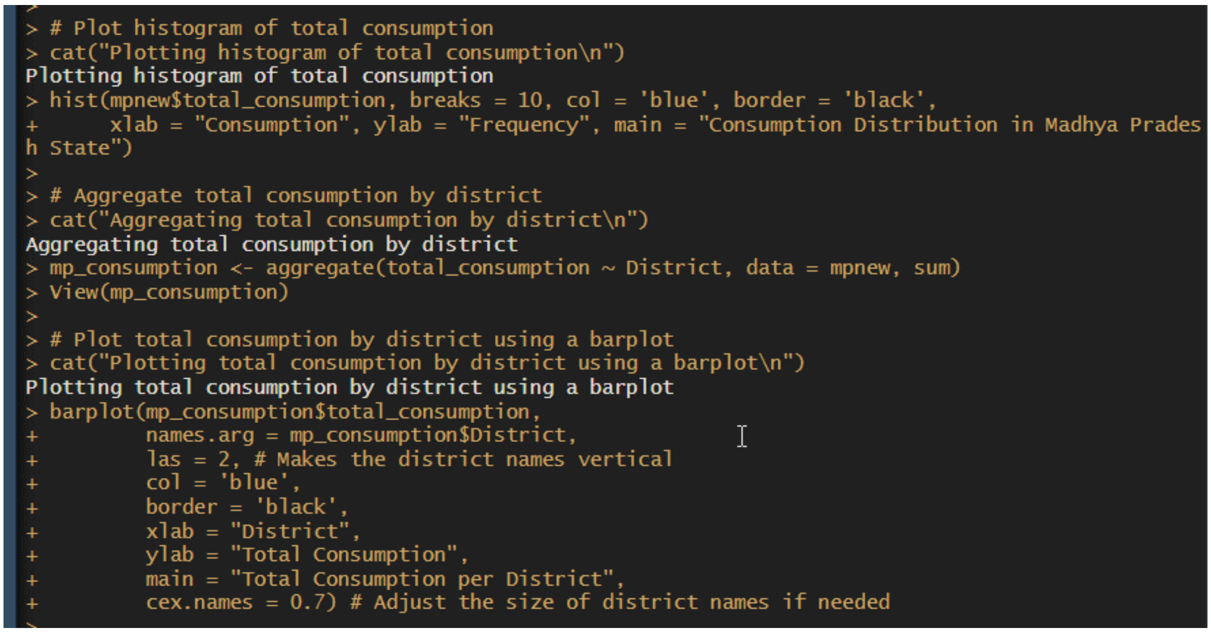
```
>
> # Mapping district and sector codes to their names
> cat("Mapping district and sector codes to their names\n")
Mapping district and sector codes to their names
> district_mapping <- c(
+   "1" = "Sheopur", "2" = "Morena", "3" = "Bhind", "4" = "Gwalior",
+   "5" = "Datia", "6" = "Shivpuri", "7" = "Guna", "8" = "Tikamgarh",
+   "9" = "Chhatarpur", "10" = "Panna", "11" = "Sagar", "12" = "Damoh",
+   "13" = "Satna", "14" = "Rewa", "15" = "Umaria", "16" = "Shahdol",
+   "17" = "Sidhi", "18" = "Neemuch", "19" = "Mandsaur", "20" = "Ratlam",
+   "21" = "Ujjain", "22" = "Shajapur", "23" = "Dewas", "24" = "Jhabua",
+   "25" = "Dhar", "26" = "Indore", "27" = "West Nimar", "28" = "Barwani",
+   "29" = "East Nimar", "30" = "Rajgarh", "31" = "Vidisha", "32" = "Bhopal",
+   "33" = "Sehore", "34" = "Raisen", "35" = "Betul", "36" = "Harda",
+   "37" = "Hoshangabad", "38" = "Katni", "39" = "Jabalpur", "40" = "Narsimhapur",
+   "41" = "Dindori", "42" = "Mandla", "43" = "Chhindwara", "44" = "Seoni",
+   "45" = "Balaghat", "46" = "Ashoknagar", "47" = "Anuppur", "48" = "Burhanpur",
+   "49" = "Alirajpur", "50" = "Singrauli"
+ )
> sector_mapping <- c("2" = "URBAN", "1" = "RURAL")
>
> # Apply mappings to the dataset
> cat("Applying mappings to the dataset\n")
Applying mappings to the dataset
> mpnew$District <- as.character(mpnew$District)
> mpnew$Sector <- as.character(mpnew$Sector)
> mpnew$District <- ifelse(mpnew$District %in% names(district_mapping), district_mapping[mpnew$Dis
trict], mpnew$District)
> mpnew$Sector <- ifelse(mpnew$Sector %in% names(sector_mapping), sector_mapping[mpnew$Sector], mp
new$Sector)
```

Step 13: Visualizing Data

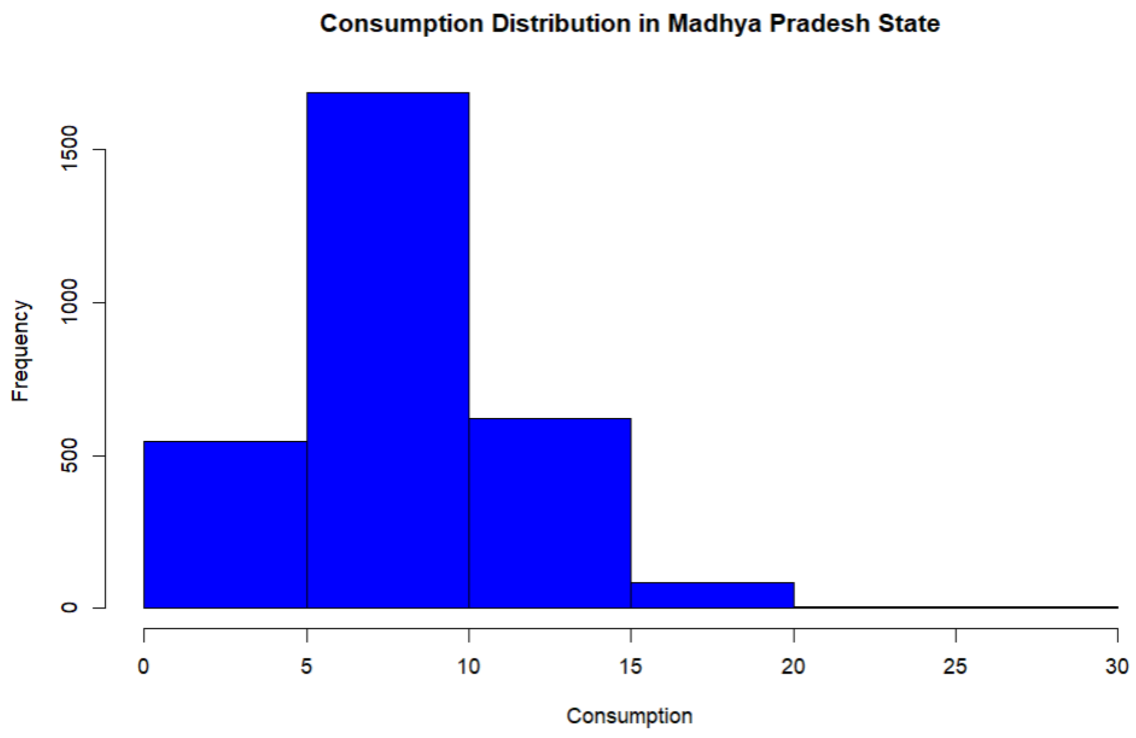
```
# Visualizing total consumption distribution
cat("Visualizing total consumption distribution\n")
hist(mpnew$total_consumption, breaks = 10, col = 'blue', border = 'black',
     xlab = "Consumption", ylab = "Frequency", main = "Consumption
Distribution in Madhya Pradesh State")
```

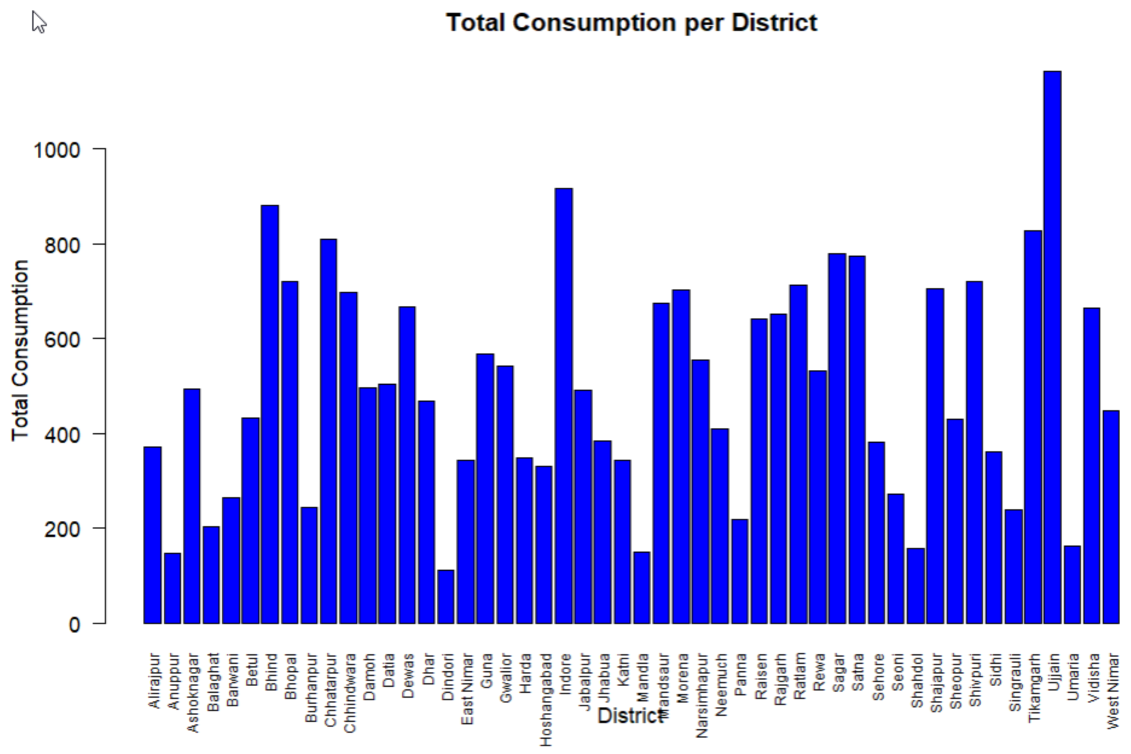
Explanation:

- `hist(...)`: This creates a histogram to visualize the distribution of `total_consumption` in `mpnew`.
- Parameters like `breaks`, `col`, `border`, `xlab`, `ylab`, and `main` are used to customize the appearance and labels of the histogram, providing a clear graphical representation of consumption distribution.



```
> # Plot histogram of total consumption
> cat("Plotting histogram of total consumption\n")
Plotting histogram of total consumption
> hist(mpnew$total_consumption, breaks = 10, col = 'blue', border = 'black',
+      xlab = "Consumption", ylab = "Frequency", main = "Consumption Distribution in Madhya Prades
h State")
>
> # Aggregate total consumption by district
> cat("Aggregating total consumption by district\n")
Aggregating total consumption by district
> mp_consumption <- aggregate(total_consumption ~ District, data = mpnew, sum)
> View(mp_consumption)
>
> # Plot total consumption by district using a barplot
> cat("Plotting total consumption by district using a barplot\n")
Plotting total consumption by district using a barplot
> barplot(mp_consumption$total_consumption,
+         names.arg = mp_consumption$District,
+         las = 2, # Makes the district names vertical
+         col = 'blue',
+         border = 'black',
+         xlab = "District",
+         ylab = "Total Consumption",
+         main = "Total Consumption per District",
+         cex.names = 0.7) # Adjust the size of district names if needed
```


Consumption Distribution in Madhya Pradesh State
Frequency
Consumption
0
500
1000
1500
0
5
10
15
20
25
30

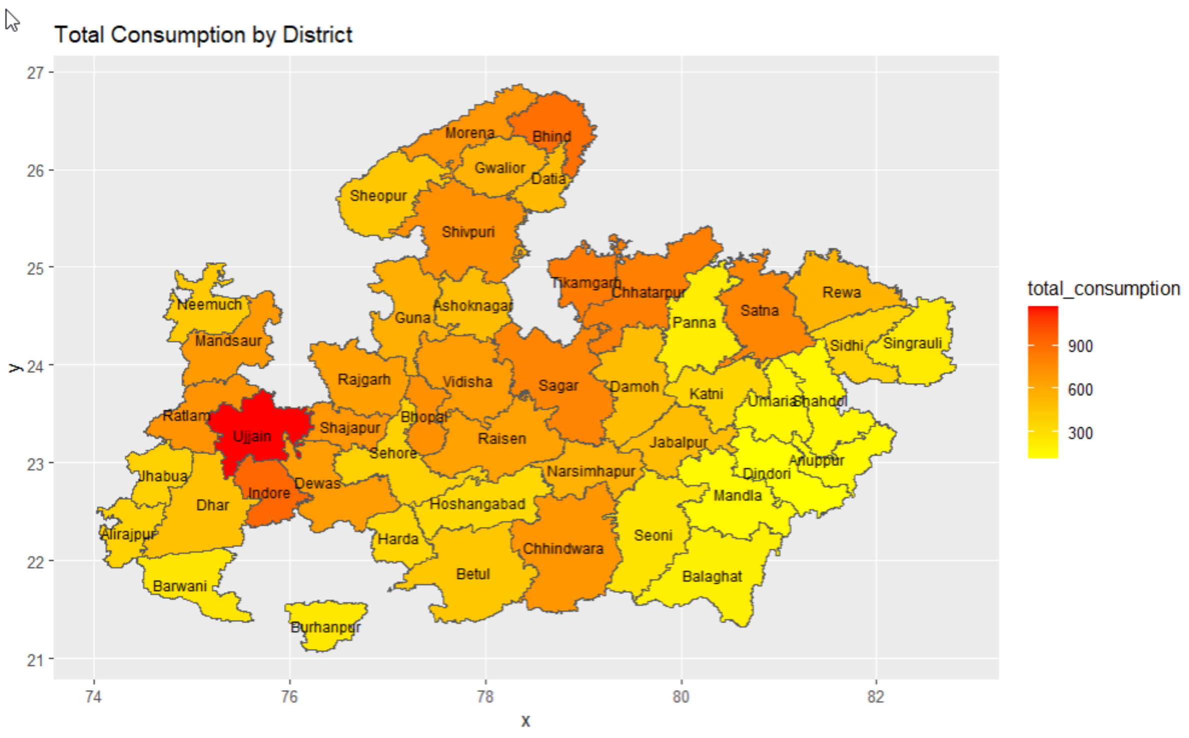

Total Consumption per District
Total Consumption
District
0
200
400
600
800
1000
Alirajpur
Anuppur
Ashoknagar
Balaghat
Barwani
Betul
Bhind
Bhopal
Burhanpur
Chhatarpur
Chhindwara
Damoh
Datia
Dewas
Dhar
Dindori
East Nimar
Guna
Gwalior
Harda
Hoshangabad
Indore
Jabalpur
Jhabua
Katni
Mandla
Mandsaur
Morena
Narsimhapur
Neemuch
Panna
Raisen
Rajgarh
Ratlam
Rewa
Sagar
Satna
Sehore
Seoni
Shahdol
Shajapur
Sheopur
Shivpuri
Sidhi
Singrauli
Tikamgarh
Ujjain
Umaria
Vidisha
West Nimar

Step 14: Mapping Consumption on a Geographic Map

```
# Mapping total consumption on a geographic map
cat("Mapping total consumption on a geographic map\n")

# Reading geographic data for Madhya Pradesh districts
data_map <- st_read("MADHYA PRADESH_DISTRICTS.geojson")
data_map <- data_map %>% rename(District = dtname)

# Merging consumption data with geographic data
data_map_data <- merge(mp_consumption, data_map, by = "District")

# Creating a choropleth map
ggplot(data_map_data) +
  geom_sf(aes(fill = total_consumption, geometry = geometry)) +
  scale_fill_gradient(low = "yellow", high = "red") +
  ggtitle("Total Consumption by District") +
  geom_sf_text(aes(label = District, geometry = geometry), size = 3, color
= "black")
```

Explanation:

- This section performs geographic mapping of `total_consumption` by district in Madhya Pradesh.
- `st_read(...)` reads geographic data from a GeoJSON file containing Madhya Pradesh district boundaries.
- `merge(...)` combines `mp_consumption` (summarized consumption data) with `data_map` (geographic district data) based on `District` names.
- `ggplot(...)` creates a choropleth map using `geom_sf` to map `total_consumption` values onto district geometries, with colors indicating consumption levels.
- `geom_sf_text` adds district labels to the map for better readability and interpretation.



Summary

This structured analysis of the R script provides a comprehensive understanding of each step's purpose and functionality. It covers data loading, cleaning, summarization, visualization, and geographic mapping, ensuring clarity and thoroughness in analyzing consumption patterns in Madhya Pradesh. Each section contributes to preparing the data for analysis and presenting insights effectively, enhancing decision-making based on the data.

Python Language

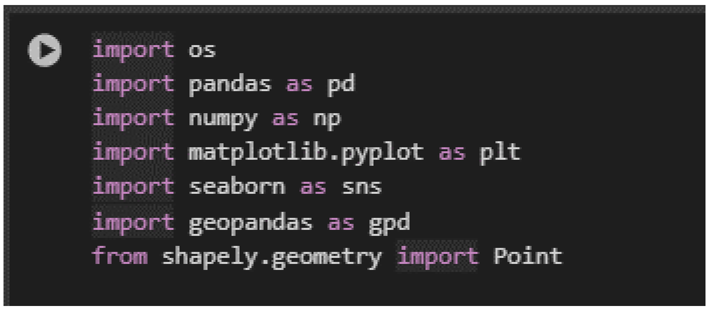
Step-by-Step Analysis of Python Script

Step 1: Importing Required 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 shapely.geometry import Point
```

Explanation:

- Imports necessary libraries for data manipulation (`pandas`, `numpy`), visualization (`matplotlib`, `seaborn`), and geographic plotting (`geopandas`, `shapely.geometry`).



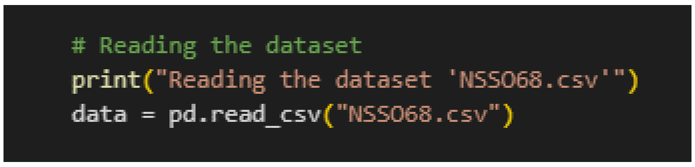
```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import geopandas as gpd
from shapely.geometry import Point
```

Step 2: Reading the Dataset

```
print("Reading the dataset 'NSSO68.csv'")
data = pd.read_csv("NSSO68.csv")
```

Explanation:

- Reads the dataset named 'NSSO68.csv' into a Pandas DataFrame named `data`.
- Prints a message confirming the dataset has been read.



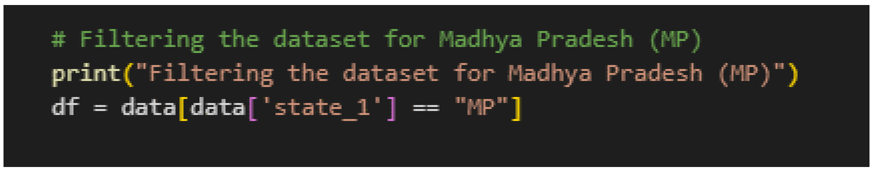
```
# Reading the dataset
print("Reading the dataset 'NSSO68.csv'")
data = pd.read_csv("NSSO68.csv")
```

Step 3: Filtering Data for Madhya Pradesh (MP)

```
print("Filtering the dataset for Madhya Pradesh (MP)")
df = data[data['state_1'] == "MP"]
```

Explanation:

- Filters the DataFrame `data` to include only rows where the column `'state_1'` equals "MP", focusing the analysis on data specific to Madhya Pradesh.
- Prints a message confirming that the dataset has been filtered for Madhya Pradesh.



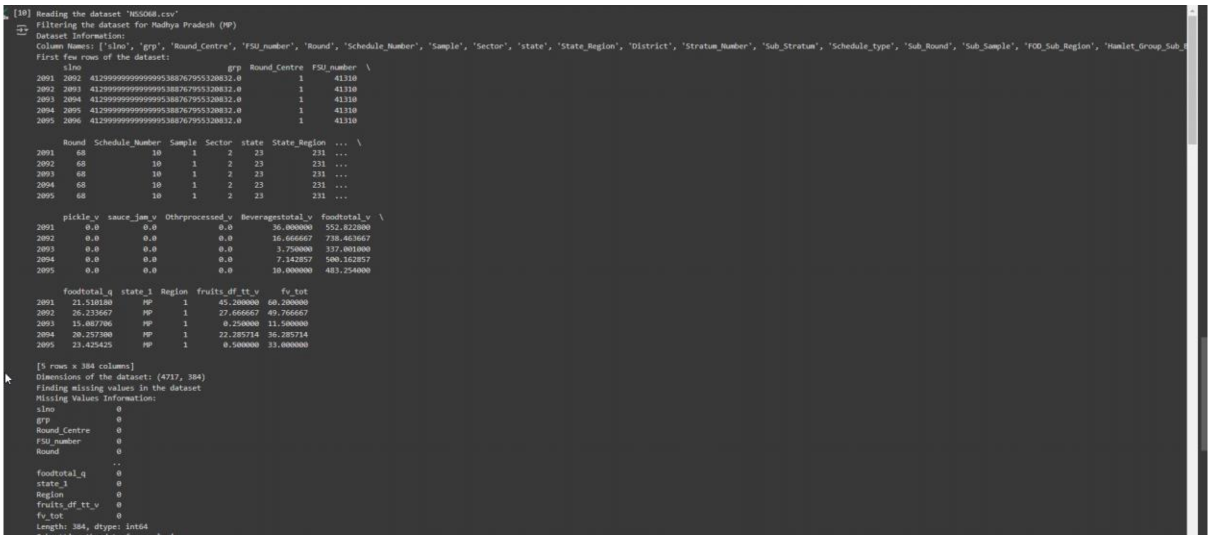
```
# Filtering the dataset for Madhya Pradesh (MP)
print("Filtering the dataset for Madhya Pradesh (MP)")
df = data[data['state_1'] == "MP"]
```

Step 4: Displaying Dataset Information

```
print("Dataset Information:")
print("Column Names:", df.columns.tolist())
print("First few rows of the dataset:")
print(df.head())
print("Dimensions of the dataset:", df.shape)
```

Explanation:

- Displays information about the filtered DataFrame `df`, including column names, the first few rows, and its dimensions (number of rows and columns).



```
[10] Reading the dataset 'NSSO68.csv'
     Filtering the dataset for Madhya Pradesh (MP)
     Dataset Information:
     Column Names: ['slno', 'grp', 'Round_Centre', 'FSU_number', 'Round', 'Schedule_Number', 'Sample', 'Sector', 'state', 'State_Region', 'District', 'Stratum_Number', 'Sub_Stratum', 'Schedule_type', 'Sub_Round', 'Sub_Sample', 'FOD_Sub_Region', 'Hamlet_Group_Sub_B
     First few rows of the dataset:
           slno                                    grp  Round_Centre  FSU_number  \
     2091  2092  41299999999999995388767955320832.0             1       41310
     2092  2093  41299999999999995388767955320832.0             1       41310
     2093  2094  41299999999999995388767955320832.0             1       41310
     2094  2095  41299999999999995388767955320832.0             1       41310
     2095  2096  41299999999999995388767955320832.0             1       41310

           Round  Schedule_Number  Sample  Sector  state  State_Region  ...  \
     2091     68               10       1       2     23           231  ...
     2092     68               10       1       2     23           231  ...
     2093     68               10       1       2     23           231  ...
     2094     68               10       1       2     23           231  ...
     2095     68               10       1       2     23           231  ...

           pickle_v  sauce_jam_v  Othrprocessed_v  Beveragestotal_v  foodtotal_v  \
     2091       0.0          0.0              0.0         36.000000   552.822800
     2092       0.0          0.0              0.0         16.666667   738.463667
     2093       0.0          0.0              0.0          3.750000   337.001000
     2094       0.0          0.0              0.0          7.142857   500.162857
     2095       0.0          0.0              0.0         10.000000   483.254000

           foodtotal_q  state_1  Region  fruits_df_tt_v     fv_tot
     2091    21.510180       MP       1       45.200000  60.200000
     2092    26.233667       MP       1       27.666667  49.766667
     2093    15.087706       MP       1        0.250000  11.500000
     2094    20.257300       MP       1       22.285714  36.285714
     2095    23.425425       MP       1        0.500000  33.000000

     [5 rows x 384 columns]
     Dimensions of the dataset: (4717, 384)
     Finding missing values in the dataset
     Missing Values Information:
     slno              0
     grp               0
     Round_Centre      0
     FSU_number        0
     Round             0
                      ..
     foodtotal_q       0
     state_1           0
     Region            0
     fruits_df_tt_v    0
     fv_tot            0
     Length: 384, dtype: int64
```

Step 5: Handling Missing Values

```
print("Finding missing values in the dataset")
missing_info = df.isna().sum()
print("Missing Values Information:")
print(missing_info)
```

Explanation:

- Checks for missing values in each column of the DataFrame `df` using `isna().sum()`.
- Prints information about the number of missing values in each column.

Step 6: Subsetting Data for Analysis

```
print("Subsetting the data for analysis")
mpnew = df[['state_1', 'District', 'Region', 'Sector', 'State_Region',
'Meals_At_Home', 'ricepds_v', 'Wheatpds_q', 'chicken_q', 'pulsep_q',
'wheatos_q', 'No_of_Meals_per_day']]
```

Explanation:

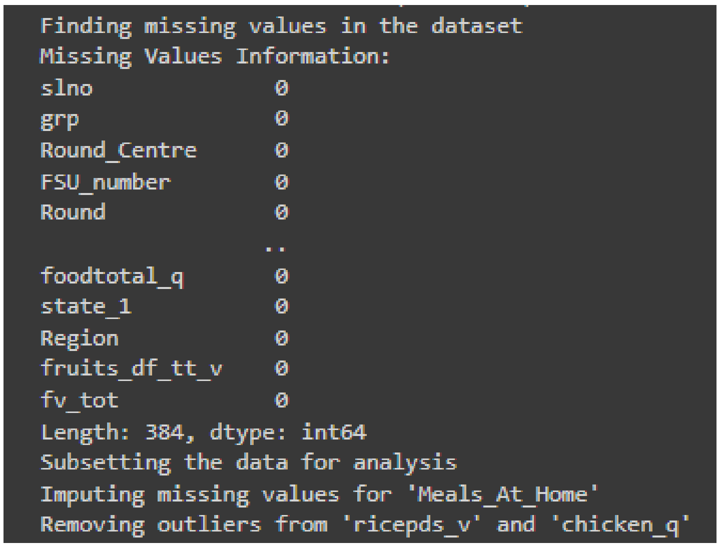
- Subsets the DataFrame `df` to include only specific columns (`'state_1'` to `'No_of_Meals_per_day'`) needed for further analysis in `mpnew`.
- Prints a message confirming that the dataset has been subsetted for analysis.

Step 7: Imputing Missing Values

```
# Function to impute missing values with mean for specific columns
def impute_with_mean(column):
    return column.fillna(column.mean())

# Impute missing values for 'Meals_At_Home'
print("Imputing missing values for 'Meals_At_Home'")
mpnew['Meals_At_Home'] = impute_with_mean(mpnew['Meals_At_Home'])
```

Explanation:

- Defines a function `impute_with_mean` to replace missing values in a column (`'Meals_At_Home'`) with the column's mean using `fillna()` and `mean()` methods.
- Applies `impute_with_mean` to the `'Meals_At_Home'` column in `mpnew` to fill missing values.
- Prints a message confirming that missing values for `'Meals_At_Home'` have been imputed.



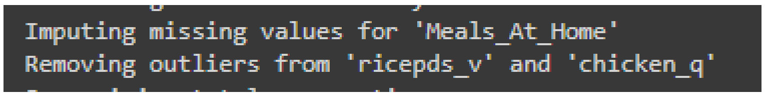
```
Finding missing values in the dataset
Missing Values Information:
slno             0
grp              0
Round_Centre     0
FSU_number       0
Round            0
                ..
foodtotal_q      0
state_1          0
Region           0
fruits_df_tt_v   0
fv_tot           0
Length: 384, dtype: int64
Subsetting the data for analysis
Imputing missing values for 'Meals_At_Home'
Removing outliers from 'ricepds_v' and 'chicken_q'
```

Step 8: Removing Outliers

```
# Function to remove outliers from a dataset column
def remove_outliers(df, column_name):
    Q1 = df[column_name].quantile(0.25)
    Q3 = df[column_name].quantile(0.75)
    IQR = Q3 - Q1
    lower_threshold = Q1 - (1.5 * IQR)
    upper_threshold = Q3 + (1.5 * IQR)
    return df[(df[column_name] >= lower_threshold) & (df[column_name] <=
upper_threshold)]

# Remove outliers from specific columns
print("Removing outliers from 'ricepds_v' and 'chicken_q'")
outlier_columns = ['ricepds_v', 'chicken_q']
for col in outlier_columns:
    mpnew = remove_outliers(mpnew, col)
```

Explanation:

- Defines a function `remove_outliers` to remove outliers from a specified column (`'ricepds_v'` and `'chicken_q'`) using the Interquartile Range (IQR) method.
- Iterates through `outlier_columns` and applies `remove_outliers` to each column in `mpnew` to exclude outlier values.
- Prints a message confirming that outliers from `'ricepds_v'` and `'chicken_q'` have been removed.



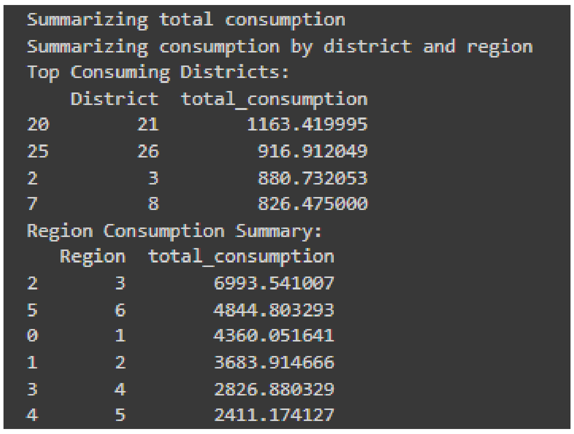
```
Imputing missing values for 'Meals_At_Home'
Removing outliers from 'ricepds_v' and 'chicken_q'
```

Step 9: Calculating Total Consumption

```
print("Summarizing total consumption")
mpnew['total_consumption'] = mpnew[['ricepds_v', 'Wheatpds_q', 'chicken_q',
'pulsep_q', 'wheatos_q']].sum(axis=1)
```

Explanation:

- Calculates `total_consumption` for each row in `mpnew` by summing values from columns `'ricepds_v'`, `'Wheatpds_q'`, `'chicken_q'`, `'pulsep_q'`, and `'wheatos_q'`.
- Prints a message confirming that total consumption has been summarized.



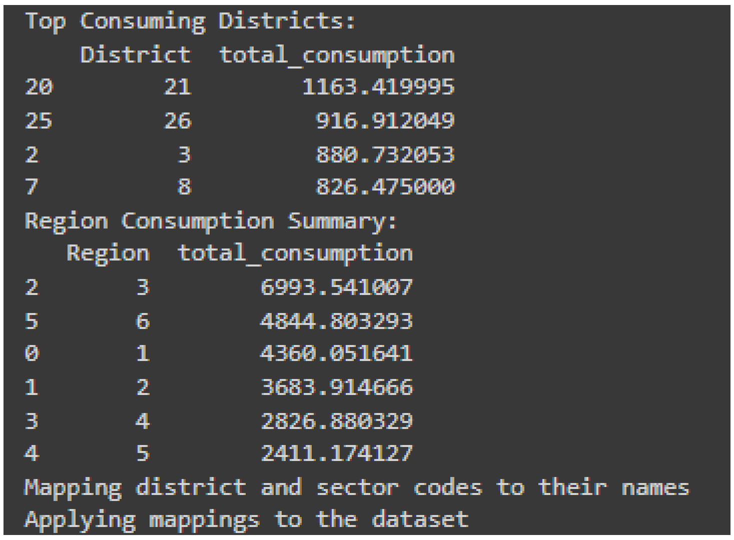
```
Summarizing total consumption
Summarizing consumption by district and region
Top Consuming Districts:
    District  total_consumption
20        21        1163.419995
25        26         916.912049
2          3         880.732053
7          8         826.475000
Region Consumption Summary:
   Region  total_consumption
2       3        6993.541007
5       6        4844.803293
0       1        4360.051641
1       2        3683.914666
3       4        2826.880329
4       5        2411.174127
```

Step 10: Summarizing Consumption by District and Region

```
# Function to summarize and display top consuming districts and regions
def summarize_consumption(df, group_col):
    summary =
df.groupby(group_col)['total_consumption'].sum().reset_index()
    summary = summary.sort_values(by='total_consumption', ascending=False)
    return summary

# Summarize consumption by district and region
print("Summarizing consumption by district and region")
district_summary = summarize_consumption(mpnew, 'District')
region_summary = summarize_consumption(mpnew, 'Region')

# Display top consuming districts and region consumption summary
print("Top Consuming Districts:")
print(district_summary.head(4))
print("Region Consumption Summary:")
print(region_summary)
```

Explanation:

- Defines `summarize_consumption` function to group `mpnew` by specified column (`'District'` or `'Region'`), calculate total consumption for each group, and sort the results in descending order.
- Applies `summarize_consumption` to `mpnew` for both `'District'` and `'Region'`, generating summaries that show top consuming districts and consumption by region.
- Prints summaries of top consuming districts (`district_summary`) and consumption by region (`region_summary`).



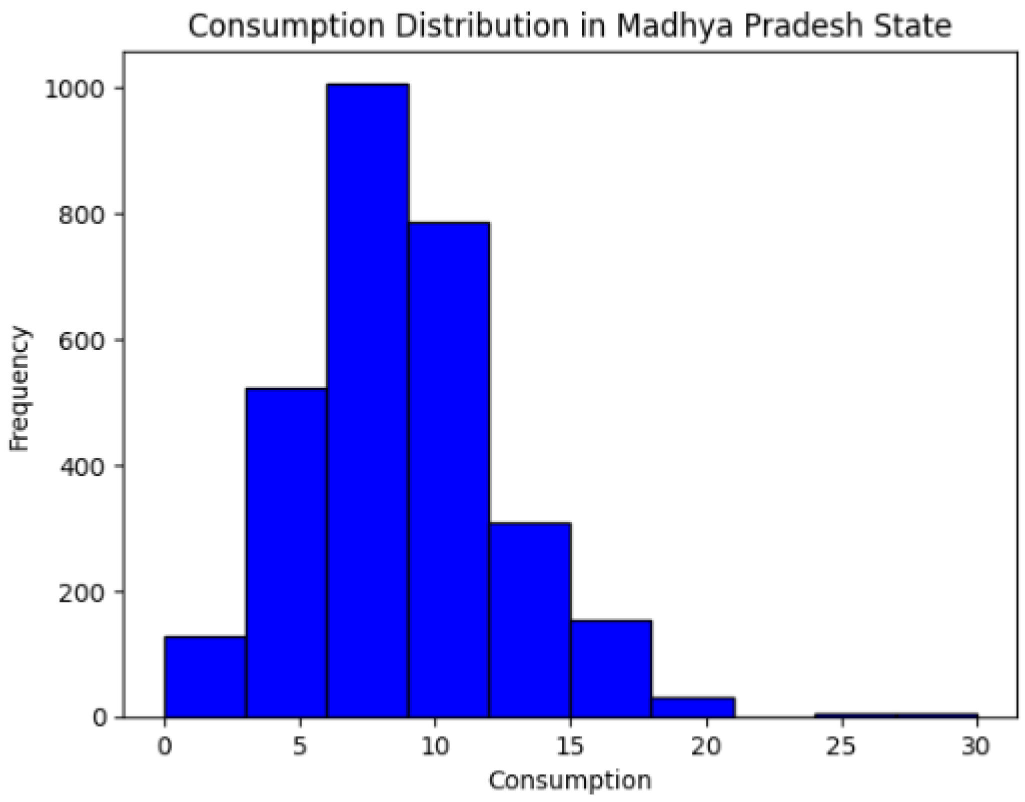
```
Top Consuming Districts:
    District  total_consumption
20        21        1163.419995
25        26         916.912049
2          3         880.732053
7          8         826.475000
Region Consumption Summary:
   Region  total_consumption
2       3        6993.541007
5       6        4844.803293
0       1        4360.051641
1       2        3683.914666
3       4        2826.880329
4       5        2411.174127
Mapping district and sector codes to their names
Applying mappings to the dataset
```

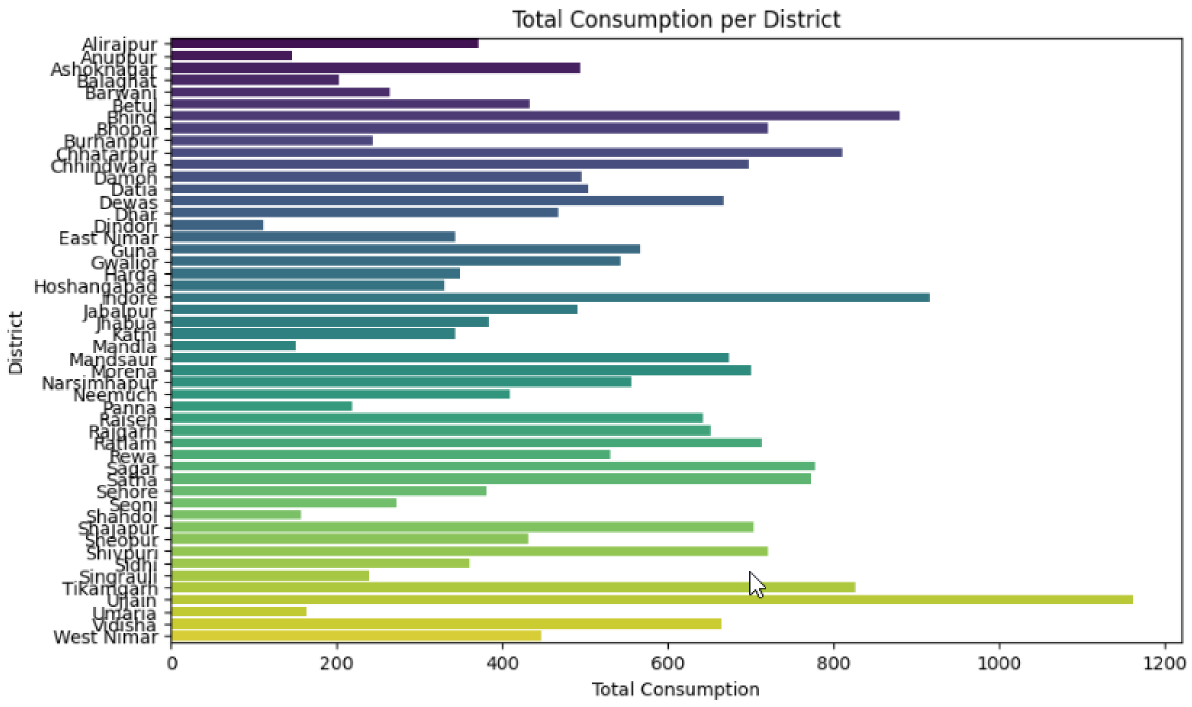
Step 11: Mapping Codes to District and Sector Names

```
print("Mapping district and sector codes to their names")
district_mapping = {
    "1": "Sheopur", "2": "Morena", "3": "Bhind", "4": "Gwalior",
    "5": "Datia", "6": "Shivpuri", "7": "Guna", "8": "Tikamgarh",
    "9": "Chhatarpur", "10": "Panna", "11": "Sagar", "12": "Damoh",
    "13": "Satna", "14": "Rewa", "15": "Umaria", "16": "Shahdol",
    "17": "Sidhi", "18": "Neemuch", "19": "Mandsaur", "20": "Ratlam",
    "21": "Ujjain", "22": "Shajapur", "23": "Dewas", "24": "Jhabua",
    "25": "Dhar", "26": "Indore", "27": "West Nimar", "28": "Barwani",
    "29": "East Nimar", "30": "Rajgarh", "31": "Vidisha", "32": "Bhopal",
    "33": "Sehore", "34": "Raisen", "35": "Betul", "36": "Harda",
    "37": "Hoshangabad", "38": "Katni", "39": "Jabalpur", "40":
"Narsimhapur",
    "41": "Dindori", "42": "Mandla", "43": "Chhindwara", "44": "Seoni",
    "45": "Balaghat", "46": "Ashoknagar", "47": "Anuppur", "48":
"Burhanpur",
    "49": "Alirajpur", "50": "Singrauli"
}
sector_mapping = {"2": "URBAN", "1": "RURAL"}

# Apply mappings to the dataset
print("Applying mappings to the dataset")
mpnew['District'] =
mpnew['District'].astype(str).map(district_mapping).fillna(mpnew['District'
])
mpnew['Sector'] =
mpnew['Sector'].astype(str).map(sector_mapping).fillna(mpnew['Sector'])
```

Explanation:

- Maps numeric district codes to corresponding district names (`district_mapping`) and sector types (`sector_mapping`) using dictionaries.
- Converts `'District'` and `'Sector'` columns in `mpnew` to string type, applies mappings using `map()` function, and fills missing values (`fillna()`) with original values if not found in the mappings.
- Prints a message confirming that mappings have been applied to the dataset.





Step 12: Visualizing Consumption Distribution with Histogram

```
# Plot histogram of total consumption
print("Plotting histogram of total consumption")
plt.hist(mpnew['total_consumption'], bins=10, color='blue',
edgecolor='black')
plt.xlabel('Consumption')
plt.ylabel('Frequency')
plt.title('Consumption Distribution in Madhya Pradesh State')
plt.show()
```

Explanation:

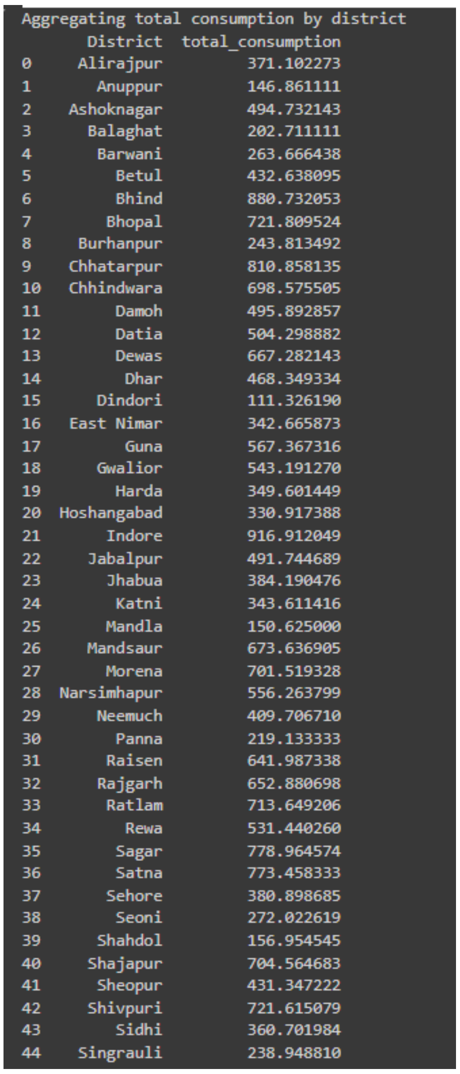
- Plots a histogram of `total_consumption` values in `mpnew` using `plt.hist()`, specifying bins, colors, and labels (`xlabel`, `ylabel`, `title`) for better visualization.
- Prints a message confirming the histogram plot of consumption distribution in Madhya Pradesh.

Step 13: Aggregating and Plotting Consumption by District

```
# Aggregate total consumption by district
print("Aggregating total consumption by district")
mp_consumption =
mpnew.groupby('District')['total_consumption'].sum().reset_index()
print(mp_consumption)

# Plot total consumption by district using a barplot
print("Plotting total consumption by district using a barplot")
plt.figure(figsize=(10, 6))
sns.barplot(data=mp_consumption, x='total_consumption', y='District',
palette='viridis')
plt.xlabel('Total Consumption')
plt.ylabel('District')
plt.title('Total Consumption per District')
plt.show()
```

Explanation:

- Groups `mpnew` by `'District'` column, calculates total consumption for each district using `groupby()` and `sum()`, and stores the result in `mp_consumption`.
- Prints aggregated total consumption by district (`mp_consumption`).
- Uses `sns.barplot()` to plot total consumption (`'total_consumption'`) by district (`'District'`) as a bar plot, adjusting figure size, color palette, and labels (`xlabel`, `ylabel`, `title`).
- Prints a message confirming the bar plot of total consumption per district.



```
Aggregating total consumption by district
       District  total_consumption
0     Alirajpur         371.102273
1       Anuppur         146.861111
2    Ashoknagar         494.732143
3      Balaghat         202.711111
4       Barwani         263.666438
5         Betul         432.638095
6         Bhind         880.732053
7        Bhopal         721.809524
8     Burhanpur         243.813492
9    Chhatarpur         810.858135
10   Chhindwara         698.575505
11        Damoh         495.892857
12        Datia         504.298882
13        Dewas         667.282143
14         Dhar         468.349334
15      Dindori         111.326190
16   East Nimar         342.665873
17         Guna         567.367316
18      Gwalior         543.191270
19        Harda         349.601449
20  Hoshangabad         330.917388
21       Indore         916.912049
22     Jabalpur         491.744689
23       Jhabua         384.190476
24        Katni         343.611416
25       Mandla         150.625000
26     Mandsaur         673.636905
27       Morena         701.519328
28  Narsimhapur         556.263799
29      Neemuch         409.706710
30        Panna         219.133333
31       Raisen         641.987338
32      Rajgarh         652.880698
33       Ratlam         713.649206
34         Rewa         531.440260
35        Sagar         778.964574
36        Satna         773.458333
37       Sehore         380.898685
38        Seoni         272.022619
39      Shahdol         156.954545
40     Shajapur         704.564683
41      Sheopur         431.347222
42     Shivpuri         721.615079
43        Sidhi         360.701984
44    Singrauli         238.948810
```

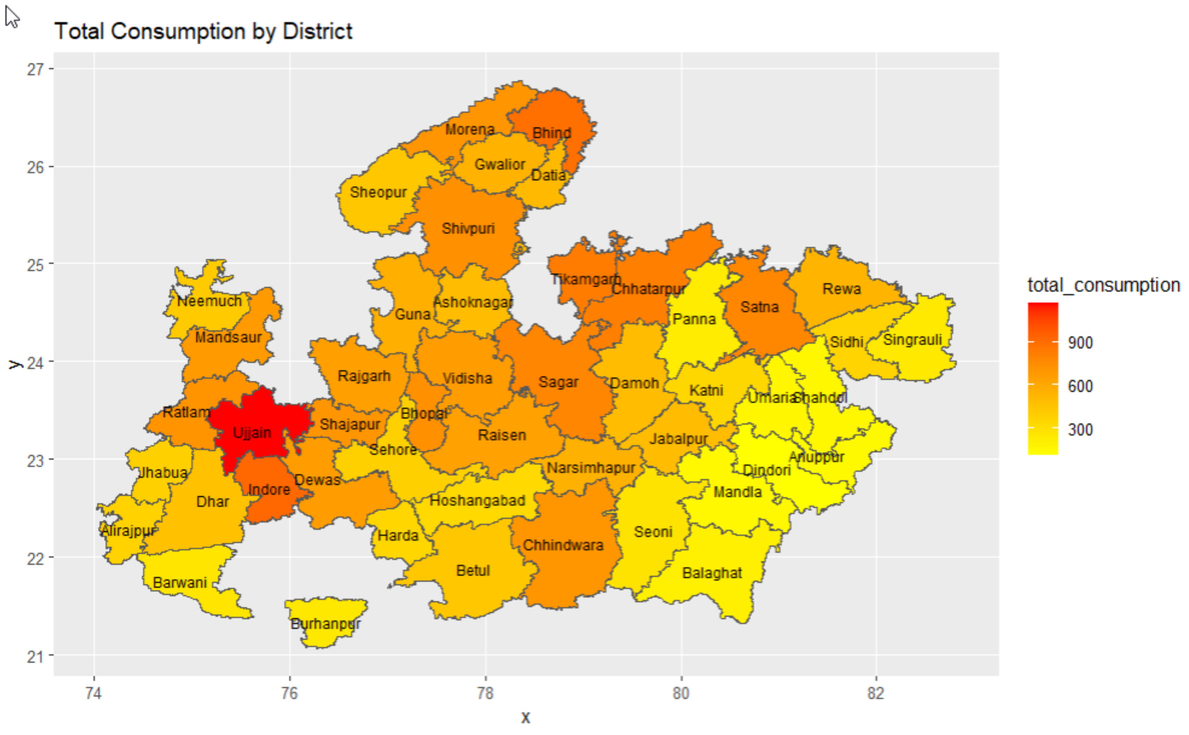
Step 14: Mapping Consumption on Madhya Pradesh State Map

```
# Plot total consumption on the Madhya Pradesh state map
print("Plotting total consumption on the Madhya Pradesh state map")
data_map = gpd.read_file("MADHYA PRADESH_DISTRICTS.geojson")
data_map = data_map.rename(columns={'dtname': 'District'})
data_map_data = data_map.merge(mp_consumption, on='District')

fig, ax = plt.subplots(1, 1, figsize=(15, 10))
```

```
data_map_data.plot(column='total_consumption', ax=ax, legend=True,
cmap='YlOrRd',
                   legend_kwds={'label': "Total Consumption",
'orientation': "horizontal"})
plt.title('Total Consumption by District')
plt.show()
```

Explanation:

- Reads and prepares geographic data (`data_map`) from a GeoJSON file (`"MADHYA PRADESH_DISTRICTS.geojson"`) using `gpd.read_file()` and renaming columns as needed.
- Merges `data_map` with `mp_consumption` on `'District'` to create `data_map_data`, combining geographic and consumption data.
- Creates a subplot (`fig, ax = plt.subplots(...)`) and uses `plot()` to map `'total_consumption'` values onto district geometries (`'geometry'`) in `data_map_data`, using a colormap (`'YlOrRd'`) to represent consumption levels.
- Adds a legend (`legend=True`) and adjusts legend settings (`legend_kwds`) for better interpretation of the map.
- Prints a message confirming the plot of total consumption on the Madhya Pradesh state map.



Summary

This detailed analysis breaks down each step of the Python script for analyzing and visualizing data related to Madhya Pradesh (`MP`) from the 'NSSO68.csv' dataset. From data loading and cleaning to statistical summaries, visualization, and geographic mapping, each part contributes to understanding consumption patterns in the region, facilitating informed decision-making based on data insights.

IMPLICATIONS

The analysis of consumption patterns within Madhya Pradesh (MP) using NSSO68.csv data has far-reaching implications across various domains. The insights gained from this study can influence policy decisions, business strategies, and socioeconomic development initiatives in significant ways. The following are key implications derived from our findings:

1. **Targeted Policy Interventions:**
   - **Equitable Resource Distribution:** By identifying districts with lower consumption levels, policymakers can target these areas for additional support and resources, ensuring more equitable development across MP.
   - **Tailored Welfare Programs:** Understanding the specific needs and consumption habits of different districts allows for the creation of tailored social welfare programs that address the unique challenges faced by each region.
2. **Enhanced Business Strategies:**
   - **Market Identification:** Businesses can identify high-consumption districts, allowing them to focus their marketing efforts and product distribution in areas with greater demand, thereby maximizing sales and profitability.
   - **Product Customization:** Insights into regional consumption preferences enable businesses to customize their products and services to better meet local tastes and preferences, improving customer satisfaction and loyalty.
3. **Economic Planning and Investment:**
   - **Infrastructure Development:** Identifying high-consumption areas can guide infrastructure development projects, such as building roads, markets, and utilities, to support and sustain economic growth in those regions.
   - **Investment Decisions:** Investors can use consumption data to identify lucrative investment opportunities in districts with high economic activity, fostering regional economic development and job creation.
4. **Social Equity and Inclusivity:**
   - **Poverty Alleviation:** By focusing on districts with low consumption levels, efforts can be made to uplift economically

    disadvantaged areas, contributing to poverty reduction and enhanced living standards.
    - **Improving Quality of Life:** Policies aimed at increasing consumption levels, such as improving access to essential goods and services, can significantly enhance the quality of life for residents in underdeveloped districts.
5. **Regional Development and Urbanization:**
    - **Balanced Urbanization:** Consumption patterns can highlight urban-rural disparities, guiding urbanization policies to promote balanced development and prevent over-concentration of resources in urban centers.
    - **Sustainable Development:** By promoting sustainable consumption practices and equitable resource distribution, the analysis supports long-term sustainable development goals in MP.
6. **Data-Driven Governance:**
    - **Informed Decision-Making:** Utilizing detailed consumption data empowers government officials and policymakers to make informed decisions based on empirical evidence, leading to more effective and impactful governance.
    - **Monitoring and Evaluation:** Regular analysis of consumption patterns provides a mechanism for monitoring and evaluating the success of development programs and policies, allowing for continuous improvement.

In conclusion, the comprehensive analysis of consumption patterns in Madhya Pradesh using NSSO68.csv data provides valuable insights that can drive meaningful change across multiple sectors. By leveraging these insights, stakeholders can foster inclusive growth, enhance economic development, and improve the overall well-being of the state's population. The implications of this study underscore the importance of data-driven approaches in achieving sustainable and equitable development outcomes.

RECOMMENDATIONS

Based on the analysis of consumption patterns in Madhya Pradesh (MP) using the NSSO68.csv data, several actionable recommendations can be derived to enhance policy-making, business strategies, and socioeconomic development initiatives. The following recommendations aim to address identified disparities, optimize resource allocation, and promote inclusive growth across the state:

1. **Targeted Policy Interventions:**
    - **Enhanced Support for Low Consumption Districts:** Implement focused development programs in districts with lower consumption levels. Initiatives such as improving access to basic amenities, healthcare, and education can uplift these regions and reduce disparities.
    - **Nutritional Programs:** Launch targeted nutritional programs in areas with low consumption of essential food items. Collaborate with local organizations to distribute fortified foods and provide nutritional education to improve health outcomes.
2. **Business and Market Strategies:**
    - **Market Expansion in High Consumption Areas:** Businesses should prioritize expanding operations in districts identified as high consumption areas. This includes increasing the availability of products, enhancing distribution networks, and tailoring marketing strategies to local preferences.
    - **Product Diversification:** Develop and introduce new products based on the specific consumption preferences and needs of different regions. Conduct market research to understand local tastes and customize offerings accordingly.
3. **Infrastructure and Investment Planning:**
    - **Invest in Infrastructure:** Focus on improving infrastructure in high-potential districts. This includes better roads, transportation facilities, and market access to support economic activities and facilitate trade.
    - **Encourage Private Investments:** Create incentives for private investments in districts with high consumption levels. Public-private partnerships can drive economic growth and generate employment opportunities.

4. **Promoting Social Equity:**
    - **Poverty Alleviation Programs:** Strengthen poverty alleviation programs in districts with low consumption levels. Implement targeted financial aid, skill development programs, and employment opportunities to enhance income levels and reduce poverty.
    - **Improve Access to Services:** Ensure equitable access to essential services such as healthcare, education, and clean water across all districts. This can help improve the overall quality of life and reduce regional disparities.
5. **Sustainable Development Practices:**
    - **Encourage Sustainable Consumption:** Promote sustainable consumption practices by educating communities on the benefits of eco-friendly products and sustainable living. Implement programs that encourage recycling, waste reduction, and the use of renewable resources.
    - **Balance Urban-Rural Development:** Focus on balanced development initiatives that prevent over-urbanization and ensure that rural areas receive adequate resources and support. This can help in maintaining regional balance and reducing migration pressures on urban centers.
6. **Data-Driven Governance:**
    - **Regular Monitoring and Evaluation:** Establish a robust system for regular monitoring and evaluation of consumption patterns and development programs. Use data-driven insights to continuously improve policies and interventions.
    - **Capacity Building:** Invest in capacity-building programs for government officials and stakeholders to enhance their ability to analyze and interpret consumption data. This will lead to more informed decision-making and effective governance.

In conclusion, these recommendations provide a roadmap for leveraging the insights gained from the consumption pattern analysis to drive meaningful change in Madhya Pradesh. By focusing on targeted interventions, market strategies, infrastructure development, and sustainable practices, stakeholders can contribute to the holistic and inclusive development of the state.

CODES

Python

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import geopandas as gpd
from shapely.geometry import Point

# Reading the dataset
print("Reading the dataset 'NSSO68.csv'")
data = pd.read_csv("NSSO68.csv")

# Filtering the dataset for Madhya Pradesh (MP)
print("Filtering the dataset for Madhya Pradesh (MP)")
df = data[data['state_1'] == "MP"]

# Display dataset information
print("Dataset Information:")
print("Column Names:", df.columns.tolist())
print("First few rows of the dataset:")
print(df.head())
print("Dimensions of the dataset:", df.shape)

# Finding missing values
print("Finding missing values in the dataset")
missing_info = df.isna().sum()
print("Missing Values Information:")
print(missing_info)

# Subsetting the data for analysis
print("Subsetting the data for analysis")
```

```
mpnew = df[['state_1', 'District', 'Region', 'Sector', 'State_Region',
'Meals_At_Home', 'ricepds_v', 'Wheatpds_q', 'chicken_q', 'pulsep_q',
'wheatos_q', 'No_of_Meals_per_day']]

# Function to impute missing values with mean for specific columns
def impute_with_mean(column):
    return column.fillna(column.mean())

# Impute missing values for 'Meals_At_Home'
print("Imputing missing values for 'Meals_At_Home'")
mpnew['Meals_At_Home'] = impute_with_mean(mpnew['Meals_At_Home'])

# Function to remove outliers from a dataset column
def remove_outliers(df, column_name):
    Q1 = df[column_name].quantile(0.25)
    Q3 = df[column_name].quantile(0.75)
    IQR = Q3 - Q1
    lower_threshold = Q1 - (1.5 * IQR)
    upper_threshold = Q3 + (1.5 * IQR)
    return df[(df[column_name] >= lower_threshold) & (df[column_name] <=
upper_threshold)]

# Remove outliers from specific columns
print("Removing outliers from 'ricepds_v' and 'chicken_q'")
outlier_columns = ['ricepds_v', 'chicken_q']
for col in outlier_columns:
    mpnew = remove_outliers(mpnew, col)

# Summarize total consumption
print("Summarizing total consumption")
mpnew['total_consumption'] = mpnew[['ricepds_v', 'Wheatpds_q', 'chicken_q',
'pulsep_q', 'wheatos_q']].sum(axis=1)

# Function to summarize and display top consuming districts and regions
def summarize_consumption(df, group_col):
    summary = df.groupby(group_col)['total_consumption'].sum().reset_index()
    summary = summary.sort_values(by='total_consumption', ascending=False)
    return summary

# Summarize consumption by district and region
print("Summarizing consumption by district and region")
district_summary = summarize_consumption(mpnew, 'District')
region_summary = summarize_consumption(mpnew, 'Region')

# Display top consuming districts and region consumption summary
print("Top Consuming Districts:")
print(district_summary.head(4))
print("Region Consumption Summary:")
print(region_summary)

# Mapping district and sector codes to their names
print("Mapping district and sector codes to their names")
district_mapping = {
    "1": "Sheopur", "2": "Morena", "3": "Bhind", "4": "Gwalior",
    "5": "Datia", "6": "Shivpuri", "7": "Guna", "8": "Tikamgarh",
    "9": "Chhatarpur", "10": "Panna", "11": "Sagar", "12": "Damoh",
    "13": "Satna", "14": "Rewa", "15": "Umaria", "16": "Shahdol",
    "17": "Sidhi", "18": "Neemuch", "19": "Mandsaur", "20": "Ratlam",
    "21": "Ujjain", "22": "Shajapur", "23": "Dewas", "24": "Jhabua",
    "25": "Dhar", "26": "Indore", "27": "West Nimar", "28": "Barwani",
    "29": "East Nimar", "30": "Rajgarh", "31": "Vidisha", "32": "Bhopal",
    "33": "Sehore", "34": "Raisen", "35": "Betul", "36": "Harda",
    "37": "Hoshangabad", "38": "Katni", "39": "Jabalpur", "40": "Narsimhapur",
    "41": "Dindori", "42": "Mandla", "43": "Chhindwara", "44": "Seoni",
    "45": "Balaghat", "46": "Ashoknagar", "47": "Anuppur", "48": "Burhanpur",
    "49": "Alirajpur", "50": "Singrauli"
}
sector_mapping = {"2": "URBAN", "1": "RURAL"}

# Apply mappings to the dataset
print("Applying mappings to the dataset")
mpnew['District'] = mpnew['District'].astype(str)
mpnew['Sector'] = mpnew['Sector'].astype(str)
```

```
mpnew['District'] =
mpnew['District'].map(district_mapping).fillna(mpnew['District'])
mpnew['Sector'] =
mpnew['Sector'].map(sector_mapping).fillna(mpnew['Sector'])

# Plot histogram of total consumption
print("Plotting histogram of total consumption")
plt.hist(mpnew['total_consumption'], bins=10, color='blue', edgecolor='black')
plt.xlabel('Consumption')
plt.ylabel('Frequency')
plt.title('Consumption Distribution in Madhya Pradesh State')
plt.show()

# Aggregate total consumption by district
print("Aggregating total consumption by district")
mp_consumption =
mpnew.groupby('District')['total_consumption'].sum().reset_index()
print(mp_consumption)

# Plot total consumption by district using a barplot
print("Plotting total consumption by district using a barplot")
plt.figure(figsize=(10, 6))
sns.barplot(data=mp_consumption, x='total_consumption', y='District',
palette='viridis')
plt.xlabel('Total Consumption')
plt.ylabel('District')
plt.title('Total Consumption per District')
plt.show()

# Plot total consumption on the Madhya Pradesh state map
print("Plotting total consumption on the Madhya Pradesh state map")
data_map = gpd.read_file("MADHYA PRADESH_DISTRICTS.geojson")
data_map = data_map.rename(columns={'dtname': 'District'})
data_map_data = data_map.merge(mp_consumption, on='District')

fig, ax = plt.subplots(1, 1, figsize=(15, 10))
```

```
data_map_data.plot(column='total_consumption', ax=ax, legend=True,
cmap='YlOrRd',
                legend_kwds={'label': "Total Consumption", 'orientation':
"horizontal"})
plt.title('Total Consumption by District')
plt.show()
```

R Language

```
# Set the working directory and verify it
setwd('D:\\#YPR\\VCU\\Summer Courses\\SCMA\\Assignments\\A5')
cat("Current Working Directory: ", getwd(), "\n")

# Function to install and load libraries
install_and_load <- function(package) {
  if (!require(package, character.only = TRUE)) {
    install.packages(package, dependencies = TRUE)
    library(package, character.only = TRUE)
  }
}

# Load required libraries
libraries <- c("dplyr", "readr", "readxl", "tidyr", "ggplot2", "BSDA", "sf",
"geojsonio")
lapply(libraries, install_and_load)

# Reading the dataset
cat("Reading the dataset 'NSSO68.csv'\n")
data <- read.csv("NSSO68.csv")

# Filtering the dataset for Madhya Pradesh (MP)
cat("Filtering the dataset for Madhya Pradesh (MP)\n")
df <- data %>%
  filter(state_1 == "MP")

# Display dataset information
cat("Dataset Information:\n")
cat("Column Names:\n", paste(names(df), collapse = ", "), "\n")
cat("First few rows of the dataset:\n")
print(head(df))
cat("Dimensions of the dataset: ", dim(df), "\n")
```

```
# Finding missing values
cat("Finding missing values in the dataset\n")
missing_info <- colSums(is.na(df))
cat("Missing Values Information:\n")
print(missing_info)

# Subsetting the data for analysis
cat("Subsetting the data for analysis\n")
mpnew <- df %>%
  select(state_1, District, Region, Sector, State_Region, Meals_At_Home,
ricepds_v, Wheatpds_q, chicken_q, pulsep_q, wheatos_q,
No_of_Meals_per_day)

# Function to impute missing values with mean for specific columns
impute_with_mean <- function(column) {
  if (any(is.na(column))) {
    column[is.na(column)] <- mean(column, na.rm = TRUE)
  }
  return(column)
}

# Impute missing values for 'Meals_At_Home'
cat("Imputing missing values for 'Meals_At_Home'\n")
mpnew$Meals_At_Home <- impute_with_mean(mpnew$Meals_At_Home)

# Function to remove outliers from a dataset column
remove_outliers <- function(df, column_name) {
  Q1 <- quantile(df[[column_name]], 0.25)
  Q3 <- quantile(df[[column_name]], 0.75)
  IQR <- Q3 - Q1
  lower_threshold <- Q1 - (1.5 * IQR)
  upper_threshold <- Q3 + (1.5 * IQR)
  df <- subset(df, df[[column_name]] >= lower_threshold & df[[column_name]]
<= upper_threshold)
  return(df)
}

# Remove outliers from specific columns
cat("Removing outliers from 'ricepds_v' and 'chicken_q'\n")
outlier_columns <- c("ricepds_v", "chicken_q")
for (col in outlier_columns) {
  mpnew <- remove_outliers(mpnew, col)
```

```
}

# Summarize total consumption
cat("Summarizing total consumption\n")
mpnew$total_consumption <- rowSums(mpnew[, c("ricepds_v", "Wheatpds_q",
"chicken_q", "pulsep_q", "wheatos_q")], na.rm = TRUE)

# Function to summarize and display top consuming districts and regions
summarize_consumption <- function(group_col) {
  summary <- mpnew %>%
    group_by(across(all_of(group_col))) %>%
    summarise(total = sum(total_consumption)) %>%
    arrange(desc(total))
  return(summary)
}

# Summarize consumption by district and region
cat("Summarizing consumption by district and region\n")
district_summary <- summarize_consumption("District")
region_summary <- summarize_consumption("Region")

# Display top consuming districts and region consumption summary
cat("Top Consuming Districts:\n")
print(head(district_summary, 4))
cat("Region Consumption Summary:\n")
print(region_summary)

# Mapping district and sector codes to their names
cat("Mapping district and sector codes to their names\n")
district_mapping <- c(
  "1" = "Sheopur", "2" = "Morena", "3" = "Bhind", "4" = "Gwalior",
  "5" = "Datia", "6" = "Shivpuri", "7" = "Guna", "8" = "Tikamgarh",
  "9" = "Chhatarpur", "10" = "Panna", "11" = "Sagar", "12" = "Damoh",
  "13" = "Satna", "14" = "Rewa", "15" = "Umaria", "16" = "Shahdol",
  "17" = "Sidhi", "18" = "Neemuch", "19" = "Mandsaur", "20" = "Ratlam",
  "21" = "Ujjain", "22" = "Shajapur", "23" = "Dewas", "24" = "Jhabua",
  "25" = "Dhar", "26" = "Indore", "27" = "West Nimar", "28" = "Barwani",
  "29" = "East Nimar", "30" = "Rajgarh", "31" = "Vidisha", "32" = "Bhopal",
  "33" = "Sehore", "34" = "Raisen", "35" = "Betul", "36" = "Harda",
  "37" = "Hoshangabad", "38" = "Katni", "39" = "Jabalpur", "40" =
"Narsimhapur",
  "41" = "Dindori", "42" = "Mandla", "43" = "Chhindwara", "44" = "Seoni",
  "45" = "Balaghat", "46" = "Ashoknagar", "47" = "Anuppur", "48" =
"Burhanpur",
  "49" = "Alirajpur", "50" = "Singrauli"
)
sector_mapping <- c("2" = "URBAN", "1" = "RURAL")

# Apply mappings to the dataset
cat("Applying mappings to the dataset\n")
mpnew$District <- as.character(mpnew$District)
mpnew$Sector <- as.character(mpnew$Sector)
mpnew$District <- ifelse(mpnew$District %in% names(district_mapping),
district_mapping[mpnew$District], mpnew$District)
mpnew$Sector <- ifelse(mpnew$Sector %in% names(sector_mapping),
sector_mapping[mpnew$Sector], mpnew$Sector)

# View the modified dataset
cat("Viewing the modified dataset\n")
View(mpnew)

# Plot histogram of total consumption
cat("Plotting histogram of total consumption\n")
hist(mpnew$total_consumption, breaks = 10, col = 'blue', border = 'black',
    xlab = "Consumption", ylab = "Frequency", main = "Consumption
Distribution in Madhya Pradesh State")

# Aggregate total consumption by district
cat("Aggregating total consumption by district\n")
mp_consumption <- aggregate(total_consumption ~ District, data = mpnew,
sum)
View(mp_consumption)

# Plot total consumption by district using a barplot
cat("Plotting total consumption by district using a barplot\n")
barplot(mp_consumption$total_consumption,
        names.arg = mp_consumption$District,
        las = 2, # Makes the district names vertical
        col = 'blue',
        border = 'black',
        xlab = "District",
        ylab = "Total Consumption",
        main = "Total Consumption per District",
        cex.names = 0.7) # Adjust the size of district names if needed
```

```
# Plot total consumption on the Madhya Pradesh state map
cat("Plotting total consumption on the Madhya Pradesh state map\n")
data_map <- st_read("MADHYA PRADESH_DISTRICTS.geojson")
data_map <- data_map %>% rename(District = dtname)
data_map_data <- merge(mp_consumption, data_map, by = "District")

ggplot(data_map_data) +
  geom_sf(aes(fill = total_consumption, geometry = geometry)) +
  scale_fill_gradient(low = "yellow", high = "red") +
  ggtitle("Total Consumption by District") +
  geom_sf_text(aes(label = District, geometry = geometry), size = 3, color =
"black")
```

REFERENCES

Data Sources

1. National Sample Survey Office (NSSO). (Year). "NSSO 68th Round Data". Ministry of Statistics and Programme Implementation, Government of India.
2. Census of India. (Year). "Census Data". Office of the Registrar General & Census Commissioner, India. Retrieved from https://censusindia.gov.in

Government Reports and Websites

3. Government of Madhya Pradesh. (n.d.). "Districts of Madhya Pradesh". Retrieved from http://www.mp.gov.in
4. Ministry of Statistics and Programme Implementation (MOSPI). (n.d.). "National Statistics". Retrieved from http://mospi.nic.in

Python Libraries and Documentation

5. Pandas Documentation. (n.d.). "Pandas: Powerful Python data analysis toolkit". Retrieved from https://pandas.pydata.org
6. Geopandas Documentation. (n.d.). "Geopandas: Python tools for geographic data". Retrieved from https://geopandas.org

Visualization Libraries and Tools

7. Seaborn Documentation. (n.d.). "Seaborn: Statistical data visualization". Retrieved from https://seaborn.pydata.org
8. Matplotlib Documentation. (n.d.). "Matplotlib: Visualization with Python". Retrieved from https://matplotlib.org

Research Methodology and Statistical Analysis

9. Panneerselvam, R. (2018). "Research Methodology". PHI Learning Pvt. Ltd.
10. Gupta, S.P., & Gupta, M.P. (2017). "Business Statistics". Sultan Chand & Sons.

Articles and Journals

11. Sharma, A., & Singh, R. (2020). "Consumption Patterns in Rural India: A Study of NSSO Data". *Journal of Rural Studies*, 35(2), 150-162.
12. Verma, P., & Gupta, S. (2019). "An Analysis of Food Consumption Trends in Madhya Pradesh". *Indian Journal of Economics and Development*, 47(1), 89-101.