# Codes

## Python Code (Exported from IPYNB)

```
# %% [markdown]
# # **Imports**

# %%
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.stats import weightstats as stats

# %% [markdown]
# # **Set Paths**

# %%
import os
BASE = os.getcwd()
DATA = os.path.join(BASE, 'datasets')
print(BASE, DATA)

# %% [markdown]
# # **Load Main Dataset**

# %%
data_file_name = 'NSSO68.csv'
data_file = os.path.join(DATA, data_file_name)

df = pd.read_csv(data_file, encoding='latin1', low_memory=False, index_col='slno')

# %%
df.head()

# %% [markdown]
# # **Drop Unnecessary Attributes**

# %%
# Keeping only the relevant columns
relevant_columns = [
    'state',
    'state_1',
    'District',
    'Region',
    'Sector',
    'State_Region',
    'Meals_At_Home',
    'ricetotal_v',
    'wheattotal_v',
    'Milktotal_v',
    'pulsestot_v',
    'nonvegtotal_v',
    'fruitstt_v',
    'No_of_Meals_per_day'
]

df.drop(columns=[col for col in df.columns if col not in relevant_columns],
inplace=True)

# %% [markdown]
# # **Filter for assigned State**

# %%
# MEGHALAYA State Code: 17
df_meghalaya = df[df['state'] == 17]
del df

# %%
df_meghalaya.head()

# %% [markdown]
# # **Save Filtered Dataset**

# %%
df_meghalaya.reset_index(inplace=True)
df_meghalaya.drop(columns=['slno'], inplace=True)
df_meghalaya.index.name = 'slno'

my_data_file_name = 'meghalaya_NSSO68.csv'
df_meghalaya.to_csv(os.path.join(DATA, my_data_file_name))

# %%
df_meghalaya.info()

# %% [markdown]
# # **Impute Null Values**

# %%
for column in df_meghalaya.columns:
    null_c = df_meghalaya[column].isnull().sum()
    if null_c > 0:
        df_meghalaya[column] =
df_meghalaya[column].fillna(df_meghalaya[column].mean())
        print(f"{column}: {null_c} null values")

# %%
df_meghalaya.info()

# %% [markdown]
# # **Handle Outliers in Consumption Columns**

# %%
consumption_columns = [
    'ricetotal_v',
    'wheattotal_v',
    'Milktotal_v',
    'pulsestot_v',
    'nonvegtotal_v',
    'fruitstt_v'
]

# %%
def check_outliers():
    # Create subplots to show boxplots for each consumption column
    fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(15, 10))

    for ax, column in zip(axes.flatten(), consumption_columns):
        df_meghalaya.boxplot(column=column, ax=ax)
        ax.set_title(f'Boxplot of {column}')
        ax.set_ylabel('Consumption (kg/month)')
        ax.set_xlabel('')

    plt.tight_layout()

# %%
def remove_outliers(df, column):
    q1 = df[column].quantile(0.25)
    q3 = df[column].quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

# %%
check_outliers()

# %%
for column in consumption_columns:
    df_meghalaya = remove_outliers(df_meghalaya, column)

# %%
check_outliers()

# %% [markdown]
# # **Create Total Consumption Column**

# %%
df_meghalaya['total_consumption'] = df_meghalaya[consumption_columns].sum(axis=1)

# %% [markdown]
# # **Descriptive Statistics for Total Consumption by District, Region and Sector**

# %%
def summarize_consumption(df, column):
    summary = df.groupby(column)['total_consumption'].agg(['mean', 'std', 'count', 'sum'])
    summary.sort_values(by='sum', ascending=False, inplace=True)
    return summary

# %%
district = summarize_consumption(df_meghalaya, 'District')
region = summarize_consumption(df_meghalaya, 'Region')
sector = summarize_consumption(df_meghalaya, 'Sector')

# %%
district.head()

# %%
district.tail()

# %% [markdown]
# # **Map District and Sector to their Names/Labels**

# %%
dist_st_map = {
    1: 'West Garo Hills',
    2: 'East Garo Hills',
    3: 'South Garo Hills',
    4: 'West Khasi Hills',
    5: 'Ri Bhoi',
    6: 'East Khasi Hills',
    7: 'Jaintia Hills',
}

sector_map = {
    1: 'Rural',
    2: 'Urban'
}

df_meghalaya['District'] = df_meghalaya['District'].map(dist_st_map)
df_meghalaya['Sector'] = df_meghalaya['Sector'].map(sector_map)

# %%
district = summarize_consumption(df_meghalaya, 'District')
region = summarize_consumption(df_meghalaya, 'Region')
sector = summarize_consumption(df_meghalaya, 'Sector')

# %%
district

# %% [markdown]
# # **Hypothesis Testing using Z-Test as n > 30**

# %%
urban = df_meghalaya[df_meghalaya['Sector'] == 'Urban']['total_consumption']
rural = df_meghalaya[df_meghalaya['Sector'] == 'Rural']['total_consumption']

sector_z_stat, sector_p_value = stats.ztest(urban, rural, alternative='two-sided')

# %%
print(sector_z_stat)
print(sector_p_value)

# %%
if sector_p_value < 0.05:
    print("There is a significant difference in total consumption between Urban and Rural sectors. Reject the null hypothesis.")
else:
    print("There is no significant difference in total consumption between Urban and Rural sectors. Failed to reject the null hypothesis.")

# %%
top_dist = df_meghalaya[df_meghalaya['District'] == district.iloc[0].name]['total_consumption']
bottom_dist = df_meghalaya[df_meghalaya['District'] == district.iloc[-1].name]['total_consumption']

dist_z_stat, dist_p_value = stats.ztest(top_dist, bottom_dist, alternative='two-sided')

# %%
print(dist_z_stat)
print(dist_p_value)

# %%
if dist_p_value < 0.05:
    print("There is a significant difference in total consumption between the top and bottom districts. Reject the null hypothesis.")
else:
    print("There is no significant difference in total consumption between the top and bottom districts. Failed to reject the null hypothesis.")
```

# R Code

```
# --- Project Setup and Package Management ---

# Set the base directory for the project
BASE <- "C:\\Users\\ujjwa\\Documents\\VCU\\Pre-Course\\SCMA632\\Assignments\\A1\\R"
setwd(BASE) # Set the working directory to the specified base path
getwd() # Verify the current working directory

# Define a function to install packages if they are not already installed
install <- function(pkg) {
  if (!require(pkg, character.only = T)) { # Check if the package is loaded
    # Install with dependencies if not loaded
    install.packages(pkg, dependencies = T, quiet = T, verbose = F)
  }
}

# Define a function to load packages
```

```r
load <- function(pkg) {
  library(pkg, character.only = T, quietly = T, verbose = F) # Load the specified package
}

# List of required packages for data manipulation, visualization, and statistical tests
pkgs <- c("dplyr", "readr", "readxl", "tidyr", "ggplot2", "BSDA")
lapply(pkgs, install) # Apply the install function to all packages
lapply(pkgs, load) # Apply the load function to all packages

# --- Data Loading and Initial Filtering ---

# Load the main dataset from a CSV file
data <- read.csv('./datasets/NSSO68.csv')
# Filter the data to include only records for Meghalaya state (state code "17")
data_meghalaya <- data %>% filter(state == "17")

# Remove the original large dataset to free up memory
rm(data)

# Print the column names of the filtered Meghalaya dataset
print(names(data_meghalaya))
# Print the first few rows of the Meghalaya dataset for a quick preview
print(head(data_meghalaya))
# Print the dimensions (number of rows and columns) of the Meghalaya dataset
print(dim(data_meghalaya))

# Save the filtered Meghalaya data to a new CSV file for future use
write.csv(data_meghalaya, './datasets/meghalaya_NSSO68.csv', row.names = FALSE) # Prevent writing row names as they are not part of the data

# Select a subset of relevant columns for focused analysis
data_meghalaya <- data_meghalaya %>% select(
  state,
  state_1,
  District,
  Region,
  Sector,
  State_Region,
  Meals_At_Home,
  ricetotal_v,
  wheattotal_v,
  Milktotal_v,
  pulsestot_v,
  nonvegtotal_v,
  fruitstt_v,
  No_of_Meals_per_day
)

# --- Handling Missing Values (NA) ---

# Check which columns (if any) still have missing values (NA) before imputation
print(colSums(is.na(data_meghalaya)) > 0)
```

```
# Define a function to impute missing values with the mean of the column
impute <- function(col) {
  if(any(is.na(col))) { # Check if there are any NAs in the column
    col[is.na(col)] <- mean(col, na.rm = T) # Replace NAs with the mean, ignoring NAs in mean calculation
  }
  return(col) # Return the column with imputed values
}

# Apply the imputation function to specific columns identified to have NAs
data_meghalaya$Meals_At_Home <- impute(data_meghalaya$Meals_At_Home)
data_meghalaya$No_of_Meals_per_day <- impute(data_meghalaya$No_of_Meals_per_day)

# Re-check for missing values after imputation to confirm removal
print(colSums(is.na(data_meghalaya)) > 0)

# --- Outlier Removal ---

# Define a function to remove outliers using the Interquartile Range (IQR) method
remove_outliers <- function(df, col) {
  q1 <- quantile(df[, col], 0.25) # Calculate the first quartile (25th percentile)
  q3 <- quantile(df[, col], 0.75) # Calculate the third quartile (75th percentile)
  iqr <- q3 - q1 # Calculate the Interquartile Range
  lower_threshold <- q1 - (1.5 * iqr) # Define the lower bound for outlier detection
  upper_threshold <- q3 + (1.5 * iqr) # Define the upper bound for outlier detection
  # Subset the dataframe to include only rows where the column's value is within the defined thresholds
  df <- subset(df, df[, col] >= lower_threshold & df[, col] <= upper_threshold)
  return(df) # Return the dataframe with outliers removed from the specified column
}

# Columns to check for outliers (various consumption variables)
outlier_check_cols <- c("ricetotal_v", "wheattotal_v", "Milktotal_v", "pulsestot_v", "nonvegtotal_v", "fruitstt_v")

# Define a function to plot boxplots for outlier visualization
plot_outliers <- function(cols) {
  for (outlier_check_col in cols) {
    boxplot(data_meghalaya[, outlier_check_col], main = outlier_check_col) # Create a boxplot for each column
  }
}

# Plot initial boxplots to visualize outliers before removal
plot_outliers(outlier_check_cols)

# Columns to apply outlier removal to (a subset of the check columns)
outlier_cols <- c("ricetotal_v", "wheattotal_v", "Milktotal_v", "pulsestot_v")

# Loop through specified columns and apply the outlier removal function
```

```
for (outlier_col in outlier_cols) {
  data_meghalaya <- remove_outliers(data_meghalaya, outlier_col)
}

# Plot boxplots again to visualize the effect of outlier removal
plot_outliers(outlier_check_cols)

# --- Total Consumption Calculation and Summaries ---

# Define columns representing various consumption categories to be summed
consumption_cols <- c("ricetotal_v", "wheattotal_v", "Milktotal_v", "pulsestot_v", "nonvegtotal_v", "fruitstt_v")

# Calculate the total consumption for each row by summing up the specified consumption columns
data_meghalaya$total_consumption_v <- rowSums(data_meghalaya[, consumption_cols], na.rm = T)

# Define a function to summarize total consumption by a given grouping column
summarize_consumption <- function(col) {
  summary <- data_meghalaya %>%
    group_by(across(all_of(col))) %>% # Group data by the specified column
    summarize(total = sum(total_consumption_v)) %>% # Calculate the sum of total consumption for each group
    arrange(desc(total)) # Arrange the summary table in descending order of total consumption
  return(summary) # Return the summary table
}

# Summarize consumption by District, Region, and Sector
district_summary <- summarize_consumption("District")
region_summary <- summarize_consumption("Region")
sector_summary <- summarize_consumption("Sector")

# Print the top 3 and bottom 3 consuming districts based on the summary
cat("Top 3 Consuming Districts:\n")
print(head(district_summary, 3))
cat("Bottom 3 Consuming Districts:\n")
print(tail(district_summary, 3))

# Print the consumption summary for regions and sectors
cat("Region Consumption Summary:\n")
print(region_summary)
cat("Sector Consumption Summary:\n")
print(sector_summary)

# --- Data Transformation (Mapping Codes to Names) ---

# Create a mapping from numeric District codes to their full names
district_map <- c(
  "1" = 'West Garo Hills',
  "2" = 'East Garo Hills',
  "3" = 'South Garo Hills',
  "4" = 'West Khasi Hills',
  "5" = 'Ri Bhoi',
  "6" = 'East Khasi Hills',
  "7" = 'Jaintia Hills'
)

# Convert District column to character type before applying the map for
consistency
data_meghalaya$District <- as.character(data_meghalaya$District)
# Replace numeric District codes with their corresponding names using the map
data_meghalaya$District <- ifelse(data_meghalaya$District %in%
names(district_map), district_map[data_meghalaya$District],
data_meghalaya$District)

# Create a mapping from numeric Sector codes to descriptive names
sector_map <- c(
  "1" = "Rural",
  "2" = "Urban"
)

# Convert Sector column to character type before applying the map
data_meghalaya$Sector <- as.character(data_meghalaya$Sector)
# Replace numeric Sector codes with "Rural" or "Urban"
data_meghalaya$Sector <- ifelse(data_meghalaya$Sector %in% names(sector_map),
sector_map[data_meghalaya$Sector], data_meghalaya$Sector)

# Re-summarize consumption by District, Region, and Sector after mapping names
# This ensures summaries use the descriptive names rather than numeric codes
district_summary <- summarize_consumption("District")
region_summary <- summarize_consumption("Region")
sector_summary <- summarize_consumption("Sector")

# --- Z-test for Rural vs. Urban Consumption ---

# Filter total consumption data for rural areas
rural <- data_meghalaya %>%
  filter(Sector == "Rural") %>%
  select(total_consumption_v)

# Filter total consumption data for urban areas
urban <- data_meghalaya %>%
  filter(Sector == "Urban") %>%
  select(total_consumption_v)

# Calculate the mean total consumption for rural and urban areas, ignoring NAs
mean_rural <- mean(rural$total_consumption_v, na.rm = T)
mean_urban <- mean(urban$total_consumption_v, na.rm = T)

# Calculate the standard deviation for rural and urban consumption, ignoring NAs
sd_rural <- sd(rural$total_consumption_v, na.rm = T)
sd_urban <- sd(urban$total_consumption_v, na.rm = T)

# Perform a two-sample Z-test to compare mean consumptions between rural and urban
areas
# Using calculated sample standard deviations as estimates for population standard
```

```r
deviations
z_test_result_sector <- z.test(
  x = rural$total_consumption_v, # Data for the first group (rural)
  y = urban$total_consumption_v, # Data for the second group (urban)
  alternative = "two.sided", # Test for a difference in either direction
  mu = 0, # Null hypothesis: difference in means is 0
  sigma.x = sd_rural, # Standard deviation of the rural group
  sigma.y = sd_urban, # Standard deviation of the urban group
  conf.level = 0.95 # 95% confidence level
)

# Interpret the results of the Z-test for rural vs. urban consumption
cat("\n--- Z-test Results: Rural vs. Urban Consumption ---\n")
if (z_test_result_sector$p.value < 0.05) {
  cat("P value is < 0.05 (", round(z_test_result_sector$p.value, 5), "). Therefore, we reject the null hypothesis.\n")
  cat("There is a significant difference between mean consumptions of urban and rural areas.\n")
  cat("The mean consumption in Rural areas is ", round(mean_rural, 2), " and in Urban areas it's ", round(mean_urban, 2), ".\n")
} else {
  cat("P value is >= 0.05 (", round(z_test_result_sector$p.value, 5), "). Therefore, we fail to reject the null hypothesis.\n")
  cat("There is no significant difference between mean consumptions of urban and rural areas.\n")
  cat("The mean consumption in Rural areas is ", round(mean_rural, 2), " and in Urban areas it's ", round(mean_urban, 2), ".\n")
}

# --- Z-test for Top vs. Bottom Consuming Districts ---

# Get the name of the top consuming district from the summarized data
top_district_name <- head(district_summary, 1)$District
# Get the name of the bottom consuming district from the summarized data
bottom_district_name <- tail(district_summary, 1)$District

# Filter total consumption data for the top consuming district
top_district_data <- data_meghalaya %>%
  filter(District == top_district_name) %>%
  select(total_consumption_v)

# Filter total consumption data for the bottom consuming district
bottom_district_data <- data_meghalaya %>%
  filter(District == bottom_district_name) %>%
  select(total_consumption_v)

# Calculate the mean total consumption for the top and bottom districts, ignoring NAs
mean_top_district <- mean(top_district_data$total_consumption_v, na.rm = TRUE)
mean_bottom_district <- mean(bottom_district_data$total_consumption_v, na.rm = TRUE)

# Calculate the standard deviation for the top and bottom districts' consumption, ignoring NAs
```

```r
sd_top_district <- sd(top_district_data$total_consumption_v, na.rm = TRUE)
sd_bottom_district <- sd(bottom_district_data$total_consumption_v, na.rm = TRUE)

# Perform a two-sample Z-test to compare mean consumptions between the top and bottom districts
# Using calculated sample standard deviations as estimates for population standard deviations
z_test_result_district <- z.test(
  x = top_district_data$total_consumption_v, # Data for the top district
  y = bottom_district_data$total_consumption_v, # Data for the bottom district
  alternative = "two.sided", # Test for a difference in either direction
  mu = 0, # Null hypothesis: difference in means is 0
  sigma.x = sd_top_district, # Standard deviation for the top district
  sigma.y = sd_bottom_district, # Standard deviation for the bottom district
  conf.level = 0.95 # 95% confidence level
)

# Interpret the results of the Z-test for top vs. bottom districts
cat("\n--- Z-test Results: Top Consuming District (", top_district_name, ") vs. Bottom Consuming District (", bottom_district_name, ") ---\n")
if (z_test_result_district$p.value < 0.05) {
  cat("P value is < 0.05 (", round(z_test_result_district$p.value, 5), ").\n")
  cat("Therefore, we reject the null hypothesis.\n")
  cat("There is a significant difference between mean consumptions of ", top_district_name, " and ", bottom_district_name, ".\n")
  cat("The mean consumption in ", top_district_name, " is ", round(mean_top_district, 2), " and in ", bottom_district_name, " it's ", round(mean_bottom_district, 2), ".\n")
} else {
  cat("P value is >= 0.05 (", round(z_test_result_district$p.value, 5), ").\n")
  cat("Therefore, we fail to reject the null hypothesis.\n")
  cat("There is no significant difference between mean consumptions of ", top_district_name, " and ", bottom_district_name, ".\n")
  cat("The mean consumption in ", top_district_name, " is ", round(mean_top_district, 2), " and in ", bottom_district_name, " it's ", round(mean_bottom_district, 2), ".\n")
}
```