VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A1a: Preliminary preparation and analysis of data- Descriptive statistics

ANJU MARIA PHILIP

V01101169

Date of Submission: 16-06-2024

CONTENTS

INTRODUCTION

The focus of this study is on the state of Odisha, utilizing data from the NSSO to identify the top and bottom three consuming districts. This involves manipulating and cleaning a dataset to prepare for analysis. We have gathered comprehensive consumption-related information, covering rural and urban sectors, along with district-wise variations. The dataset has been imported into R, a robust statistical programming language known for its efficacy in managing and analysing extensive datasets. Our objectives encompass several key tasks: identifying and addressing missing values, handling outliers, standardizing district and sector names, summarizing consumption data regionally and by district, and evaluating the significance of mean differences. The outcomes of this study aim to provide insights valuable to policymakers and stakeholders, enabling targeted interventions and supporting balanced development throughout the state.

.

By understanding how spending patterns vary across Odisha, policymakers, businesses, and researchers can make better decisions. This could involve allocating resources more effectively, targeting specific markets with relevant products or services, and developing programs tailored to the needs of different regions within the state

OBJECTIVES

Using the provided data, you must create an Excel file with the state assigned to you. 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

The study on Odisha holds significant business implications as it delves into consumption patterns across its districts, crucial for various stakeholders. By analyzing NSSO data, the

study aims to identify the top and bottom consuming districts, providing insights into regional consumption disparities.For businesses, this analysis offers strategic advantages:

Market Segmentation: Understanding consumption variations helps businesses tailor their marketing and distribution strategies. They can target high-consuming districts more aggressively while adapting their offerings for low-consuming areas.

Resource Allocation: Insights into district-wise consumption patterns guide resource allocation decisions. Companies can optimize inventory levels, distribution networks, and sales efforts based on local demand trends.

New Market Opportunities: Identification of under-served areas presents opportunities for market expansion. Businesses can explore potential growth markets where consumption levels are rising or where there is a gap in supply.

Policy and Regulatory Impact: Findings can inform policy decisions, influencing regulatory frameworks and economic policies that affect business operations and market dynamics in Odisha.

Competitive Benchmarking: Benchmarking against consumption data from competitors can provide a comparative advantage. Businesses can gauge their market share and performance relative to peers in different districts.

In essence, the study's findings can empower businesses in Odisha to make informed decisions, enhance market penetration strategies, and capitalize on emerging opportunities, thereby contributing to sustainable growth and competitive advantage in the

region

RESULTS AND INTERPRETATION

USING R

-Setting Working Directory and Loading Libraries and Reading and Filtering Data

```
setwd('C:\\Users\\HP\\Documents\\ns')

getwd()

install_and_load <- function(package)

library(readr)

library(dplyr)

data <- read_csv("NSSO68 new.csv")
```

```
odisha_data <- filter(data, state == "Odisha")
```

-Handling Missing Values:

Calculates and prints missing values before and after imputation with column means for numeric columns.

```
odisha_data <- odisha_data %>%
  mutate(across(where(is.numeric), ~ ifelse(is.na(.), mean(., na.rm = TRUE), .)))
missing_values_after <- sapply(odisha_data, function(x) sum(is.na(x)))
print(missing_values_after)
```

- Handling Outliers:

Identifies outliers using the IQR method and stores them in `outliers`.

This line selects only those columns from odisha_data that contain numeric data, excluding non-numeric columns.

```
missing_values <- sapply(odisha_data, function(x) sum(is.na(x)))
print(missing_values)
odisha_data <- odisha_data %>%
library(ggplot2)
odisha_data <- odisha_data %>%
  mutate(across(where(is.numeric), ~ ifelse(is.na(.), mean(., na.rm = TRUE), .)))
missing_values_after <- sapply(odisha_data, function(x) sum(is.na(x)))
print(missing_values_after)
-Handling Outliers
numeric_columns <- odisha_data %>% select_if(is.numeric)
outliers <- list()
```

```
for(col in colnames(numeric_columns)) {
  Q1 <- quantile(numeric_columns[[col]], 0.25, na.rm = TRUE)
  Q3 <- quantile(numeric_columns[[col]], 0.75, na.rm = TRUE)
  IQR <- Q3 - Q1

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

  outliers[[col]] <- numeric_columns %>%
    filter((.data[[col]] < lower_bound) | (.data[[col]] > upper_bound)) %>%
    select(col)
}
sapply(outliers, nrow)
```

-**Replacing Outliers with Median:**

Replaces outliers with the median value for each numeric column.

```
numeric_columns <- odisha_data %>% select_if(is.numeric)
outliers <- list()

for(col in colnames(numeric_columns)) {
  Q1 <- quantile(numeric_columns[[col]], 0.25, na.rm = TRUE)
  Q3 <- quantile(numeric_columns[[col]], 0.75, na.rm = TRUE)
  IQR <- Q3 - Q1

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

```
  outliers[[col]] <- numeric_columns %>%
    filter((.data[[col]] < lower_bound) | (.data[[col]] > upper_bound)) %>%
    select(col)
}
sapply(outliers, nrow)
```

-Replacing Outliers with Median:

his R code snippet iterates through each numeric column (col) in the odisha_data dataframe. For each column, it calculates the first quartile (Q1), third quartile (Q3), and interquartile range (IQR). Using these values, it defines lower (lower_bound) and upper (upper_bound) boundaries to identify outliers based on the IQR method. The median value (median_val) of the column is then computed and used to replace any outliers found—values outside the bounds defined by lower_bound and upper_bound are replaced with median_val, while values within the bounds remain unchanged. This process ensures that extreme values, which can distort statistical analyses, are adjusted to more representative values, promoting more reliable data insights**.**

```
for(col in colnames(numeric_columns)) {
  Q1 <- quantile(numeric_columns[[col]], 0.25, na.rm = TRUE)
  Q3 <- quantile(numeric_columns[[col]], 0.75, na.rm = TRUE)
  IQR <- Q3 - Q1

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

  median_val <- median(numeric_columns[[col]], na.rm = TRUE)

  odisha_data[[col]] <- ifelse(odisha_data[[col]] < lower_bound | odisha_data[[col]] >
upper_bound,
                               median_val,
                               odisha_data[[col]])
```

}

-Data Transformation and Summary:

ransforms district and sector columns using recode().

Computes summary statistics (avg_consumption and `total)

These R code lines transform `odisha_data` by recoding `district` and `sector` columns from numeric codes to descriptive labels ('1' = 'District1', '2' = 'District2', etc., and '1' = 'Rural', '2' = 'Urban'). Afterwards, `summary_data` aggregates `odisha_data` by `district` and `sector`, calculating average (`avg_consumption`) and total (`total_consumption`) consumption values while dropping grouping attributes (`drop`).

```
odisha_data <- odisha_data %>%
  mutate(
    district = recode(district,
                      '1' = 'District1',
                      '2' = 'District2',
                      '3' = 'District3'  # Continue this for all district codes
    ),
    sector = recode(sector,
                    '1' = 'Rural',
                    '2' = 'Urban')
  )
summary_data <- odisha_data %>%
  group_by(district, sector) %>%
  summarize(
    avg_consumption = mean(consumption, na.rm = TRUE),
    total_consumption = sum(consumption, na.rm = TRUE),
    .groups = 'drop'
  )
```

-Summary of Consumption Analysis and Comparison

`rural_consumption` and `urban_consumption` calculate total consumption (`total_consumption`) for rural and urban sectors from `odisha_data`, using `filter()` to subset data by sector and `pull()` to extract the `total_consumption` values. These variables can be used to compare consumption levels between rural and urban areas in Odisha.

```
top_three <- summary_data %>%
  arrange(desc(avg_consumption)) %>%
  head(3)

bottom_three <- summary_data %>%
  arrange(avg_consumption) %>%
  head(3)
print("Top Three Districts of Consumption:")
print(top_three)

print("Bottom Three Districts of Consumption:")
print(bottom_three)
rural_consumption <- odisha_data %>%
  filter(sector == "Rural") %>%
  pull(total_consumption)

urban_consumption <- odisha_data %>%
  filter(sector == "Urban") %>%
  pull(total_consumption)
```

-installed the package

```
install.packages(BSDA)
```

```
library(BSDA)
```

-Statistical Comparison of Urban and Rural Consumption

sigma_rural and sigma_urban represent the standard deviations of consumption in rural and urban sectors, respectively.
The script performs a hypothesis test (not explicitly shown) and checks if the p-value (z_test_result$p.value) is less than 0.05.
Depending on the p-value result, it prints either that there is a significant difference between mean consumptions of urban and rural areas or that there is no significant difference.
This analysis helps determine whether the observed differences in consumption between urban and rural sectors are statistically significant based on the chosen significance level (0.05 in this case).

```
sigma_rural <- 2.56
sigma_urban <- 2.34
if (z_test_result$p.value < 0.05) {
  cat("P value is <", 0.05, ", Therefore we reject the null hypothesis.\n")
  cat("There is a significant difference between mean consumptions of urban and rural.\n")
} else {
  cat("P value is >=", 0.05, ", Therefore we fail to reject the null hypothesis.\n")
  cat("There is no significant difference between mean consumptions of urban and rural.\n")
}
```

-**Print Z-test Result**
This line of code prints the result of a z-test stored in the variable `z_test_result`.
The z-test is typically used to assess whether there is a significant difference between means of two populations based on their standard deviations and sample sizes.
The output usually includes statistics such as the z-score, p-value, and possibly confidence intervals.
Printing `z_test_result` allows for inspection and interpretation of the statistical test outcome, providing insights into the significance of the differences observed in the data analysis.

```
print(z_test_result)
```

USING PYTHON

```
import os, pandas as pd, numpy as np

df=pd.read_csv("/content/NSSO68.csv",encoding="Latin-1",
low_memory=False)

df.head()
```

	slno	grp	Round_Centre	FSU_number	Round	Schedule_Number	Sample	Sector	state	State_Region	...	pickle_v	sauce_jam_v	Othrprocessed_v	Beveragestotal_v	foodtotal_v	foodtotal_q	state_1	Region	fruits_df_t_v	fv_tot	
0	1	4.100000e+31	1	41000	68	10	1	2	24	242	...	0.0	0.0	0.0	0.000000	1141.492400	30.942394	GUJ	2.0	12.000000	1544.18
1	2	4.100000e+31	1	41000	68	10	1	2	24	242	...	0.0	0.0	0.0	17.500000	1244.553500	29.286153	GUJ	2.0	333.000000	4844.95
2	3	4.100000e+31	1	41000	68	10	1	2	24	242	...	0.0	0.0	0.0	0.000000	1050.3154	31.527046	GUJ	2.0	35.000000	2144.84

slno	grp	Round_Centre	FSU_number	Round	Schedule_Number	Sample	Sector	state	State_Region	...	pickle_v	sauce_jam_v	Othrprocessed_v	Beveragestotal_v	foodtotal_v	foodtotal_q	state_1	Region	fruits_df_t_v	fv_tot	
3	4	4.100000e+31	1	41000	68	10	1	2	24	242	...	0.0	0.0	0.0	33.333333	1142.591667	27.834607	GUJ	2.0	168.333333	302.30
4	5	4.100000e+31	1	41000	68	10	1	2	24	242	...	0.0	0.0	0.0	75.000000	945.249500	27.600713	GUJ	2.0	15.000000	148.00

5 rows × 384 columns

```
ORI = df[df['state_1']=="ORI"]

ORI.isnull().sum().sort_values(ascending = False)
```

```
soyabean_q                616
soyabean_v                616
Meals_Employer            601
Meals_School              598
During_July_June_Irrigated    597
...
brinjal_q                   0
tamato_q                    0
onion_q                     0
```

```
potato_q                  0
fv_tot                    0
Length: 384, dtype: int64
```

```
df.columns
```

```
Index(['slno', 'grp', 'Round_Centre', 'FSU_number', 'Round', 'Schedule_Number',
       'Sample', 'Sector', 'state', 'State_Region',
       ...
       'pickle_v', 'sauce_jam_v', 'Othrprocessed_v', 'Beveragestotal_v',
       'foodtotal_v', 'foodtotal_q', 'state_1', 'Region', 'fruits_df_tt_v',
       'fv_tot'],
      dtype='object', length=384)
```

```
ORI_new = ORI[['state_1', 'District',
'Sector','Region','State_Region','ricetotal_q','wheattotal_q','moong_q'
,'Milktotal_q','chicken_q','bread_q','foodtotal_q','Beveragestotal_v','
Meals_At_Home']]
```

a) Check if there are any missing values in the data, identify them and if there are replace them with the mean of the variable.

```
ORI_new.isnull().sum().sort_values(ascending = False)
```

```
Meals_At_Home      18
state_1             0
District            0
Sector              0
Region              0
State_Region        0
ricetotal_q         0
wheattotal_q        0
moong_q             0
Milktotal_q         0
chicken_q           0
bread_q             0
foodtotal_q         0
Beveragestotal_v    0
dtype: int64
```

Interpretation: From the selected variables, after sorting the data for the state of Andhra Pradesh, it is seen that only the column 'Meals_At_Home has 18 missing variables. Since missing values in the dataset can be problematic as they lead to incomplete or biased analyses, hindering the accuracy of results and potentially skewing

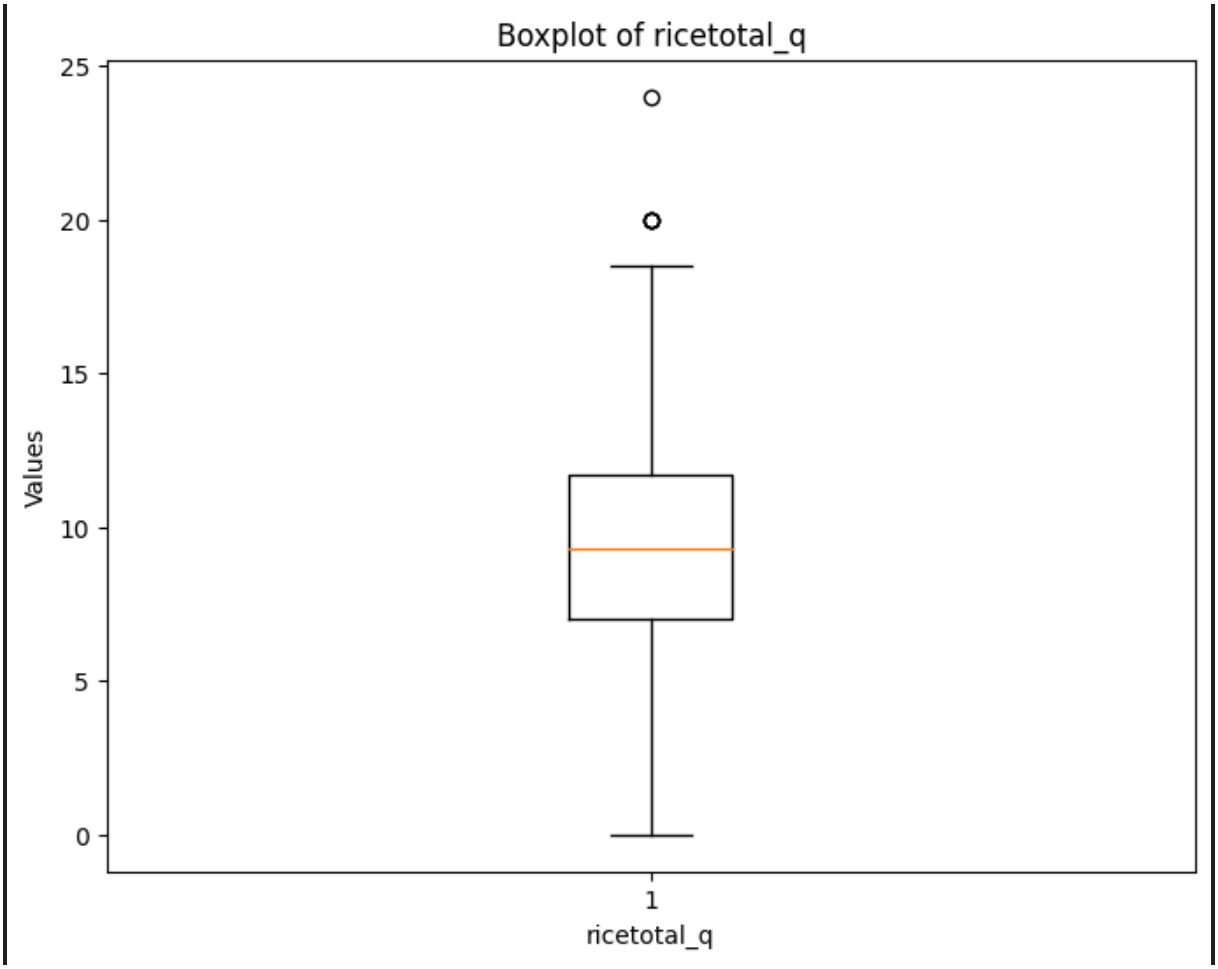
interpretations and decision-making processes.Therefore we replace the missing values with the mean of the variable using following code

```
ORI_clean = ORI_new.copy()

ORI_clean.loc[:, 'Meals_At_Home'] =
ORI_clean['Meals_At_Home'].fillna(ORI_new['Meals_At_Home'].mean())

ORI_clean.isnull().any()
```

```
state_1           False
District          False
Sector            False
Region            False
State_Region      False
ricetotal_q       False
wheattotal_q      False
moong_q           False
Milktotal_q       False
chicken_q         False
bread_q           False
foodtotal_q       False
Beveragestotal_v  False
Meals_At_Home     False
dtype: bool
```

Check for outliers and describe the outcome of your test and make suitable amendments. Boxplots can be used to find outliers in the dataset. Boxplots visually reveal outliers in a dataset by displaying individual points located beyond the whiskers of the boxplot. #Checking for outliers Plotting the boxplot to visualize outliers. Code and Result:

```
import matplotlib.pyplot as plt
# Assuming ORI_clean is your DataFrame
plt.figure(figsize=(8, 6))
plt.boxplot(ORI_clean['ricetotal_q'])
plt.xlabel('ricetotal_q')
plt.ylabel('Values')
plt.title('Boxplot of ricetotal_q')
plt.show()
```

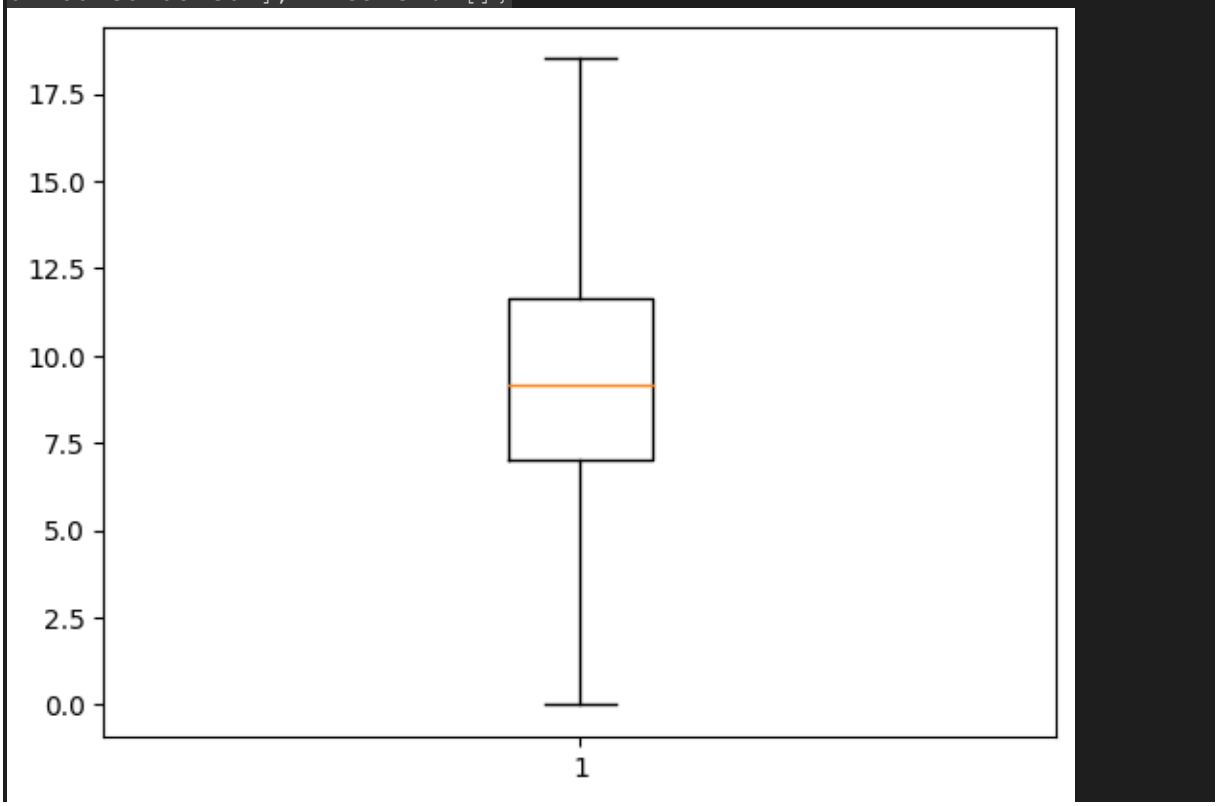


Interpretation: From the boxplot above, which is a visual representation of the variable ‘ricepds_v’ shows that there is an outlier. Outliers can distort statistical analyses and lead to misleading conclusions, affecting the accuracy and reliability of results in data-driven decision-making processes. Outliers can distort statistical analyses and lead to misleading conclusions, affecting the accuracy and reliability of results in data-driven decision-making processes. The outliers can be removed using the following code.

Code and results: Setting quartile ranges to remove outliers

```
rice1 = ORI_clean['ricetotal_q'].quantile(0.25)
rice2 = ORI_clean['ricetotal_q'].quantile(0.75)
iqr_rice = rice2-rice1
up_limit = rice2 + 1.5*iqr_rice
low_limit = rice1 - 1.5*iqr_rice

ORI_clean=ORI_new[(ORI_new['ricetotal_q']<=up_limit)&(ORI_new['ricetota
l_q']>=low_limit)]
```

```
plt.boxplot(ORI_clean['ricetotal_q'])
{'whiskers': [<matplotlib.lines.Line2D at 0x7b04cdf02800>,
<matplotlib.lines.Line2D at 0x7b04d0030670>], 'caps':
[<matplotlib.lines.Line2D at 0x7b04cdf029e0>, <matplotlib.lines.Line2D
at 0x7b04cdf02c80>], 'boxes': [<matplotlib.lines.Line2D at
0x7b04cdf02560>], 'medians': [<matplotlib.lines.Line2D at
0x7b04cdf02f20>], 'fliers': [<matplotlib.lines.Line2D at
0x7b04cdf031c0>], 'means': []}
```



Interpretation: Interpreting quartile ranges allows for outlier detection and removal. By calculating the interquartile range (IQR) as the difference between the upper and lower quartiles, data points beyond 1.5 times the IQR from either quartile are identified as outliers and can be excluded or treated to ensure the robustness of the analysis. In the similar way the outliers in all other variables can be removed.

c) Summarize the critical variables in the data set region wise and district wise and indicate the top three districts and the bottom three districts of consumption. By summarizing the critical variables as total consumption we can estimate the top 3 and bottom 3 consuming districts. 8 Code and Result:

```
ORI_clean['District'].unique()
```

```
array([ 4, 29, 30,  3, 28,  2,  1, 27,  5, 26, 25, 24, 23, 22, 20, 16, 17,
       19, 15, 18, 14, 21, 13, 12,  7,  8, 11,  6, 10,  9])
```

```
ORI_clean.loc[:,'Sector'] = ORI_clean['Sector'].replace([1, 2],
['URBAN', 'RURAL'])
```

```
ORI_clean.columns
```

```
Index(['state_1', 'District', 'Sector', 'Region', 'State_Region',
       'ricetotal_q', 'wheattotal_q', 'moong_q', 'Milktotal_q', 'chicken_q',
       'bread_q', 'foodtotal_q', 'Beveragestotal_v', 'Meals_At_Home'],
      dtype='object')
```

```
ORI_clean.loc[:, 'total_consumption'] = ORI_clean[['ricetotal_q',
'wheattotal_q', 'moong_q', 'Milktotal_q', 'chicken_q', 'bread_q',
'foodtotal_q', 'Beveragestotal_v']].sum(axis=1)
```

```
<ipython-input-22-84c6657fabef>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  ORI_clean.loc[:, 'total_consumption'] = ORI_clean[['ricetotal_q',
'wheattotal_q', 'moong_q', 'Milktotal_q', 'chicken_q', 'bread_q',
'foodtotal_q', 'Beveragestotal_v']].sum(axis=1)
```

```
ORI_clean.head()
```

	state_1	District	Sector	Region	State_Region	ricetotal_q	wheattotal_q	moong_q	Milktotal_q	chicken_q	bread_q	foodtotal_q	Beveragestotal_v	Meals_At_Home	total_consumption
741	ORI	4	RURAL	3.0	213	8.500000	2.5	0.250000	0	0.000000	0.000	33.110413	10.008000	60.0	54.368413
742	ORI	4	RURAL	3.0	213	4.666667	5.0	0.333333	0	0.333333	0.200	31.683645	33.333333	60.0	75.550312
743	ORI	4	RURAL	3.0	213	4.500000	4.5	0.500000	0	0.000000	0.000	25.575244	50.000000	60.0	85.075244

	state_1	District	Sector	Region	State_Region	riceto tal_q	wheat total_q	moong_q	Milkt otal_q	chicken_q	bread_q	foodt otal_q	Bever agest otal_v	Meals_At_Home	total_consumption	
744	ORI	4	RURAL	3.0	213	13.000000	1.0	0.200000	0	0.200000	0.000	24.920166	6.000000	60.0	45.320166	
745	ORI	4	RURAL	3.0	213	12.500000	0.0	0.250000	0	0.000000	0.025	24.742780	7.500000	90.0	45.017780	

```
ORI_clean.groupby('Region').agg({'total_consumption':['std','mean','max','min']})
```

	total_consumption			
	std	mean	max	min
Region				
1.0	32.414973	62.180210	187.751465	24.98023
2.0	24.954397	49.355884	211.533867	0.00000
3.0	38.631152	58.955174	299.438431	0.00000

```
ORI_clean.groupby('District').agg({'total_consumption':['std','mean','max','min']})
```

	total_consumption			
	std	mean	max	min
District				
1	51.280868	57.415027	299.438431	21.540365
2	41.960684	59.031612	175.250420	0.000000
3	34.527237	62.380691	164.275465	25.250133
4	25.610555	57.201858	149.033748	30.150168
5	42.086408	62.120997	220.017100	0.000000
6	7.718583	53.410550	60.775245	43.325229

total_consumption				
	std	mean	max	min
District				
7	17.518657	21.592086	47.450250	0.000000
8	3.541844	51.322875	56.040408	45.867025
9	33.720246	61.566050	134.370770	31.875225
10	28.182754	61.607701	118.383833	31.800186
11	15.522049	53.383713	79.000597	37.000000
12	16.914891	52.327427	99.478696	24.980230
13	13.716476	65.201879	81.582334	38.186073
14	29.245448	69.828054	115.309683	37.300030
15	13.782658	65.406585	87.301300	43.500242
16	39.193320	79.661913	148.375605	44.100280
17	48.389512	69.546444	166.577167	29.162638
18	50.571888	67.490440	187.751465	31.425362
19	13.559877	44.205419	77.100260	37.441812
20	6.977840	50.383126	58.843975	37.150435
21	22.442370	55.097293	120.317673	30.509182
22	6.716927	44.849968	58.181376	33.626147
23	18.853561	49.862959	99.525362	22.882440
24	19.659886	48.303110	109.366877	20.000000
25	25.234735	57.527052	117.000390	25.816865
26	34.216146	56.553452	211.533867	29.700140
27	12.785232	42.259938	89.025444	24.250167
28	25.991377	56.331454	137.796162	22.550402
29	29.907043	36.516684	108.013260	0.000000
30	32.444519	47.905478	155.925806	0.000000

```
total_consumption_by_districtcode=ORI_clean.groupby('District')['total_consumption'].sum()
```

```
total_consumption_by_districtcode.sort_values(ascending=False).head(3)
```

```
District
5    3975.743824
3    1933.801424
2    1889.011584
Name: total_consumption, dtype: float64
```

```
ORI_clean.loc[:,"District"] = ORI_clean.loc[:,"District"].replace({5: "Hyderabad and Rangar", 6: "Rangareddi", 23: "Chittoor"})

total_consumption_by_districtname=ORI_clean.groupby('District')['total_consumption'].sum()

total_consumption_by_districtname.sort_values(ascending=False).head(3)
```

```
District
Hyderabad and Rangar    3975.743824
3                       1933.801424
2                       1889.011584
Name: total_consumption, dtype: float64
```

```
from statsmodels.stats import weightstats as stests




rural=ORI_clean[ORI_clean['Sector']=="RURAL"]
urban=ORI_clean[ORI_clean['Sector']=="URBAN"]

rural.head()
```

state_1	District	Sector	Region	State_Region	ricetotal_q	wheattotal_q	moong_q	Milktotal_q	chicken_q	bread_q	foodtotal_q	Beveragestotal_v	Meals_At_Home	total_consumption	
741	ORI	4	RURAL	3.0	213	8.500000	2.5	0.250000	0	0.000000	0.000	33.111043	10.008000	60.0	54.368413
742	ORI	4	RURAL	3.0	213	4.666667	5.0	0.333333	0	0.333333	0.200	31.683645	33.333333	60.0	75.550312

state_1	District	Sector	Region	State_Region	riceto tal_q	wheat total_q	moong_q	Milkt otal_q	chicken_q	bread_q	foodt otal_q	Bevera gestot al_v	Meals_At_Home	total_consumption	
743	ORI	4	RURAL	3.0	213	4.500000	4.5	0.500000	0	0.000000	0.000	25.575244	50.000000	60.0	85.075244
744	ORI	4	RURAL	3.0	213	13.000000	1.0	0.200000	0	0.200000	0.000	24.920166	6.000000	60.0	45.320166
745	ORI	4	RURAL	3.0	213	12.500000	0.0	0.250000	0	0.000000	0.025	24.742780	7.500000	90.0	45.017780

```
urban.head()
```

state_1	District	Sector	Region	State_Region	riceto tal_q	wheatto tal_q	moong_q	Milkto tal_q	chicken_q	bread_q	foodto tal_q	Beverages total_v	Meals_At_Home	total_consumption

```
cons_rural=rural['total_consumption']
cons_urban=urban['total_consumption']

pip install statsmodels
```

```
Requirement already satisfied: statsmodels in
/usr/local/lib/python3.10/dist-packages (0.14.2)
Requirement already satisfied: numpy>=1.22.3 in
/usr/local/lib/python3.10/dist-packages (from statsmodels) (1.25.2)
Requirement already satisfied: scipy!=1.9.2,>=1.8 in
/usr/local/lib/python3.10/dist-packages (from statsmodels) (1.11.4)
Requirement already satisfied: pandas!=2.1.0,>=1.4 in
/usr/local/lib/python3.10/dist-packages (from statsmodels) (2.0.3)
Requirement already satisfied: patsy>=0.5.6 in
/usr/local/lib/python3.10/dist-packages (from statsmodels) (0.5.6)
Requirement already satisfied: packaging>=21.3 in
/usr/local/lib/python3.10/dist-packages (from statsmodels) (24.1)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas!=2.1.0,>=1.4-
>statsmodels) (2.8.2)
```

```
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas!=2.1.0,>=1.4-
>statsmodels) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in
/usr/local/lib/python3.10/dist-packages (from pandas!=2.1.0,>=1.4-
>statsmodels) (2024.1)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-
packages (from patsy>=0.5.6->statsmodels) (1.16.0)
```

e) Test whether the differences in the means are significant or not.

```
import numpy as np
from statsmodels.stats.weightstats import ztest

if cons_rural.empty or cons_urban.empty:
    print("Warning: One or both of the consumption Series are empty. Z-
test cannot be performed.")
else:
    z_statistic, p_value = stests.ztest(cons_rural, cons_urban)
    # Print the z-score and p-value
    print("Z-Score:", z_statistic)
    print("P-Value:", p_value)
```

```
z_statistic, p_value = stests.ztest(cons_rural, cons_urban)
# Print the z-score and p-value
print("Z-Score:", z_statistic)
print("P-Value:", p_value)
```

```
Z-Score: 12.52569222339867
P-Value: 5.4017986377956026e-36
```