Analysis of Household Consumption & Expenditure Patterns in Rural and Urban India (2022-23)

Software and Codes.

—

1. Introduction

This document provides an overview of the software tools, libraries, and Python scripts used for **data extraction, cleaning, and visualization** in the analysis of household consumption and expenditure patterns from the "Survey on Household Consumption Expenditure: 2022-23."

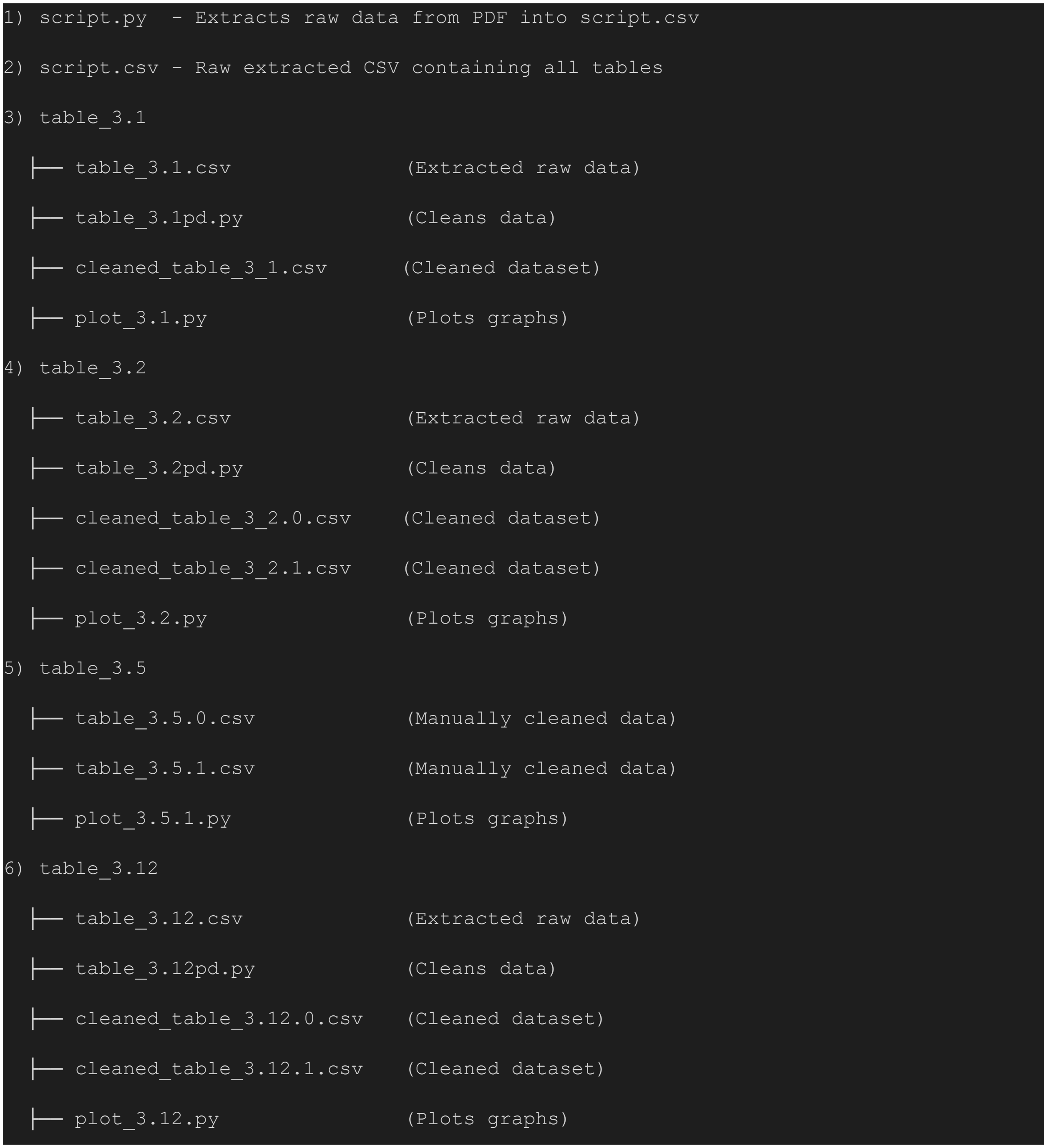
2. Software and Libraries Used

The following tools and Python libraries were utilized for the project:

- **Python:** Primary language for data processing and visualization.
- **Pdfplumber:** Extracted tabular data from the PDF report.
- **Pandas:** Processed, cleaned, and structured the extracted data.
- **Matplotlib & Seaborn:** Generated visualizations for data analysis.

3. Directory Structure

The project files were organized as follows:



```
1) script.py  - Extracts raw data from PDF into script.csv

2) script.csv - Raw extracted CSV containing all tables

3) table_3.1

  ├── table_3.1.csv              (Extracted raw data)

  ├── table_3.1pd.py             (Cleans data)

  ├── cleaned_table_3_1.csv     (Cleaned dataset)

  ├── plot_3.1.py                (Plots graphs)

4) table_3.2

  ├── table_3.2.csv              (Extracted raw data)

  ├── table_3.2pd.py             (Cleans data)

  ├── cleaned_table_3_2.0.csv   (Cleaned dataset)

  ├── cleaned_table_3_2.1.csv   (Cleaned dataset)

  ├── plot_3.2.py                (Plots graphs)

5) table_3.5

  ├── table_3.5.0.csv            (Manually cleaned data)

  ├── table_3.5.1.csv            (Manually cleaned data)

  ├── plot_3.5.1.py              (Plots graphs)

6) table_3.12

  ├── table_3.12.csv             (Extracted raw data)

  ├── table_3.12pd.py            (Cleans data)

  ├── cleaned_table_3.12.0.csv   (Cleaned dataset)

  ├── cleaned_table_3.12.1.csv   (Cleaned dataset)

  ├── plot_3.12.py               (Plots graphs)
```

4. Code Documentation

4.1 Data Extraction (script.py):

The following script was used to extract tabular data from the “**Survey on Household Consumption Expenditure: 2022-23”** PDF file:

```
import pdfplumber
import pandas as pd
pdf_path = "Report_591_HCES_2022-23New.pdf"    # Path to the HCES 2022-23 PDF file

# Keywords to detect relevant tables
keywords = ["Average MPCE across fractile classes of MPCE",
           "Average MPCE and urban-rural differences in MPCE in 2022-23",
           "Absolute and percentage break-up of MPCE by item groups in 2022-23",
           "Average MPCE (Rs.) by social group in 2022-23",]

extracted_tables = []
with pdfplumber.open(pdf_path) as pdf:      # Open the PDF
   for page in pdf.pages:
       text = page.extract_text()
       if text and any(keyword in text for keyword in keywords):   # Check if page contains relevant data
           table = page.extract_table()
           if table:
               extracted_tables.append(table)

# Convert extracted tables into a structured DataFrame
dataframes = []
```

```
for table in extracted_tables:
   df = pd.DataFrame(table)
   dataframes.append(df)

# Merge all extracted tables into a single CSV file
if dataframes:
   final_df = pd.concat(dataframes, ignore_index=True)
   final_df.to_csv("script.csv", index=False)
   print("Extracted relevant data and saved as 'script.csv'!")
else:
   print("No relevant tables found!")
```

4.2 Data Cleaning (Example for table 3.2 - table_3.2pd.py):

For tables that required data cleaning, the following **python** script was used (**example for Table 3.2**):

```
import pandas as pd
df = pd.read_csv("./table_3.2/table_3.2.csv", encoding="utf-8")# Load raw CSV file
df_cleaned_1 = df.iloc[2:].reset_index(drop=True)     # Remove extra header rows
df_cleaned_2 = df.iloc[2:].reset_index(drop=True)
# Rename columns
df_cleaned_1.columns = ["Major State", "Rural MPCE", "Urban MPCE", "Rural-Urban Difference (%)"]
df_cleaned_2.columns = ["Major State", "Rural MPCE", "Urban MPCE", "Rural-Urban Difference (%)"]
# Keep only relevant columns
df_cleaned_1 = df_cleaned_1[["Major State", "Rural MPCE", "Urban MPCE"]]
```

```
df_cleaned_2 = df_cleaned_2[["Major State", "Rural-Urban Difference (%)"]]


# Convert MPCE values to floats
df_cleaned_1["Rural MPCE"] = df_cleaned_1["Rural MPCE"].str.replace(",", "").astype(float)
df_cleaned_1["Urban MPCE"] = df_cleaned_1["Urban MPCE"].str.replace(",", "").astype(float)
df_cleaned_2["Rural-Urban Difference (%)"] = df_cleaned_2["Rural-Urban Difference (%)"].astype(float)
# Save the cleaned file
df_cleaned_1.to_csv("./table_3.2/cleaned_table_3_2.0.csv", index=False)
df_cleaned_2.to_csv("./table_3.2/cleaned_table_3_2.1.csv", index=False)
print("Data cleaning completed successfully!")
```

4.3 Data Visualization:

Table 3.1 (plot_3.1.py) - **Bar Graph**:

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


df = pd.read_csv("./table_3.1/cleaned_table_3_1.csv") # Load the cleaned CSV file
plt.figure(figsize=(12, 6))


# Extract data for plotting
x_labels = df["Fractile Class"]
rural_values = df["Rural MPCE"]
urban_values = df["Urban MPCE"]
```

```
x = range(len(x_labels))
bar_width = 0.4

# Plot bars for Rural and Urban MPCE
plt.bar(x, rural_values, width=bar_width, label="Rural MPCE", color="skyblue",
alpha=0.8)
plt.bar([p + bar_width for p in x], urban_values, width=bar_width, label="Urban MPCE",
color="salmon", alpha=0.8)

# Labels and Titles
plt.xlabel("Fractile Class of MPCE")
plt.ylabel("Average MPCE (₹)")
plt.title("Comparison of Rural vs. Urban Average MPCE Across Fractile Classes")
plt.xticks([p + bar_width / 2 for p in x], x_labels, rotation=45)
plt.legend()
plt.grid()
plt.show()
```

Table 3.2 (plot_3.2.py) - **Bar Graph**:

```
import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_csv("./table_3.2/cleaned_table_3_2.0.csv")# Load cleaned Table 3.2.1 CSV
plt.figure(figsize=(14, 6))

x_labels = df["Major State"]      # Extract data for plotting
rural_values = df["Rural MPCE"]
urban_values = df["Urban MPCE"]
x = range(len(x_labels))
```

```
bar_width = 0.3
# Plot bars for Rural and Urban MPCE
plt.bar(x, rural_values, width=bar_width, label="Rural MPCE", color="skyblue", alpha=0.8)
plt.bar([p + bar_width for p in x], urban_values, width=bar_width, label="Urban MPCE", color="salmon", alpha=0.8)

plt.xlabel("Major States")
plt.ylabel("Average MPCE (₹)")
plt.title("Comparison of Rural vs. Urban Average MPCE Across Major States")
plt.xticks([p + bar_width / 2 for p in x], x_labels, rotation=45)
plt.legend()
plt.grid()
plt.show()

df = pd.read_csv("./table_3.2/cleaned_table_3_2.1.csv")# Load cleaned Table 3.2.1 CSV
plt.figure(figsize=(10, 8))

difference = df["Rural-Urban Difference (%)"]
state = df["Major State"]

plt.barh(state, difference, color = "salmon", height = 0.5)# plot horizontal bar-graph

plt.xlabel("Rural-Urban Difference (%)", fontsize=12)
plt.ylabel("State", fontsize=12)
plt.title("Urban-Rural MPCE Difference (%) by State (2022-23)", fontsize=14)
plt.grid()
plt.show()
```

Table 3.5 (plot_3.5.py) - **Nested pie-chart:**

```
import pandas as pd

import matplotlib.pyplot as plt


def plot_nested_pie(csv_file, title):

   df = pd.read_csv(csv_file)


   # Separate major categories and subcategories

   major = df[df['Category'] == 'Major']

   subcats = df[df['Category'] != 'Major']


   # Data for the pie charts

   inner_sizes, inner_labels = major['Percentage'], major['Item']

   outer_sizes, outer_labels = subcats['Percentage'], subcats['Item']


   # Colors for the pie charts

   inner_colors = ['#ff9999', '#66b3ff']

   outer_colors = ["#2980B9", "#5D6D7E", "#1ABC9C", "#D35400", "#E74C3C", "#2980B9", "#9B59B6", "#16A085", "#F39C12", "#8E44AD", "#34495E",

                   "#F1C40F", "#7F8C8D", "#27AE60", "#8E44AD", "#C0392B", "#D5DBDB", "#F5B041", "#85929E", "#A569BD", "#1F618D", "#F7DC6F"]


   # Create the nested pie chart

   fig, ax = plt.subplots(figsize=(10, 8))


   # Outer pie chart (subcategories)
```

```
    wedges_outer, _, _ = ax.pie(outer_sizes, colors=outer_colors, startangle=90, 
autopct='%1.1f%%', pctdistance=0.85)

    # Inner pie chart (major categories)
    wedges_inner, _, _ = ax.pie(inner_sizes, colors=inner_colors, radius=0.7, 
startangle=90, autopct='%1.1f%%', pctdistance=0.75)

    # Draw a circle in the center to create the donut effect
    ax.add_artist(plt.Circle((0, 0), 0.4, color='white'))

    # Equal aspect ratio ensures that pie is drawn as a circle
    ax.axis('equal')

    # Add a title
    plt.title(title)

    # Add a legend
    plt.legend(wedges_inner + wedges_outer, inner_labels.tolist() + 
outer_labels.tolist(),
               title="Items", loc="center left", bbox_to_anchor=(1, 0.5))

    # Adjust layout and display
    plt.tight_layout()
    plt.show()

# Plot the first CSV file
plot_nested_pie('./table_3.5/table_3.5.0.csv', 'Percentage break-up of MPCE by item 
groups in Rural India in 2022-23: All-India')

# Plot the second CSV file
```

```
plot_nested_pie('./table_3.5/table_3.5.1.csv', 'Percentage break-up of MPCE by item groups in Urban India in 2022-23: All-India')
```

Table 3.12 (plot_3.12.py) - **Heat-map:**

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df_rural = pd.read_csv("./table_3.12/cleaned_table_3.12.0.csv")
df_urban = pd.read_csv("./table_3.12/cleaned_table_3.12.1.csv")

# Set "State" as the index for heatmap plotting
df_rural.set_index("State", inplace=True)
df_urban.set_index("State", inplace=True)
plt.figure(figsize=(12, 8))

# Create a heatmap for Rural MPCE
sns.heatmap(df_rural, cmap="YlGn", annot=True, fmt=".0f", linewidths=0.5)

plt.title("Average Rural MPCE (₹) by Social Group in 2022-23, Major States", fontsize=14)
plt.xlabel("Social Group", fontsize=12)
plt.ylabel("State", fontsize=12)
plt.figtext(0.5, 0.01, "# includes not reporting cases (i.e., those who have not reported social group) also.",  ha="center", fontsize=8, style="italic")

plt.show()

# Create a separate heatmap for Urban MPCE
plt.figure(figsize=(12, 8))
```

```
sns.heatmap(df_urban, cmap="YlGn", annot=True, fmt=".0f", linewidths=0.5)


plt.title("Average Urban MPCE (₹) by Social Group in 2022-23, Major States", fontsize=14)

plt.xlabel("Social Group", fontsize=12)

plt.ylabel("State", fontsize=12)

plt.figtext(0.5, 0.01, "# includes not reporting cases (i.e., those who have not reported social group) also.",  ha="center", fontsize=8, style="italic")

plt.show()
```

5. Conclusion

This document summarizes the software and scripts used in **extracting, cleaning, and visualizing data**. By automating the process using Python, I ensured **data accuracy, efficient processing, and clear visual representation**, making the analysis useful for policymakers and stakeholders.