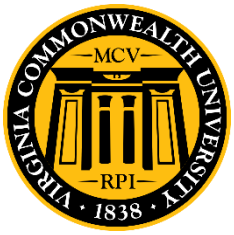


VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A4: Multivariate Analysis and Business Analytics Applications

JESIN KANDATHY JOY

V01110163

Date of Submission: 08-07-2024

CONTENTS

Introduction:

This report presents a detailed analysis using various statistical techniques applied to distinct datasets. The analysis includes Principal Component Analysis (PCA) and Factor Analysis using the "Survey.csv" dataset to uncover underlying data dimensions. Additionally, Cluster Analysis is conducted on the same dataset to profile respondents based on background variables. Multidimensional Scaling is performed using the "icecream.csv" dataset to interpret product similarities and differences. Lastly, Conjoint Analysis is applied to "pizza_data.csv" to explore consumer preferences and decision-making processes. The insights derived from these analyses provide valuable implications for strategic decision-making in relevant domains.

Objectives:

1. **To Perform Principal Component Analysis and Factor Analysis:** Explore the underlying structure within the "Survey.csv" dataset to identify key dimensions and factors influencing responses.
2. **To Conduct Cluster Analysis:** Segment respondents based on demographic and behavioral characteristics using the "Survey.csv" dataset to understand distinct groups within the sample.
3. **To Apply Multidimensional Scaling:** Analyze similarities and dissimilarities among ice cream products in the "icecream.csv" dataset to visualize their perceptual positions in a low-dimensional space.
4. **To Conduct Conjoint Analysis:** Investigate consumer preferences and trade-offs in the context of pizza attributes using the "pizza_data.csv" dataset to derive insights into product positioning and market strategy.

Business Significance:

1. **Principal Component Analysis and Factor Analysis:** Identifying key factors in the "Survey.csv" dataset helps businesses pinpoint drivers of customer preferences. This insight enhances marketing strategies and segmentation efforts, improving customer satisfaction and resource allocation.
2. **Cluster Analysis:** Segmenting respondents in the "Survey.csv" dataset allows businesses to tailor marketing and service strategies to specific customer groups. This targeted approach boosts engagement, conversion rates, and overall marketing effectiveness.
3. **Multidimensional Scaling:** Visualizing product perceptions in the "icecream.csv" dataset informs product positioning and innovation strategies. Businesses can optimize product portfolios and meet consumer preferences more effectively.
4. **Conjoint Analysis:** Analyzing consumer preferences in the "pizza_data.csv" dataset guides pricing strategies and product feature prioritization. This understanding helps businesses launch products that better align with market demands, enhancing competitiveness and profitability.

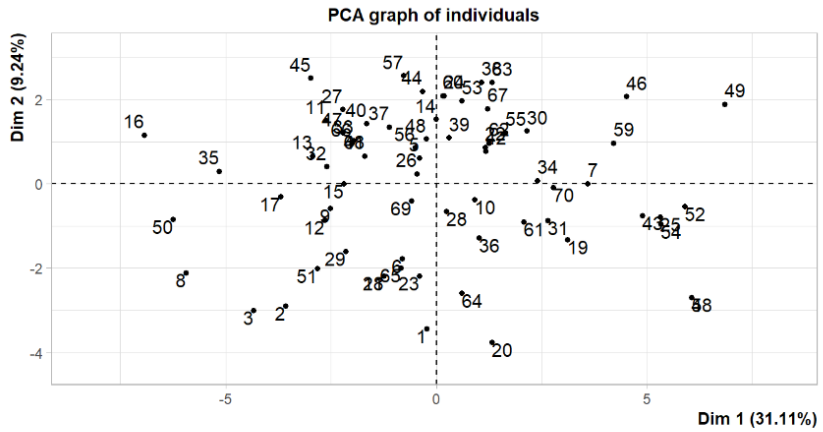
R code and results:

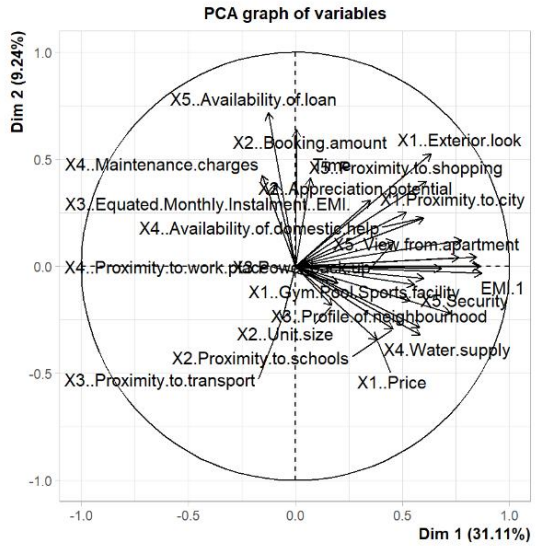
Q1:

```r
# Performing PCA using GPArotation
library(GPArotation)
pca_1 <- principal(sur_int, 5, n.obs = 70, rotate = "promax")
print(pca_1)
```

```
## Principal Components Analysis
## Call: principal(r = sur_int, nfactors = 5, rotate = "promax", n.obs = 70)
## Standardized loadings (pattern matrix) based upon correlation matrix
##                                              RC1   RC5   RC3   RC2   RC4   h2
## X1.Proximity.to.city                       -0.02  0.36  0.62  0.22 -0.33 0.71
## X2.Proximity.to.schools                    -0.06  0.26  0.49 -0.12  0.16 0.44
## X3..Proximity.to.transport                 -0.08  0.02 -0.22  0.16  0.76 0.55
## X4..Proximity.to.work.place                -0.34 -0.01  0.95  0.12 -0.02 0.71
## X5..Proximity.to.shopping                   0.76 -0.12  0.12  0.23 -0.06 0.65
## X1..Gym.Pool.Sports.facility                0.47 -0.06  0.22 -0.13  0.23 0.45
## X2..Parking.space                           0.56  0.00  0.16 -0.17 -0.01 0.46
## X3.Power.back.up                            0.41 -0.27  0.57  0.09  0.00 0.58
## X4.Water.supply                             0.27  0.26  0.06  0.03  0.71 0.76
## X5.Security                                 0.84 -0.16 -0.24 -0.15  0.32 0.69
## X1..Exterior.look                           0.72  0.21 -0.16  0.21 -0.35 0.79
## X2..Unit.size                              -0.16  0.61 -0.28 -0.20 -0.10 0.39
## X3..Interior.design.and.branded.components  0.55  0.28  0.13 -0.07 -0.06 0.61
## X4..Layout.plan..Integrated.etc..           0.25  0.43  0.28 -0.06 -0.14 0.55
## X5..View.from.apartment                     0.80  0.21 -0.16 -0.08 -0.04 0.71
```

```r
# Performing PCA using FactoMineR
library(FactoMineR)
pca_2 <- PCA(sur_int, scale.unit = TRUE)
```

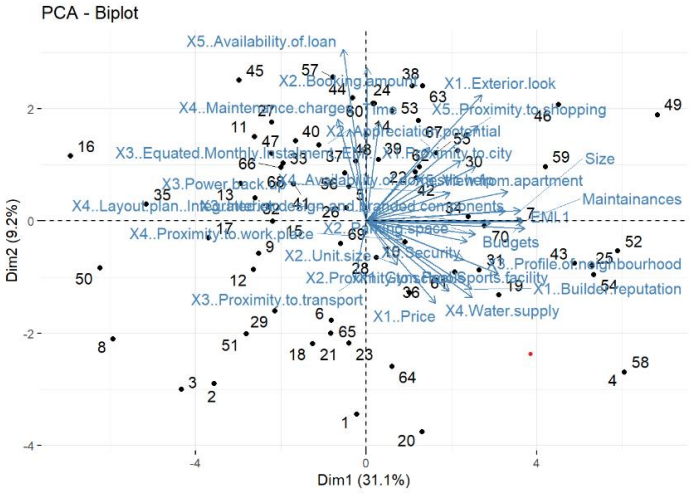





```
summary(pca_2)
```

```
##
## Call:
## PCA(X = sur_int, scale.unit = TRUE)
##
##
## Eigenvalues
##                        Dim.1   Dim.2   Dim.3   Dim.4   Dim.5   Dim.6   Dim.7
## Variance               9.022   2.678   2.095   1.858   1.683   1.370   1.326
## % of var.             31.110   9.236   7.225   6.406   5.804   4.723   4.574
## Cumulative % of var.  31.110  40.346  47.571  53.977  59.781  64.504  69.078
##                        Dim.8   Dim.9  Dim.10  Dim.11  Dim.12  Dim.13  Dim.14
## Variance               1.201   1.023   0.808   0.758   0.697   0.596   0.550
## % of var.              4.140   3.526   2.786   2.614   2.404   2.055   1.897
## Cumulative % of var.  73.218  76.745  79.531  82.145  84.549  86.604  88.501
##                       Dim.15  Dim.16  Dim.17  Dim.18  Dim.19  Dim.20  Dim.21
## Variance               0.524   0.459   0.384   0.345   0.298   0.250   0.220
## % of var.              1.805   1.583   1.326   1.188   1.029   0.863   0.757
## Cumulative % of var.  90.306  91.889  93.215  94.403  95.432  96.295  97.052
##                       Dim.22  Dim.23  Dim.24  Dim.25  Dim.26  Dim.27  Dim.28
## Variance               0.201   0.194   0.142   0.112   0.082   0.054   0.047
## % of var.              0.692   0.669   0.488   0.385   0.281   0.186   0.162
## Cumulative % of var.  97.743  98.413  98.901  99.286  99.567  99.753  99.915
```

```
# Using factoextra to plot the PCA biplot
library(factoextra)
fviz_pca_biplot(pca_2, repel = TRUE)
```



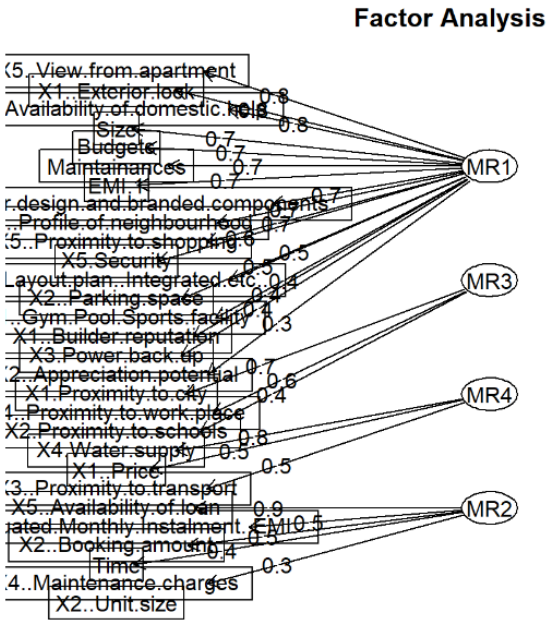
```r
# Factor Analysis
factor_analysis<-fa(sur_int,nfactors = 4,rotate = "varimax")
names(factor_analysis)
```

```
##  [1] "residual"      "dof"           "chi"           "nh"
##  [5] "rms"           "EPVAL"         "crms"          "EBIC"
##  [9] "ESABIC"        "fit"           "fit.off"       "sd"
## [13] "factors"       "complexity"    "n.obs"         "objective"
## [17] "criteria"      "STATISTIC"     "PVAL"          "Call"
## [21] "null.model"    "null.dof"      "null.chisq"    "TLI"
## [25] "CFI"           "RMSEA"         "BIC"           "SABIC"
## [29] "r.scores"      "R2"            "valid"         "score.cor"
## [33] "weights"       "rotation"      "hyperplane"    "communality"
## [37] "communalities" "uniquenesses"  "values"        "e.values"
## [41] "loadings"      "model"         "fm"            "rot.mat"
## [45] "Structure"     "method"        "scores"        "R2.scores"
## [49] "r"             "np.obs"        "fn"            "Vaccounted"
## [53] "ECV"
```

```r
print(factor_analysis$loadings,reorder=TRUE)
```

```
##
## Loadings:
##                                              MR1    MR3    MR4    MR2
## X1.Proximity.to.city                          0.338  0.719 -0.170
## X2.Proximity.to.schools                       0.222  0.406  0.258 -0.186
## X3..Proximity.to.transport                   -0.121 -0.225  0.501
## X4..Proximity.to.work.place                          0.615
## X5..Proximity.to.shopping                     0.652  0.113         0.251
## X1..Gym.Pool.Sports.facility                  0.426  0.149  0.223 -0.139
## X2..Parking.space                             0.529  0.184        -0.161
## X3.Power.back.up                              0.356  0.303
## X4.Water.supply                               0.382         0.752
## X5.Security                                   0.594 -0.195  0.275
## X1..Exterior.look                             0.783        -0.268  0.285
## X2..Unit.size                                 0.129                -0.132
## X3..Interior.design.and.branded.components    0.691  0.282
## X4..Layout.plan..Integrated.etc..             0.539  0.429        -0.125
## X5..View.from.apartment                       0.828
## X1..Price                                     0.154  0.201  0.532
## X2..Booking.amount                            0.108        -0.140  0.519
```

```r
fa.diagram(factor_analysis)
```



```
print(factor_analysis$communality)
```

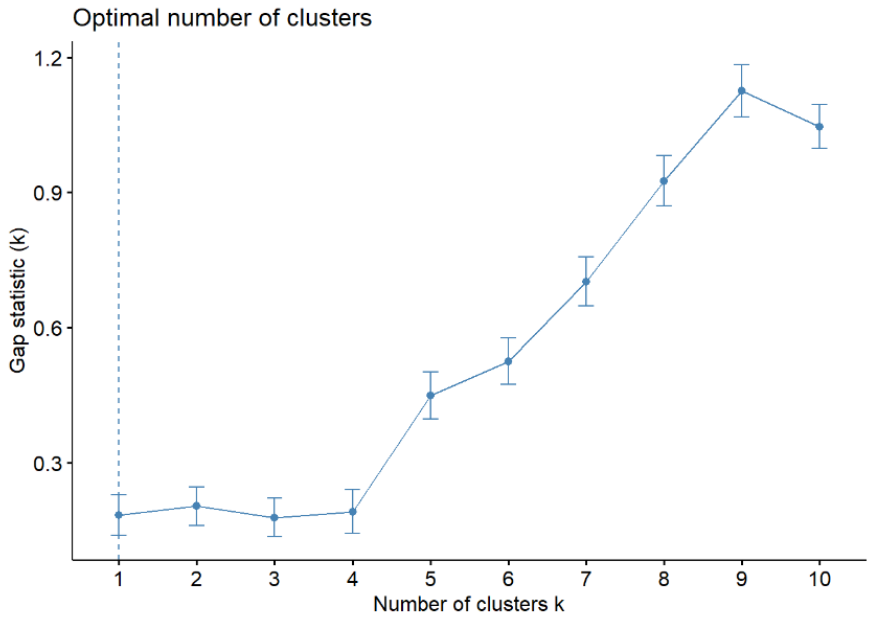
```
##                      X1.Proximity.to.city
##                                 0.66641636
##                    X2.Proximity.to.schools
##                                 0.31505092
##                 X3..Proximity.to.transport
##                                 0.32255808
##                X4..Proximity.to.work.place
##                                 0.38704235
##                  X5..Proximity.to.shopping
##                                 0.50168188
##               X1..Gym.Pool.Sports.facility
##                                 0.27313215
##                           X2..Parking.space
##                                 0.34504278
##                            X3.Power.back.up
```

```
print(factor_analysis$scores)
```

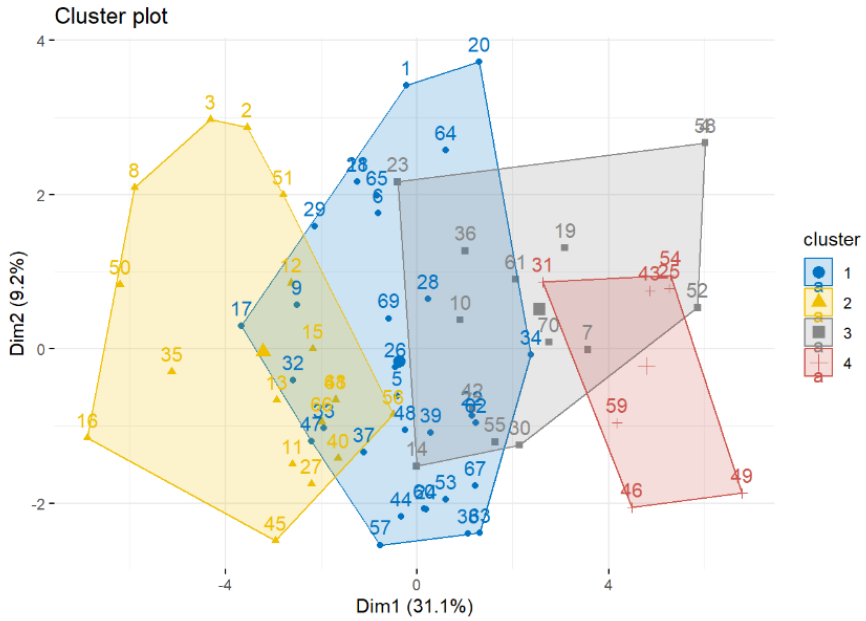
```
##              MR1         MR3          MR4         MR2
##  [1,] -0.5876607 -0.11251823  2.050119670 -0.80909066
##  [2,] -1.4792705 -0.36503965  0.775643390 -1.16305313
##  [3,] -0.5478608 -3.35305405  0.945557247 -1.32923887
##  [4,]  1.8807274  0.13402370  0.266228876 -2.06159338
##  [5,]  0.3295689 -0.18023376 -1.170876218 -0.36010367
##  [6,] -0.2866020 -0.47379445  0.367382156 -0.21010883
##  [7,]  0.5003641  0.65274995  1.058454876  0.43822528
##  [8,] -1.9088343 -0.71638354  0.125781819 -1.29105206
##  [9,] -0.7069313  0.10856075 -0.040407655  0.37696557
## [10,]  0.1573651  0.97898633 -0.458207894  0.32151704
## [11,] -0.3560346 -0.87215158 -0.548893781  0.20034247
```

Q2:

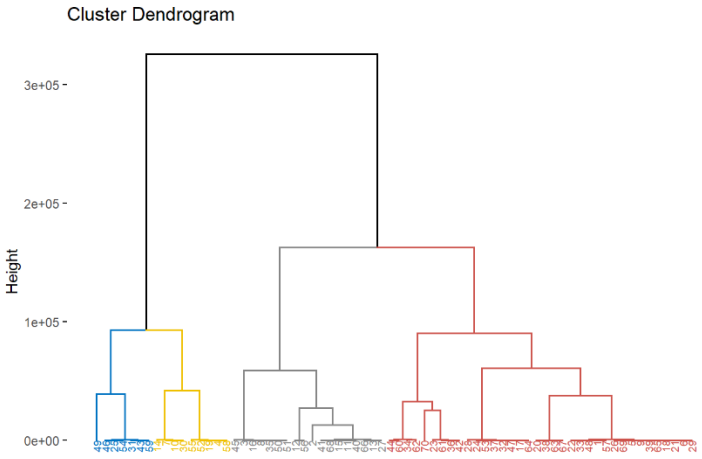
```
# Determining Optimal Number of Clusters with Gap Statistic
fviz_nbclust(sur_int,kmeans,method = "gap_stat")
```



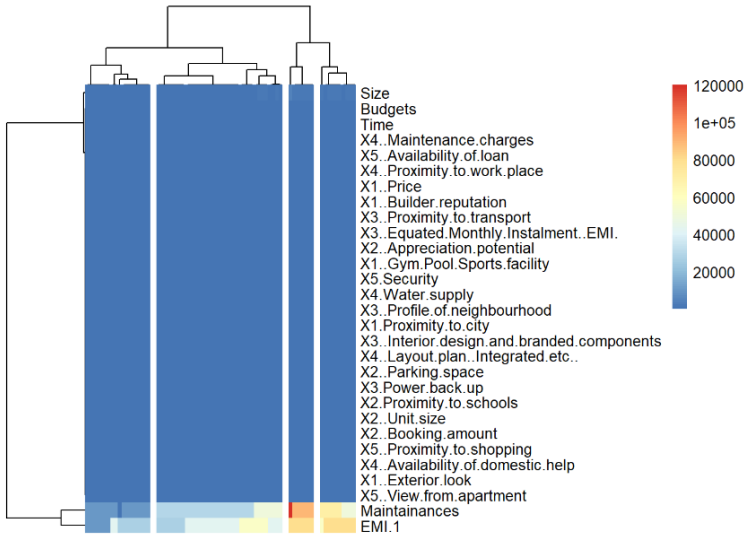
```
# Performing k-means Clustering
set.seed(123)
km.res<-kmeans(sur_int,4,nstart = 25)

# Visualizing k-means Clustering Results
fviz_cluster(km.res,data=sur_int,palette="jco", ggtheme = theme_minimal())
```




```
# Hierarchical Clustering (Dendrogram)
res.hc <- hclust(dist(sur_int), method = "ward.D2")
fviz_dend(res.hc,cex=0.5,k=4,palette = "jco")
```

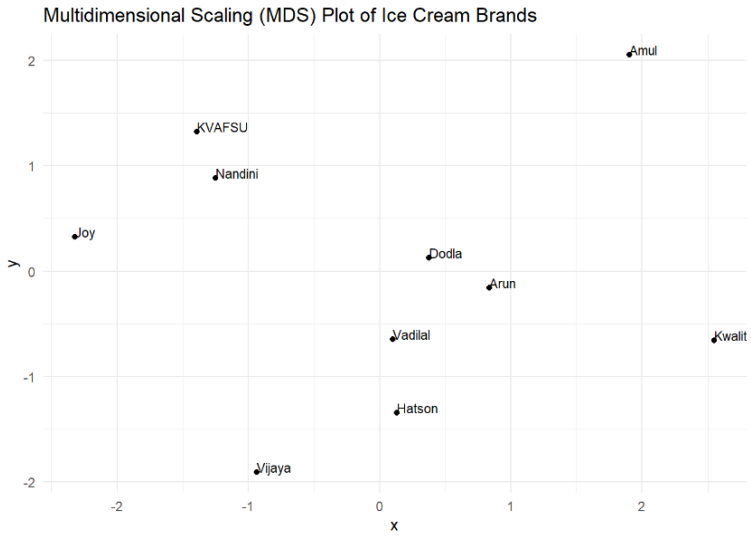


```r
#  Heatmap of Clustered Data
library(pheatmap)
pheatmap(t(sur_int),cutree_cols = 4)
```



Q3:

```r
# Extract the attributes for MDS (excluding the Brand column)
icecream_mds <- icecream[, -1]

# Perform Multidimensional Scaling (MDS)
mds <- cmdscale(dist(icecream_mds))

# Plot the MDS results
plot_data <- data.frame(
  x = mds[, 1],  # X-axis coordinates
  y = mds[, 2],  # Y-axis coordinates
  brand = icecream$Brand  # Brand names
)

ggplot(plot_data, aes(x, y, label = brand)) +
  geom_point() +
  geom_text(size = 3, hjust = 0, vjust = 0) +
  labs(title = "Multidimensional Scaling (MDS) Plot of Ice Cream Brands") +
  theme_minimal()
```



Q4:

```
# Load the dataset
df <- read.csv('E:\\JESIN\\DOCUMENTS\\scma\\A4\\pizza_data.csv')

# Convert categorical variables to factors
df$brand <- as.factor(df$brand)
df$price <- as.factor(df$price)
df$weight <- as.factor(df$weight)
df$crust <- as.factor(df$crust)
df$cheese <- as.factor(df$cheese)
df$size <- as.factor(df$size)
df$toppings <- as.factor(df$toppings)
df$spicy <- as.factor(df$spicy)

# Set sum contrasts for categorical variables
contrasts(df$brand) <- contr.sum(length(unique(df$brand)))
contrasts(df$price) <- contr.sum(length(unique(df$price)))
contrasts(df$weight) <- contr.sum(length(unique(df$weight)))
contrasts(df$crust) <- contr.sum(length(unique(df$crust)))
contrasts(df$cheese) <- contr.sum(length(unique(df$cheese)))
contrasts(df$size) <- contr.sum(length(unique(df$size)))
contrasts(df$toppings) <- contr.sum(length(unique(df$toppings)))
contrasts(df$spicy) <- contr.sum(length(unique(df$spicy)))

# Define the model formula
model <- as.formula("ranking ~ brand + price + weight + crust + cheese + size + toppings + spicy")

# Fit the OLS model
model_fit <- lm(model, data = df)

# Print the summary of the model
summary(model_fit)
```

```
## Residuals:
##      1      2      3      4      5      6      7      8      9     10     11
## -0.125  0.125  0.125 -0.125 -0.125  0.125 -0.125  0.125  0.125 -0.125 -0.125
##     12     13     14     15     16
## -0.125  0.125  0.125  0.125 -0.125
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.500e+00  1.250e-01  68.000  0.00936 **
## brand1      -6.910e-16  2.165e-01   0.000  1.00000
## brand2      -3.059e-16  2.165e-01   0.000  1.00000
## brand3      -2.500e-01  2.165e-01  -1.155  0.45437
## price1       7.500e-01  2.165e-01   3.464  0.17891
## price2      -4.783e-16  2.165e-01   0.000  1.00000
## price3       6.190e-16  2.165e-01   0.000  1.00000
## weight1      5.000e+00  2.165e-01  23.094  0.02755 *
## weight2      2.000e+00  2.165e-01   9.238  0.06865 .
## weight3     -1.250e+00  2.165e-01  -5.774  0.10918
## crust1       1.750e+00  1.250e-01  14.000  0.04540 *
## cheese1     -2.500e-01  1.250e-01  -2.000  0.29517
## size1       -2.500e-01  1.250e-01  -2.000  0.29517
## toppings1    1.125e+00  1.250e-01   9.000  0.07045 .
## spicy1       7.500e-01  1.250e-01   6.000  0.10514
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5 on 1 degrees of freedom
## Multiple R-squared:  0.9993, Adjusted R-squared:  0.989
## F-statistic: 97.07 on 14 and 1 DF,  p-value: 0.0794
```

```r
# List of conjoint attributes
conjoint_attributes <- c('brand', 'price', 'weight', 'crust', 'cheese', 'size', 'toppings', 'spicy')

# Initialize lists to store results
level_name <- list()
part_worth <- list()
part_worth_range <- c()
important_levels <- list()

# Loop through each attribute to calculate part-worths
for (item in conjoint_attributes) {
  nlevels <- length(unique(df[[item]]))
  levels <- levels(df[[item]])
  level_name[[item]] <- levels

  # Extract part-worths for the current attribute
  coef_names <- names(coef(model_fit))
  attribute_coef <- coef_names[grepl(item, coef_names)]

  new_part_worth <- coef(model_fit)[attribute_coef]
  new_part_worth <- c(new_part_worth, (-1) * sum(new_part_worth))

  # Ensure the part-worths are in the correct order
  part_worth[[item]] <- setNames(new_part_worth, levels)

  # Identify the most important level
  important_levels[[item]] <- which.max(new_part_worth)
  part_worth_range <- c(part_worth_range, max(new_part_worth) - min(new_part_worth))
}

# Calculate attribute importance
attribute_importance <- round(100 * (part_worth_range / sum(part_worth_range)), 2)

# Print results
print("-----------------------------------------------------------------")
```

```
## $brand
## [1] "Dominos"    "Onesta"     "Oven Story" "Pizza hut"
##
## $price
## [1] "$1.00" "$2.00" "$3.00" "$4.00"
##
## $weight
## [1] "100g" "200g" "300g" "400g"
##
## $crust
## [1] "thick" "thin"
##
## $cheese
## [1] "Cheddar"    "Mozzarella"
##
## $size
## [1] "large"   "regular"
##
## $toppings
## [1] "mushroom" "paneer"
##
## $spicy
## [1] "extra"  "normal"
```

```
## $brand
##       Dominos        Onesta    Oven Story     Pizza hut
## -6.909540e-16 -3.059415e-16 -2.500000e-01  2.500000e-01
##
## $price
##         $1.00         $2.00         $3.00         $4.00
##  7.500000e-01 -4.782659e-16  6.189536e-16 -7.500000e-01
##
## $weight
##  100g  200g  300g  400g
##  5.00  2.00 -1.25 -5.75
##
## $crust
## thick  thin
##  1.75 -1.75
##
## $cheese
##    Cheddar Mozzarella
##      -0.25       0.25
##
## $size
##   large regular
##   -0.25    0.25
##
## $toppings
## mushroom   paneer
##    1.125   -1.125
##
## $spicy
##  extra normal
##   0.75  -0.75
```

```r
print(part_worth_range)
```

```
## [1]  0.50  1.50 10.75  3.50  0.50  0.50  2.25  1.50
```

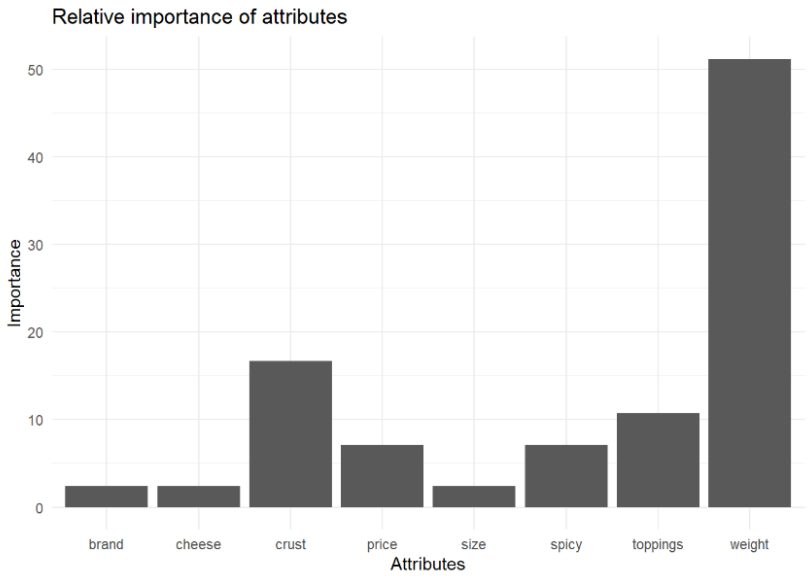
```
print(important_levels)
```

```
## $brand
##
## 4
##
## $price
## price1
##      1
##
## $weight
## weight1
##       1
##
## $crust
## crust1
##      1
##
## $cheese
##
## 2
##
## $size
##
## 2
##
## $toppings
## toppings1
##         1
##
## $spicy
## spicy1
##      1
```

```
print(attribute_importance)
```

```
## [1]  2.38  7.14 51.19 16.67  2.38  2.38 10.71  7.14
```

```
# Plot the relative importance of attributes
ggplot(data.frame(Attribute = conjoint_attributes, Importance = attribute_importance),
       aes(x = Attribute, y = Importance)) +
  geom_bar(stat = "identity") +
  labs(title = 'Relative importance of attributes', x = 'Attributes', y = 'Importance') +
  theme_minimal()
```



```
# Calculate utility for each profile
utility <- apply(df, 1, function(row) {
  sum(sapply(conjoint_attributes, function(attr) part_worth[[attr]][row[[attr]]]))
})

df$utility <- utility

# Print the profile with the highest utility score
print("The profile that has the highest utility score:")
```

```
## [1] "The profile that has the highest utility score:"
```

```
print(df[which.max(df$utility), ])
```

```
##         brand price weight crust     cheese  size toppings spicy ranking
## 10 Oven Story $4.00   100g thick Mozzarella large mushroom extra      16
##    utility
## 10   7.625
```

```
# Print preferred levels for each attribute
for (i in conjoint_attributes) {
  print(paste("Preferred level in", i, "is ::", level_name[[i]][important_levels[[i]]]))
}
```

```
## [1] "Preferred level in brand is :: Pizza hut"
## [1] "Preferred level in price is :: $1.00"
## [1] "Preferred level in weight is :: 100g"
## [1] "Preferred level in crust is :: thick"
## [1] "Preferred level in cheese is :: Mozzarella"
## [1] "Preferred level in size is :: regular"
## [1] "Preferred level in toppings is :: mushroom"
## [1] "Preferred level in spicy is :: extra"
```

Python code and results:

Q1:

```
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from factor_analyzer import FactorAnalyzer
import matplotlib.pyplot as plt
import seaborn as sns
from adjustText import adjust_text
```

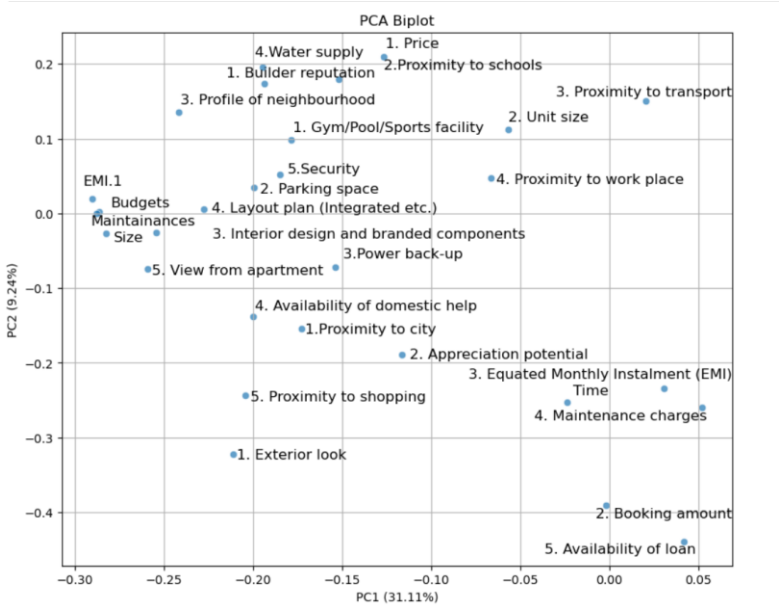
```
# Load the dataset
survey_df = pd.read_csv('E:\\JESIN\\DOCUMENTS\\scma\\A4\\Survey.csv', header=0)

# Check dataset structure
print(survey_df.shape)
print(survey_df.columns)
print(survey_df.head())
print(survey_df.info())

# Remove NA values
print(survey_df.isna().sum())  # Check for NA values
survey_df = survey_df.dropna()  # Drop rows with NA values
```

```
# Select columns of interest for PCA and factor analysis
sur_int = survey_df.iloc[:, 17:46]

# Standardize the data
scaler = StandardScaler()
sur_int_scaled = scaler.fit_transform(sur_int)
```

```
# Performing PCA using scikit-learn
pca = PCA(n_components=5)
pca.fit(sur_int_scaled)

# Explained variance
explained_variance = pca.explained_variance_ratio_ * 100
print(f"Explained variance by each component: {explained_variance}")
```

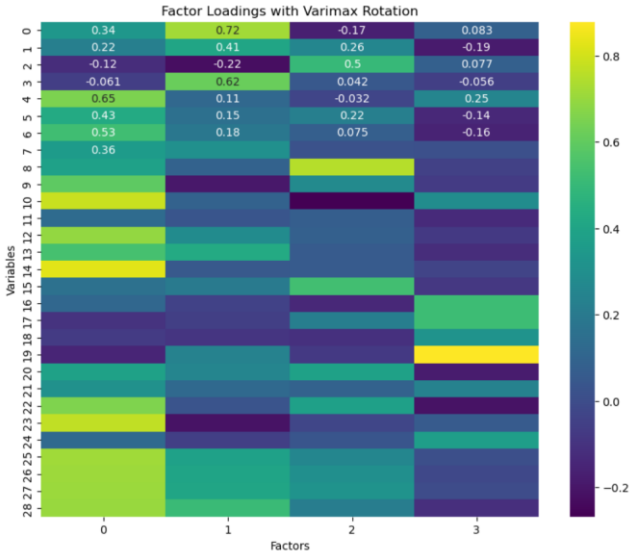
```
Explained variance by each component: [31.11037264  9.23601355  7.22509998  6.4059801   5.80393243]
```

```
# PCA biplot function
def biplot_scores(pca, X):
    plt.figure(figsize=(10, 8))
    sns.scatterplot(x=pca.components_[0], y=pca.components_[1], alpha=0.7)
    texts = []
    for i, txt in enumerate(X.columns):
        texts.append(plt.text(pca.components_[0][i], pca.components_[1][i], txt, fontsize=12))
    adjust_text(texts)

    xlabel = f'PC1 ({explained_variance[0]:.2f}%)'
    ylabel = f'PC2 ({explained_variance[1]:.2f}%)'

    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.title('PCA Biplot')
    plt.grid(True)
    plt.show()

# Plot the PCA biplot
biplot_scores(pca, sur_int)
```

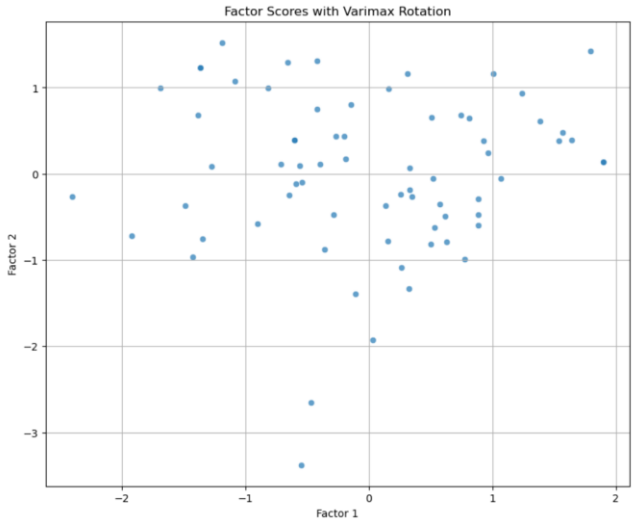


```python
# Performing Factor Analysis using GPArotation equivalent
fa = FactorAnalyzer(rotation='promax', n_factors=5)
fa.fit(sur_int_scaled)

# Get factor loadings
loadings = fa.loadings_
print(f"Factor Loadings:\n{loadings}")

# Get communalities
communalities = fa.get_communalities()
print(f"Communalities:\n{communalities}")

# Get factor scores
factor_scores = fa.transform(sur_int_scaled)
print(f"Factor Scores:\n{factor_scores}")

# Additional Factor Analysis with Varimax rotation
fa_varimax = FactorAnalyzer(rotation='varimax', n_factors=4)
fa_varimax.fit(sur_int_scaled)

# Get Varimax factor loadings
varimax_loadings = fa_varimax.loadings_
print(f"Varimax Factor Loadings:\n{varimax_loadings}")

# Get Varimax communalities
varimax_communalities = fa_varimax.get_communalities()
print(f"Varimax Communalities:\n{varimax_communalities}")

# Get Varimax factor scores
varimax_factor_scores = fa_varimax.transform(sur_int_scaled)
print(f"Varimax Factor Scores:\n{varimax_factor_scores}")
```

```python
# Plotting the factor loadings
def plot_factor_loadings(loadings, title):
    plt.figure(figsize=(10, 8))
    sns.heatmap(loadings, annot=True, cmap='viridis')
    plt.title(title)
    plt.xlabel('Factors')
    plt.ylabel('Variables')
    plt.show()

plot_factor_loadings(varimax_loadings, 'Factor Loadings with Varimax Rotation')
```




```python
# Plotting factor scores
plt.figure(figsize=(10, 8))
sns.scatterplot(x=varimax_factor_scores[:, 0], y=varimax_factor_scores[:, 1], alpha=0.7)
plt.xlabel('Factor 1')
plt.ylabel('Factor 2')
plt.title('Factor Scores with Varimax Rotation')
plt.grid(True)
plt.show()
```

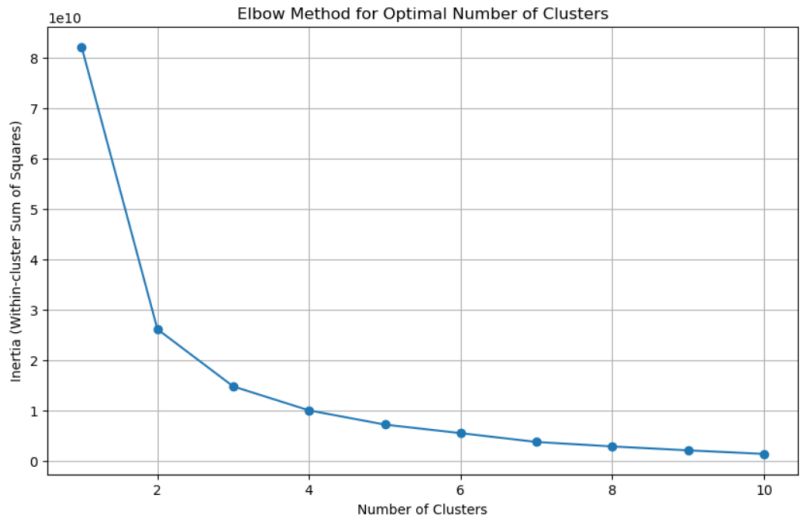


Q2:

```
# Import required libraries
import pandas as pd
import numpy as np
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import KMeans
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Load the dataset
survey_df = pd.read_csv('E:\\JESIN\\DOCUMENTS\\scma\\A4\\Survey.csv')

# Select columns of interest for clustering
sur_int = survey_df.iloc[:, 17:46]
```

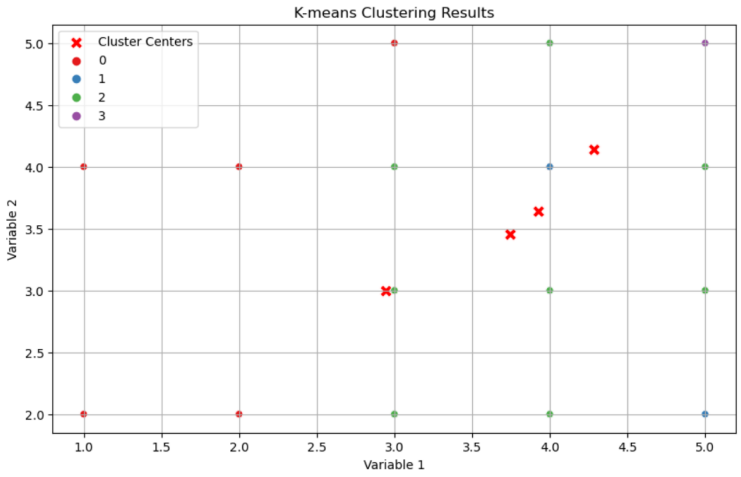
```
# Determine Optimal Number of Clusters using the Elbow Method
inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=123, n_init=25)
    kmeans.fit(sur_int)
    inertia.append(kmeans.inertia_)

# Plotting the Elbow Method
plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), inertia, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia (Within-cluster Sum of Squares)')
plt.title('Elbow Method for Optimal Number of Clusters')
plt.grid(True)
plt.show()
```



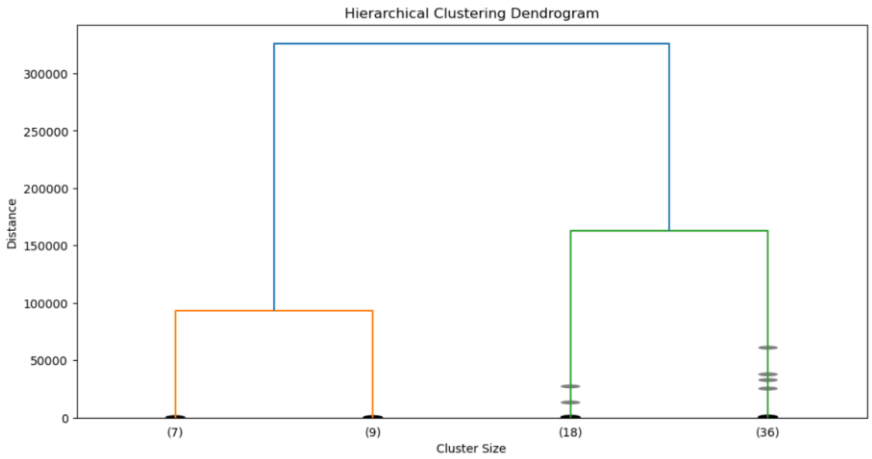
```
# Perform k-means clustering
kmeans = KMeans(n_clusters=4, random_state=123, n_init=25)
km_res = kmeans.fit(sur_int)
```

```
C:\Users\user\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with M
s. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

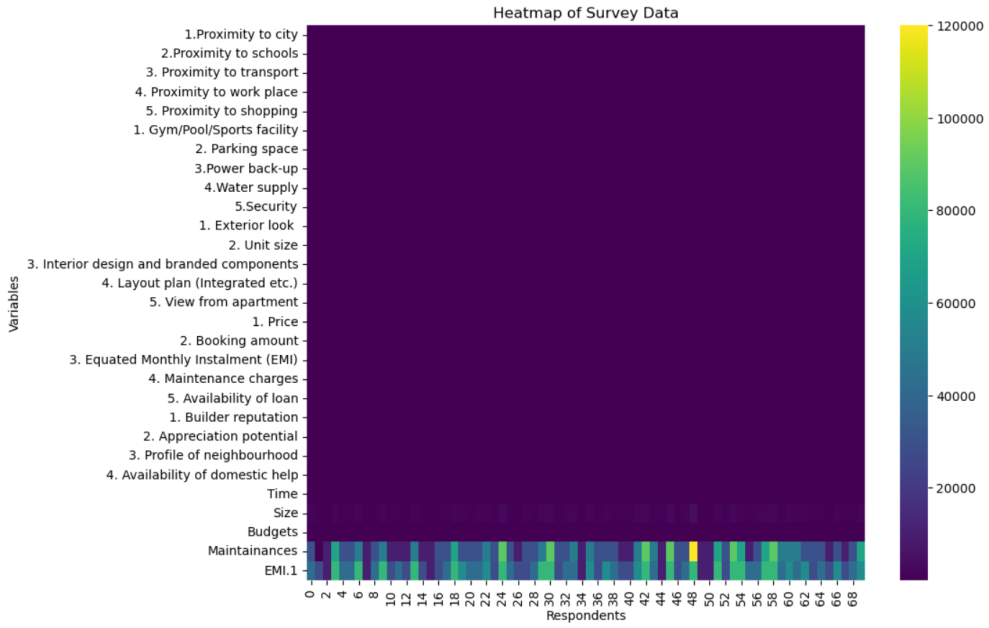
```
# Visualizing k-means Clustering Results
plt.figure(figsize=(10, 6))
sns.scatterplot(x=km_res.cluster_centers_[:, 0], y=km_res.cluster_centers_[:, 1], color='red', marker='X', s=100, label='Cluster Centers')
sns.scatterplot(x=sur_int.iloc[:, 0], y=sur_int.iloc[:, 1], hue=km_res.labels_, palette='Set1', legend='full')
plt.title('K-means Clustering Results')
plt.xlabel('Variable 1')
plt.ylabel('Variable 2')
plt.grid(True)
plt.legend()
plt.show()
```




```
# Perform Hierarchical Clustering (Ward's method)
Z = linkage(sur_int, method='ward')

# Plotting the Dendrogram
plt.figure(figsize=(12, 6))
dendrogram(Z, p=4, truncate_mode='lastp', orientation='top', leaf_font_size=10, show_contracted=True)
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Cluster Size')
plt.ylabel('Distance')
plt.show()
```



```
# Heatmap of clustered data
plt.figure(figsize=(10, 8))
sns.heatmap(sur_int.T, cmap='viridis', cbar=True)
plt.title('Heatmap of Survey Data')
plt.xlabel('Respondents')
plt.ylabel('Variables')
plt.show()
```



Q3:

```
# Import required libraries
import pandas as pd
import numpy as np
from scipy.spatial.distance import pdist, squareform
from sklearn.manifold import MDS
import matplotlib.pyplot as plt
```

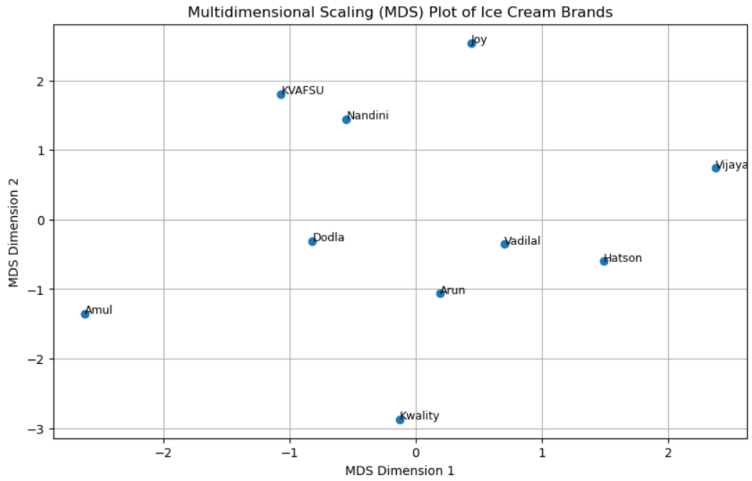
```
# Load the dataset
icecream = pd.read_csv('E:\\JESIN\\DOCUMENTS\\scma\\A4\\icecream.csv')
```

```
# Display the dataset to confirm it's loaded correctly
print(icecream.head())
```

```
    Brand  Price  Availability  Taste  Flavour  Consistency  Shelflife
0     Amul      4             5      4        3            4          3
1  Nandini      3             2      3        2            3          3
2  Vadilal      2             2      4        3            4          4
3   Vijaya      3             1      3        5            3          4
4    Dodla      3             3      3        4            4          3
```

```
# Extract the attributes for MDS (excluding the Brand column)
icecream_mds = icecream.iloc[:, 1:]  # Assuming 'Brand' is the first column

# Calculate the distance matrix
dist_matrix = squareform(pdist(icecream_mds))

# Perform Multidimensional Scaling (MDS)
mds = MDS(n_components=2, dissimilarity='precomputed', random_state=42)
mds_results = mds.fit_transform(dist_matrix)
```

```
# Create a DataFrame for the MDS results
plot_data = pd.DataFrame({
    'x': mds_results[:, 0],  # X-axis coordinates
    'y': mds_results[:, 1],  # Y-axis coordinates
    'brand': icecream['Brand']  # Brand names
})

# Plot the MDS results
plt.figure(figsize=(10, 6))
plt.scatter(plot_data['x'], plot_data['y'])

for i in range(plot_data.shape[0]):
    plt.text(plot_data['x'][i], plot_data['y'][i], plot_data['brand'][i], fontsize=9)

plt.title('Multidimensional Scaling (MDS) Plot of Ice Cream Brands')
plt.xlabel('MDS Dimension 1')
plt.ylabel('MDS Dimension 2')
plt.grid(True)
plt.show()
```



Q4:

```
import pandas as pd, numpy as np

df=pd.read_csv('pizza_data.csv')
```

```
import statsmodels.api as sm
import statsmodels.formula.api as smf

model='ranking ~ C(brand,Sum)+C(price,Sum)+C(weight,Sum)+C(crust,Sum)+C(cheese,Sum)+C(size,Sum)+C(toppings,Sum)+C(spicy,Sum)'
model_fit=smf.ols(model,data=df).fit()

print(model_fit.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                ranking   R-squared:                       0.999
Model:                            OLS   Adj. R-squared:                  0.989
Method:                 Least Squares   F-statistic:                     97.07
Date:                Mon, 08 Jul 2024   Prob (F-statistic):             0.0794
Time:                        10:30:08   Log-Likelihood:                 10.568
No. Observations:                  16   AIC:                             8.864
Df Residuals:                       1   BIC:                             20.45
Df Model:                          14
Covariance Type:            nonrobust
===================================================================================================
                                      coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------------------
Intercept                          8.5000      0.125     68.000      0.009       6.912      10.088
C(brand, Sum)[S.Dominos]       -1.11e-15      0.217  -5.13e-15      1.000      -2.751       2.751
C(brand, Sum)[S.Onesta]        7.327e-15      0.217   3.38e-14      1.000      -2.751       2.751
C(brand, Sum)[S.Oven Story]       -0.2500      0.217     -1.155      0.454      -3.001       2.501
C(price, Sum)[S.$1.00]             0.7500      0.217      3.464      0.179      -2.001       3.501
C(price, Sum)[S.$2.00]        -9.992e-15      0.217  -4.62e-14      1.000      -2.751       2.751
C(price, Sum)[S.$3.00]         3.997e-15      0.217   1.85e-14      1.000      -2.751       2.751
C(weight, Sum)[S.100g]             5.0000      0.217     23.094      0.028       2.249       7.751
C(weight, Sum)[S.200g]             2.0000      0.217      9.238      0.069      -0.751       4.751
C(weight, Sum)[S.300g]            -1.2500      0.217     -5.774      0.109      -4.001       1.501
C(crust, Sum)[S.thick]             1.7500      0.125     14.000      0.045       0.162       3.338
C(cheese, Sum)[S.Cheddar]         -0.2500      0.125     -2.000      0.295      -1.838       1.338
C(size, Sum)[S.large]             -0.2500      0.125     -2.000      0.295      -1.838       1.338
C(toppings, Sum)[S.mushroom]       1.1250      0.125      9.000      0.070      -0.463       2.713
C(spicy, Sum)[S.extra]             0.7500      0.125      6.000      0.105      -0.838       2.338
==============================================================================
Omnibus:                       30.796   Durbin-Watson:                   2.000
Prob(Omnibus):                  0.000   Jarque-Bera (JB):                2.667
Skew:                           0.000   Prob(JB):                        0.264
Kurtosis:                       1.000   Cond. No.                         2.00
==============================================================================
```

```
conjoint_attributes = ['brand','price','weight','crust','cheese','size','toppings','spicy']
```

```
level_name = []
part_worth = []
part_worth_range = []
important_levels = {}
end = 1  # Initialize index for coefficient in params

for item in conjoint_attributes:
    nlevels = len(list(np.unique(df[item])))
    level_name.append(list(np.unique(df[item])))

    begin = end
    end = begin + nlevels -1

    new_part_worth = list(model_fit.params[begin:end])
    new_part_worth.append((-1)*sum(new_part_worth))
    important_levels[item] = np.argmax(new_part_worth)
    part_worth.append(new_part_worth)
    print(item)
    part_worth_range.append(max(new_part_worth) - min(new_part_worth))

print("------------------------------------------------------------")
print("level name:")
print(level_name)
print("npw with sum element:")
print(new_part_worth)
print("imp level:")
print(important_levels)
print("part worth:")
print(part_worth)
print("part_worth_range:")
print(part_worth_range)
print(len(part_worth))
print("important levels:")
print(important_levels)
```

```
brand
price
weight
crust
cheese
size
toppings
spicy
```

```
level name:
[['Dominos', 'Onesta', 'Oven Story', 'Pizza hut'], ['$1.00', '$2.00', '$3.00', '$4.00'], ['100g', '200g', '300g', '400g'], ['thick', 'thin'], ['Cheddar', 'Mozzarella'], ['large', 'regular'], ['mus
hroom', 'paneer'], ['extra', 'normal']]
npw with sum element:
[0.749999999999999, -0.749999999999999]
imp level:
{'brand': 3, 'price': 0, 'weight': 0, 'crust': 0, 'cheese': 1, 'size': 1, 'toppings': 0, 'spicy': 0}
part worth:
[[-1.1102230246251565e-15, 7.327471962526033e-15, -0.25000000000000555, 0.24999999999999933], [0.7500000000000084, -9.992007221626409e-15, 3.9968028886505635e-15, -0.7500000000000024], [5.0, 1.999
9999999999973, -1.2499999999999947, -5.750000000000003], [1.7499999999999996, -1.7499999999999996], [-0.2500000000000001, 0.2500000000000001], [-0.24999999999999883, 0.24999999999999883], [1.12499
99999999978, -1.1249999999999978], [0.749999999999999, -0.749999999999999]]
part_worth_range:
[0.5000000000000049, 1.5000000000000109, 10.750000000000004, 3.499999999999999, 0.5000000000000002, 0.49999999999999767, 2.2499999999999956, 1.499999999999998]
8
important levels:
{'brand': 3, 'price': 0, 'weight': 0, 'crust': 0, 'cheese': 1, 'size': 1, 'toppings': 0, 'spicy': 0}
```

```
attribute_importance = []

for i in part_worth_range:
    attribute_importance.append(round(100*(i/sum(part_worth_range)),2))

print(attribute_importance)
```

```
[2.38, 7.14, 51.19, 16.67, 2.38, 2.38, 10.71, 7.14]
```

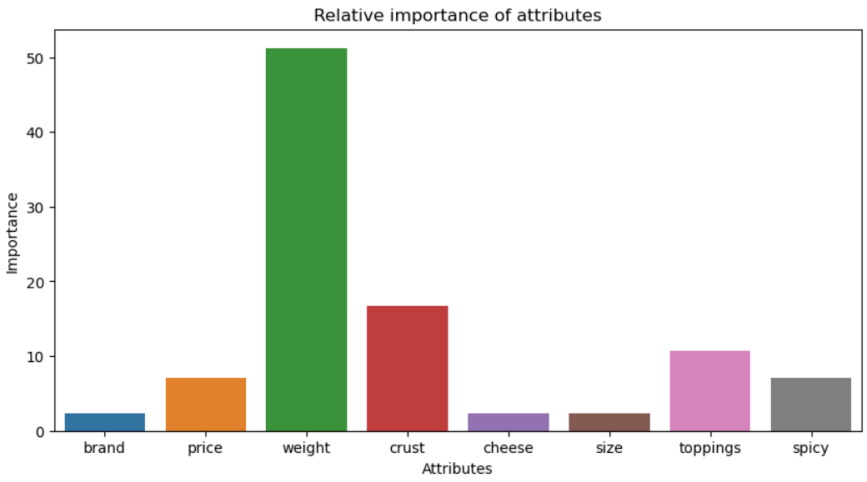
```
part_worth_dict={}
attrib_level={}

for item,i in zip(conjoint_attributes,range(0,len(conjoint_attributes))):
    print("Attribute :",item)
    print("    Relative importance of attribute ",attribute_importance[i])
    print("    Level wise part worths: ")
    for j in range(0,len(level_name[i])):
        print(i)
        print(j)
        print("          {}:{}".format(level_name[i][j],part_worth[i][j]))
        part_worth_dict[level_name[i][j]]=part_worth[i][j]
        attrib_level[item]=(level_name[i])

part_worth_dict
```

```
Attribute : brand
    Relative importance of attribute  2.38
    Level wise part worths:
0
0
          Dominos:-1.1102230246251565e-15
0
1
          Onesta:7.327471962526033e-15
0
2
          Oven Story:-0.25000000000000555
0
3
          Pizza hut:0.24999999999999933
Attribute : price
    Relative importance of attribute  7.14
    Level wise part worths:
1
0
          $1.00:0.7500000000000084
1
1
          $2.00:-9.992007221626409e-15
1
2
```

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10,5))
sns.barplot(x=conjoint_attributes,y=attribute_importance)
plt.title('Relative importance of attributes')
plt.xlabel('Attributes')
plt.ylabel('Importance')
```

```
C:\Users\user\anaconda3\Lib\site-packages\seaborn\_oldcore.
in a future version.
  order = pd.unique(vector)
```

```
Text(0, 0.5, 'Importance')
```




```
utility = []

for i in range(df.shape[0]):
    score = part_worth_dict[df['brand'][i]]+part_worth_dict[df['price'][i]]+part_worth_dict[df['weight'][i]]+part_worth_d
    utility.append(score)

df['utility'] = utility
utility
```

```
[2.6250000000000093,
 3.3750000000000044,
 0.37500000000000033,
 -6.375000000000003,
 -0.37500000000000944,
 4.375000000000007,
 -1.3749999999999791,
 -4.624999999999996,
 -3.6250000000000147,
 7.6249999999999885,
 -5.3749999999999885,
 -2.3750000000000133,
 1.3750000000000053,
 6.374999999999997,
 -7.625000000000003,
 5.624999999999996]
```

```
print("The profile that has the highest utility score :",'\n', df.iloc[np.argmax(utility
```

```
The profile that has the highest utility score :
 brand       Oven Story
price            $4.00
weight            100g
crust            thick
cheese      Mozzarella
size              large
toppings      mushroom
spicy             extra
ranking             16
utility          7.625
Name: 9, dtype: object
```

```
for i,j in zip(attrib_level.keys(),range(0,len(conjoint_attributes))):
    print("Preferred level in {} is :: {}".format(i,level_name[j][important_levels[i]]))
```

```
Preferred level in brand is :: Pizza hut
Preferred level in price is :: $1.00
Preferred level in weight is :: 100g
Preferred level in crust is :: thick
Preferred level in cheese is :: Mozzarella
Preferred level in size is :: regular
Preferred level in toppings is :: mushroom
Preferred level in spicy is :: extra
```

Interpretations:

Q1:

R:

Principal Components Analysis (PCA)

Using GPArotation Package:

- The first five components explain 73% of the total variance in the data.
- Component Interpretations:
    - RC1: Strong loadings from variables related to proximity to shopping, security, and exterior look.
    - RC2: High loadings from variables like gym/pool facilities, parking space, and water supply.
    - RC3: Emphasizes proximity to work and its convenience.
    - RC4: Reflects factors like interior design and layout plan.
    - RC5: Relates to financial aspects such as price, booking amount, and loan availability.
- The model fits well with an off-diagonal fit of 0.95, indicating good model fit.

Component Name	What it defines
RC1	Residential Proximity
RC2	Amenities and Services
RC3	Workplace Proximity
RC4	Internal Features
RC5	Financial Factors

Using FactoMineR Package:

- The first principal component (Dim.1) accounts for 31.1% of the variance, and the top five components explain 59.8% cumulatively.
- Individuals and variables are plotted to show their relationships across dimensions.
- Interpretation of Dimensions:
    - Dim.1: Strongly associated with variables like proximity to shopping and security.
    - Dim.2: Reflects factors related to amenities and size.
    - Dim.3: Emphasizes features like neighborhood profile and availability of help.
    - Dim.4: Focuses on financial factors such as pricing and maintenance charges.
- The Biplot Analysis shows relationships between variables and dimensions, highlighting correlations and overlaps.

Factor Analysis

- Variables are grouped into factors that explain the underlying structure of the data.
- Interpretation of Factors:
    - MR1: Combines variables related to proximity, exterior appearance, and builder reputation.
    - MR2: Reflects financial aspects like pricing, booking amounts, and loan availability.
    - MR3: Includes variables related to amenities, interior design, and layout plans.
    - MR4: Emphasizes neighborhood profile and availability of domestic help.
- Communality and Scores shows how well each variable contributes to each factor and provides individual scores for each observation.

Factor Name	Latent Variable Name
MR1	Proximity and Exterior
MR2	Financial and Loan Availability
MR3	Amenities and Internal Features
MR4	Neighborhood Profile and Domestic Help

Python:

Principal Component Analysis (PCA)

- PCA identified five principal components (PC1 to PC5).
    - PC1 (31.11%): This component explains the most variance, capturing 31.11% of the total variability in the dataset.
    - PC2 (9.24%), PC3 (7.23%), PC4 (6.41%), PC5 (5.80%): These components explain additional portions of the variance, totaling approximately 60% when combined.
- Biplot shows the relationships and projections of features onto PC1 and PC2, aiding in understanding which features are most influential.

Factor Analysis

- Factor Loadings
    - Factor analysis explored underlying factors (latent variables) in the dataset.
    - Loadings: These coefficients indicate the correlation between each variable and each extracted factor.
    - Communalities: Each variable's communality shows how much of its variance is explained by all extracted factors.
- Varimax Rotation
    - Varimax rotation was applied to simplify and enhance the interpretability of factor analysis results.
    - Rotated Loadings: These show the new, rotated factor loadings after applying Varimax rotation.
    - Rotated Communalities: Indicate the variance explained by the rotated factors for each variable.
    - Factor Scores: Scores of each observation on each factor, indicating their relationship strength with each rotated factor.

Q2:

R:

- The Gap Statistic suggests that there can be a maximum of 10 clusters, indicating the dataset can be partitioned into at the most 10 distinct groups.
- K-means clustering identifies 4 respondent groups with similar characteristics, visualized through cluster boundaries. The cluster plot clearly defines the clusters.
- The Cluster Dendrogram shows the nested structure and relationships between clusters.
- The Heatmap visualizes patterns and similarities within each cluster, highlighting prominent variables for each group.

Python:

- The Elbow Method suggest that 10 clusters can be created.
- K-means clustering groups respondents into 4 clusters, with a scatter plot showing cluster distribution and their respective centers.
- The Dendrogram confirms 4 main clusters, illustrating the hierarchical relationship between them.
- The Heatmap displays variable intensity for each respondent, identifying patterns and similarities within clusters.

Q3:

R:

- MDS is performed to reduce the dimensionality of the dataset, producing a two-dimensional representation of the ice cream brands based on the similarities/dissimilarities in the dataset.
- For the MDS plot, the resulting two-dimensional MDS coordinates are plotted. Each point represents an ice cream brand, with proximity indicating similarity in attributes. The plot helps visualize the relative positioning and clustering of brands based on their attributes.

Python:

- A distance matrix is calculated using pairwise distances between the attributes of the ice cream brands, forming the basis for MDS analysis.
- MDS is performed to reduce the dimensionality of the dataset, producing a two-dimensional representation of the ice cream brands based on the distance matrix.
- The resulting two-dimensional MDS coordinates are plotted. Again each point represents an ice cream brand, with proximity indicating similarity in attributes.

Q4:

R and Python (both gave the exact same result):

- The starting point of the model is 8.500. This is the baseline preference score when all attributes are at their basic levels. The p-value of 0.009 shows that this starting point is statistically important.
- weight1: This value is 5.000, meaning that if the weight changes from the basic level to weight1, the preference score goes up by 5 points. The p-value of 0.028 indicates that this is an important factor.
- crust1: This value is 1.750, meaning that changing the crust to crust1 increases the preference score by 1.750 points. The p-value of 0.045 shows that this is also an important factor.

- Calculate Part-Worth Utilities: These values show how much each level of an attribute affects the overall preference.
- Compute Attribute Importance: This shows how much each attribute (like weight, crust, toppings) matters in the overall preference. Importance is based on the range of part-worth values for each attribute.
- Most Important Attributes:
    - weight (51.19%): Weight is the most important attribute, making up 51.19% of the overall preference.
    - crust (16.67%): Crust is the second most important, contributing 16.67%.
    - toppings (10.71%): Toppings are also important, contributing 10.71%.
- Preferred Levels: The most liked levels for each attribute are:
    - brand: Pizza hut
    - price: $1.00
    - weight: 100g
    - crust: thick
    - cheese: Mozzarella
    - size: regular
    - toppings: mushroom
    - spicy: extra
- Highest Utility Profile: The best combination of attributes according to the model is:
    - Brand: Oven Story
    - Price: $4.00
    - Weight: 100g
    - Crust: thick
    - Cheese: Mozzarella
    - Size: large
    - Toppings: mushroom
    - Spicy: extra
- This profile has the highest preference score of 7.625, making it the most preferred combination.