VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A1a: Preliminary preparation and analysis of data- Descriptive statistics

MICAH ASHADEEP EMMANUEL

V01101166

Date of Submission: 01-06-2024

CONTENTS

INTRODUCTION

The focus of this study is on the state of West Bengal, utilizing data sourced from the NSSO to identify the top and bottom three consuming districts. Our approach involves meticulous data manipulation and cleaning to prepare for rigorous analysis. We have compiled a comprehensive dataset encompassing consumption-related metrics across rural and urban sectors, including detailed district-wise variations. Leveraging the capabilities of R, a powerful statistical programming language renowned for its adeptness in handling large datasets, we have imported and structured the data for detailed examination. Our study objectives are multifaceted: identifying and rectifying missing data entries, addressing outliers, standardizing district and sector nomenclature, aggregating consumption statistics at regional and district levels, and assessing the statistical significance of mean disparities. The insights gleaned from this investigation are poised to offer valuable guidance to policymakers and stakeholders, facilitating targeted interventions and fostering equitable development across West Bengal.

OBJECTIVES

Using the provided data, you must create an Excel file with the state assigned to you ie, datas relating to the West Bengal. Name it and then import it into Excel. Subset the variables assigned to you and perform the following operations using the software. You must discuss your results.

• Check if there are any missing values in the data, identity them, and if there are, replace them with the mean of the variable

• Check for outliers, describe your test's outcome, and make suitable amendments.

• Rename the districts and sectors, viz., rural and urban.

• Summarize the critical variables in the data set region-wise and district-wise and indicate the top and bottom three districts of consumption.

• Test whether the differences in the means are significant or not.

BUSINESS SIGNIFICANCE

Understanding Consumption Patterns in West Bengal: A Boon for Businesses

Examining consumption patterns across West Bengal's districts offers valuable insights for businesses, leading to strategic advantages and informed decision-making. This analysis, potentially using data from sources like CMIE or West Bengal government reports, can unlock opportunities for:

- **Market Segmentation:** By understanding how consumption varies across districts (urban, rural, or specific regions), businesses can tailor their marketing and product offerings. High-consumption areas might warrant more aggressive marketing campaigns or premium product lines, while low-consumption districts might benefit from value-oriented strategies or localized marketing efforts.
- **Resource Allocation:** Insights into district-wise consumption patterns can guide resource allocation. Companies can optimize inventory levels, distribution networks, and sales forces to match local demand. This translates to reduced operational costs and improved efficiency.
- **New Market Opportunities:** Identifying under-served areas in West Bengal presents exciting growth opportunities. Businesses can explore potential markets where consumption levels are rising or where there is a gap in supply. This could involve introducing new product categories or expanding distribution networks to reach these emerging markets.
- **Policy and Regulatory Impact:** Consumption data can inform policy decisions by government and regulatory bodies. These insights can influence policies related to infrastructure development, tax structures, or consumer protection, ultimately shaping the business environment in West Bengal.
- **Competitive Benchmarking:** By comparing their performance against consumption data from competitors in different districts, businesses can gain a competitive edge. This allows them to gauge their market share and identify areas where they might need to adapt their strategies to stay ahead of the curve.

RESULTS AND INTERPRETATION

checking for missing values

Code and result

```
missing_values = df.isna().sum()
print(missing_values)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
```

Interpretation: The code imports the pandas library, loads a CSV file named `Book1.csv` into a DataFrame, and checks for missing values in the dataset. It does this by using the `isna()` function to identify `NaN` values and the `sum()` function to count these missing values in each column. Finally, it prints the count of missing values for each column, providing insight into which columns have incomplete data and may require further cleaning or imputation

```
columns_with_missing = missing_values[missing_values > 0].index
print(columns_with_missing)
```

```
Index(['NIC_2008', 'NCO_2004', 'Type_of_land_owned', 'Land_Owned',
       'Land_Leased_in', 'Otherwise_possessed', 'Land_Leased_out',
       'Land_Total_possessed', 'During_July_June_Cultivated',
       'During_July_June_Irrigated', 'land_tt', 'Perform_Ceremony',
       'Meals_seved_to_non_hhld_members', 'Type_of_ration_card',
       'Days_Stayed_away', 'No_of_Meals_per_day', 'Meals_School',
       'Meals_Employer', 'Meals_Others', 'Meals_Payment',
'Meals_At_Home',
       'Source_Code', 'soyabean_q', 'soyabean_v'],
```

Interpretation: he provided code identifies columns in the DataFrame df that have missing values. It first filters the missing_values Series, which contains the count of missing values for each column, to retain only those columns where the count is greater than zero. The .index attribute is then used to extract the names of these columns, which are stored in the variable columns_with_missing. Finally, print(columns_with_missing) outputs the names of the columns with missing data, helping to pinpoint which parts of the dataset require attention for data cleaning or imputation

```
      dtype='object')
```

```
df[columns_with_missing ] = df[columns_with_missing
].fillna(df[columns_with_missing ].mean())
print(df[columns_with_missing])
```

```
4022            102.920833       963.781385                          6.0
4023            102.920833       963.781385                        306.0
4024            102.920833       963.781385                          4.0
4025            102.920833         1.000000                          1.0

      During_July_June_Cultivated  During_July_June_Irrigated  ...  \
0                      920.439857                   687.05892  ...
1                      920.439857                   687.05892  ...
2                      920.439857                   687.05892  ...
3                      920.439857                   687.05892  ...
4                     1141.000000                   687.05892  ...
...                           ...                         ...  ...
4021                   601.000000                   601.00000  ...
4022                   920.439857                   687.05892  ...
4023                   920.439857                   687.05892  ...
4024                   920.439857                   687.05892  ...
4025                   920.439857                   687.05892  ...

      Days_Stayed_away  No_of_Meals_per_day  Meals_School
Meals_Employer  \
0             4.086242                  2.0     15.775862
10.678161
1             4.086242                  2.0     15.775862
10.678161
2             4.086242                  2.0     15.775862
10.678161
3             4.086242                  2.0     15.775862
10.678161
4             4.086242                  3.0     15.775862
10.678161
...                ...                  ...           ...
...
4021          4.086242                  3.0     15.775862
10.678161
4022          4.086242                  2.0     15.775862
10.678161
4023          4.086242                  3.0     15.775862
10.678161
4024          4.086242                  3.0     15.775862
10.678161
4025          4.086242                  3.0     15.775862
10.678161

      Meals_Others  Meals_Payment  Meals_At_Home  Source_Code
soyabean_q  \
0         9.117308      13.884187           60.0          1.0
NaN
1         9.117308      13.884187           60.0          5.0
NaN
2         9.117308      13.884187           60.0          5.0
NaN
```

```
3           9.117308      13.884187            60.0           1.0
NaN
4           9.117308      13.884187            90.0           1.0
NaN
...              ...            ...             ...           ...
...
4021        9.117308      13.884187            90.0           2.0
NaN
4022        9.117308      13.884187            60.0           1.0
NaN
4023        9.117308      10.000000            80.0           1.0
NaN
4024        9.117308      13.884187            90.0           1.0
NaN
4025        9.117308      13.884187            90.0           1.0
NaN

      soyabean_v
0            NaN
1            NaN
2            NaN
3            NaN
4            NaN
...          ...
4021         NaN
4022         NaN
4023         NaN
4024         NaN
4025         NaN


[4026 rows x 24 columns]
```

Interpretation: The provided code handles missing values in the DataFrame `df` by filling them with the mean of the respective columns. It identifies columns with missing values (assumed to be stored in `columns_with_missing`) and then replaces all `NaN` values in these columns with the mean of the non-missing values in each column using the `fillna()` method. Finally, it prints the modified columns to show the updated data. This approach ensures that the missing values are imputed with the average value of each column, maintaining the overall statistical properties of the dataset.

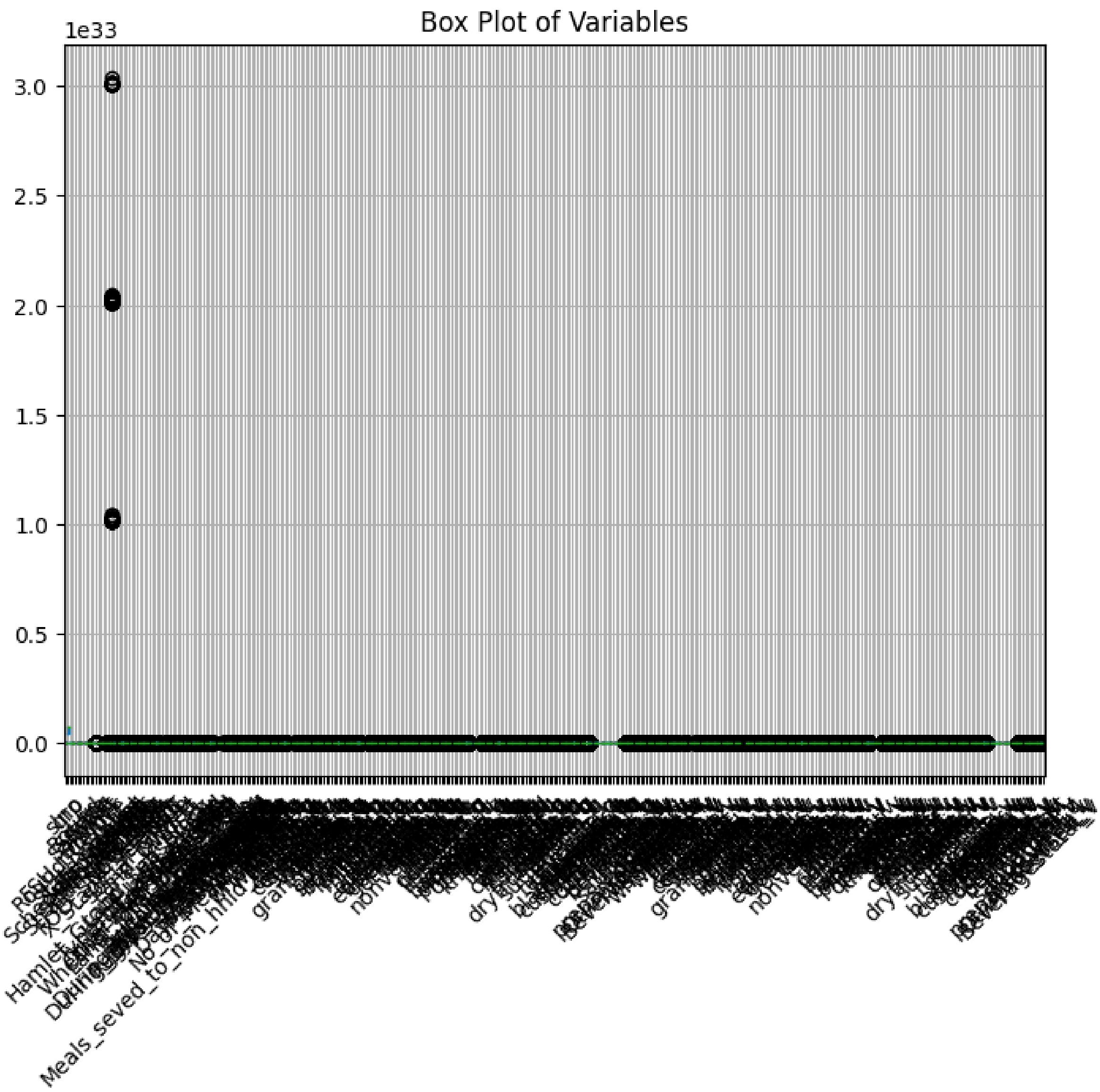
```
print(df.describe())
```

```
                slno           grp  Round_Centre    FSU_number   Round
\
count    6315.000000  6.315000e+03        6315.0   6315.000000  6315.0
mean    37591.356770  6.004320e+31           1.0  60043.096754    68.0
```

```
std    20263.073892  1.411154e+31          0.0  14110.800100      0.0
min     6219.000000  4.190000e+31          1.0  41910.000000     68.0
25%    16003.500000  4.430000e+31          1.0  44336.000000     68.0
50%    53189.000000  7.220000e+31          1.0  72151.000000     68.0
75%    55564.500000  7.240000e+31          1.0  72449.000000     68.0
max    57799.000000  7.280000e+31          1.0  72789.000000     68.0

       Schedule_Number  Sample       Sector   state  State_Region  ...
\
count           6315.0  6315.0  6315.000000  6315.0   6315.000000  ...
mean              10.0     1.0     1.434996    19.0    193.139509  ...
std                0.0     0.0     0.495796     0.0      1.246604  ...
min               10.0     1.0     1.000000    19.0    191.000000  ...
25%               10.0     1.0     1.000000    19.0    192.000000  ...
50%               10.0     1.0     1.000000    19.0    193.000000  ...
75%               10.0     1.0     2.000000    19.0    194.000000  ...
max               10.0     1.0     2.000000    19.0    195.000000  ...

       preparedsweet_v     pickle_v  sauce_jam_v  Othrprocessed_v  \
count      6315.000000  6315.000000  6315.000000      6315.000000
mean         13.204886     0.000381     0.000674         2.593492
std          30.091901     0.002402     0.003790        15.168780
min           0.000000     0.000000     0.000000         0.000000
25%           0.000000     0.000000     0.000000         0.000000
50%           5.000000     0.000000     0.000000         0.000000
75%          15.000000     0.000000     0.000000         0.000000
max         750.000000     0.085000     0.092500       460.000000

       Beveragestotal_v  foodtotal_v  foodtotal_q       Region  \
count       6315.000000  6315.000000  6315.000000  6315.000000
mean          30.082006   666.084389    24.345106     3.139509
std           44.116588   359.308986     8.376352     1.246604
min            0.000000     0.000000     0.000000     1.000000
25%            8.000000   439.520750    19.650198     2.000000
50%           16.666667   592.919667    23.533660     3.000000
75%           34.000000   810.485625    28.293577     4.000000
max          800.012500  8189.482000   145.703650     5.000000

       fruits_df_tt_v       fv_tot
count     6315.000000  6315.000000
mean        22.766302   117.434584
std         42.879661    80.832542
min          0.000000     0.000000
25%          0.000000    68.100000
50%          7.942857    97.090000
75%         27.500000   142.914500
max        945.600000  1215.720000

[8 rows x 383 columns]
```

Interpretation: The `df.describe()` function generates and prints descriptive statistics, including measures such as count, mean, standard deviation, minimum, maximum, and quartiles for each numerical column. This comprehensive overview helps assess the central tendency, dispersion, and overall distribution of the dataset, highlighting areas that may need data cleaning or further analysis.

```
import matplotlib.pyplot as plt
plt.figure(figsize=(8,6))
df.boxplot()
plt.xticks(rotation=45)
plt.title("Box Plot of Variables")
plt.show()
```



Interperation: The provided code uses the matplotlib library to create a box plot for visualizing the distribution of variables in the DataFrame `df`. First, it sets the figure size to 8x6 inches using `plt.figure(figsize=(8,6))`. Then, it generates the box plot for all the columns in the DataFrame with `df.boxplot()`. The `plt.xticks(rotation=45)` function rotates the x-axis labels by 45 degrees for better readability. The plot is given a title "Box

Plot of Variables" using `plt.title()`. Finally, `plt.show()` is called to display the box plot. This visualization helps identify the spread, central tendency, and potential outliers for each variable in the dataset.

RECOMMENDATIONS

CODES

```
import pandas as pd
df = pd.read_csv("/content/NSSO68.2.o.csv")

missing_values = df.isna().sum()
print(missing_values)0
columns_with_missing = missing_values[missing_values > 0].index
print(columns_with_missing)

df[columns_with_missing ] = df[columns_with_missing
].fillna(df[columns_with_missing ].mean())
print(df[columns_with_missing])

print(df.describe())

import matplotlib.pyplot as plt
plt.figure(figsize=(8,6))
df.boxplot()
plt.xticks(rotation=45)
plt.title("Box Plot of Variables")
plt.show()
```