VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A4 : Multivariate Analysis and Business Analytics Applications

AAKASH K

V01110153

Date of Submission: 08-07-2024

CONTENTS

Sl. No.	Title	Page No.
1.	Introduction	**3**
2.	Objective	**4**
3.	Business Significance	**4**
4.	R code Results and Interpretations	**5**
5.	Python code Results and Interpretations	**20**

Introduction:

The assignment involves conducting multivariate analysis on the “survey.csv”, “icecream.csv” and "pizza_data.csv" datasets.

Survey dataset offers a detailed view of residential property preferences across major cities such as Bangalore, Chennai, and Delhi, focusing on factors that influence potential homebuyers. It includes demographic details like age, sex, occupation, and income, along with preferences regarding house types, budgets, and desired amenities. By examining variables such as planning time frames, proximity to amenities, and media influences, this dataset aims to reveal patterns and trends in homebuying behavior. The insights gained from this analysis can help real estate professionals tailor their strategies to better align with consumer needs and preferences.

The Icecream dataset is a detailed record of various ice cream brands, capturing key attributes that influence consumer preferences. It includes six distinct brands: Amul, Nandini, Vadilal, Vijaya, Dodla, Hatson, Arun, Joy, Kwality, and KVAFSU. For each brand, the dataset documents six attributes: Price, Availability, Taste, Flavour, Consistency, and Shelflife. These attributes are quantified on a scale from 1 to 5, where higher values typically represent more favorable conditions or qualities. This data is instrumental for analyzing consumer preferences, brand positioning, and the overall market landscape in the ice cream sector. By applying techniques such as Multidimensional Scaling (MDS), one can gain insights into the relative positioning of these brands based on their attributes.

The pizza dataset captures various attributes of pizzas offered by different brands, specifically Dominos, Pizza Hut, Onesta, and Oven Story. It includes detailed information on each pizza, such as the price, weight, crust type (thin or thick), cheese type (Mozzarella or Cheddar), size (regular or large), toppings, spiciness level (normal or extra), and a ranking score. This dataset enables the analysis of how different combinations of attributes influence the overall ranking of pizzas, providing insights into consumer preferences and brand performance in the competitive pizza market.

Objective:

1. **Perform Principal Component Analysis (PCA) and Factor Analysis on the "Survey.csv"** dataset to identify and understand the underlying dimensions of the data. Validate the results by examining the variance explained by the principal components and factor loadings.

2. **Conduct Cluster Analysis on the "Survey.csv"** dataset to group respondents based on their background variables. Utilize appropriate clustering algorithms (e.g., k-means, hierarchical clustering) to identify distinct clusters. Evaluate the clustering results using metrics like silhouette scores and interpret the characteristics of each cluster to provide insights into respondent segmentation.

3. **Apply Multidimensional Scaling (MDS) to the "icecream.csv"** dataset to visualize the similarities or dissimilarities between different ice cream brands based on their attributes. Interpret the MDS plot to understand the relative positioning of brands in a reduced dimensional space, and discuss how this visualization can inform brand positioning and consumer preference analysis.

4. **Conduct Conjoint Analysis on the "pizza_data.csv" dataset** to determine the part-worth utilities of different pizza attributes (e.g., price, crust, cheese). Analyze the results to understand consumer preferences and attribute importance.

Business Significance:

1. Principal Component Analysis (PCA) and Factor Analysis:

   - Identifying key data dimensions helps simplify complex datasets, allowing businesses to focus on critical factors influencing customer opinions. This leads to more targeted marketing and improved decision-making.

2. Cluster Analysis:

   - Segmenting respondents into distinct clusters enables tailored marketing strategies and personalized offers, enhancing customer engagement and satisfaction.

3. Multidimensional Scaling (MDS):

    - Visualizing brand perceptions helps businesses understand their competitive position and identify opportunities for differentiation and innovation in the market.

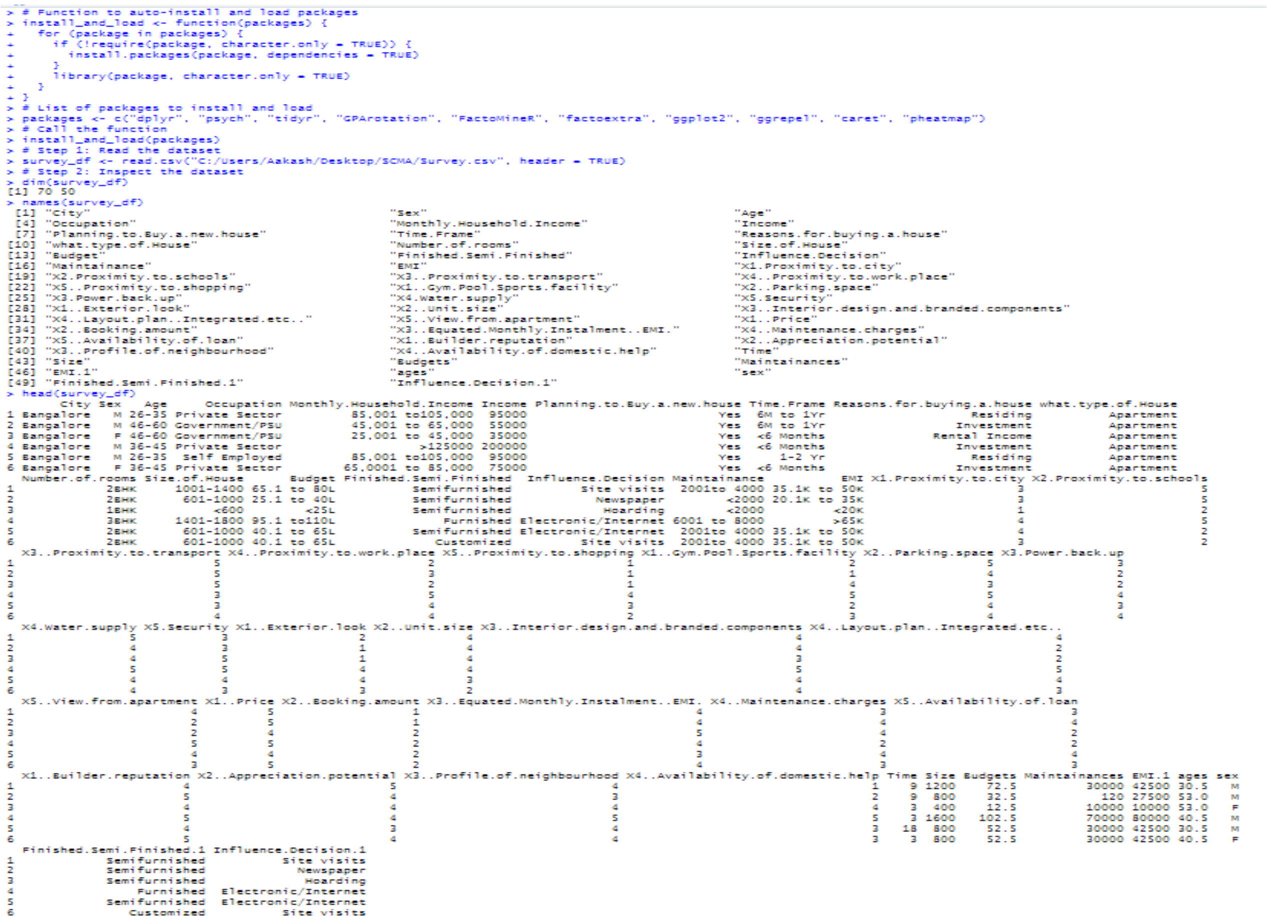
4. Conjoint Analysis:

    - Revealing customer preferences for various pizza attributes aids in optimizing product offerings and pricing strategies, leading to increased customer satisfaction and revenue growth.

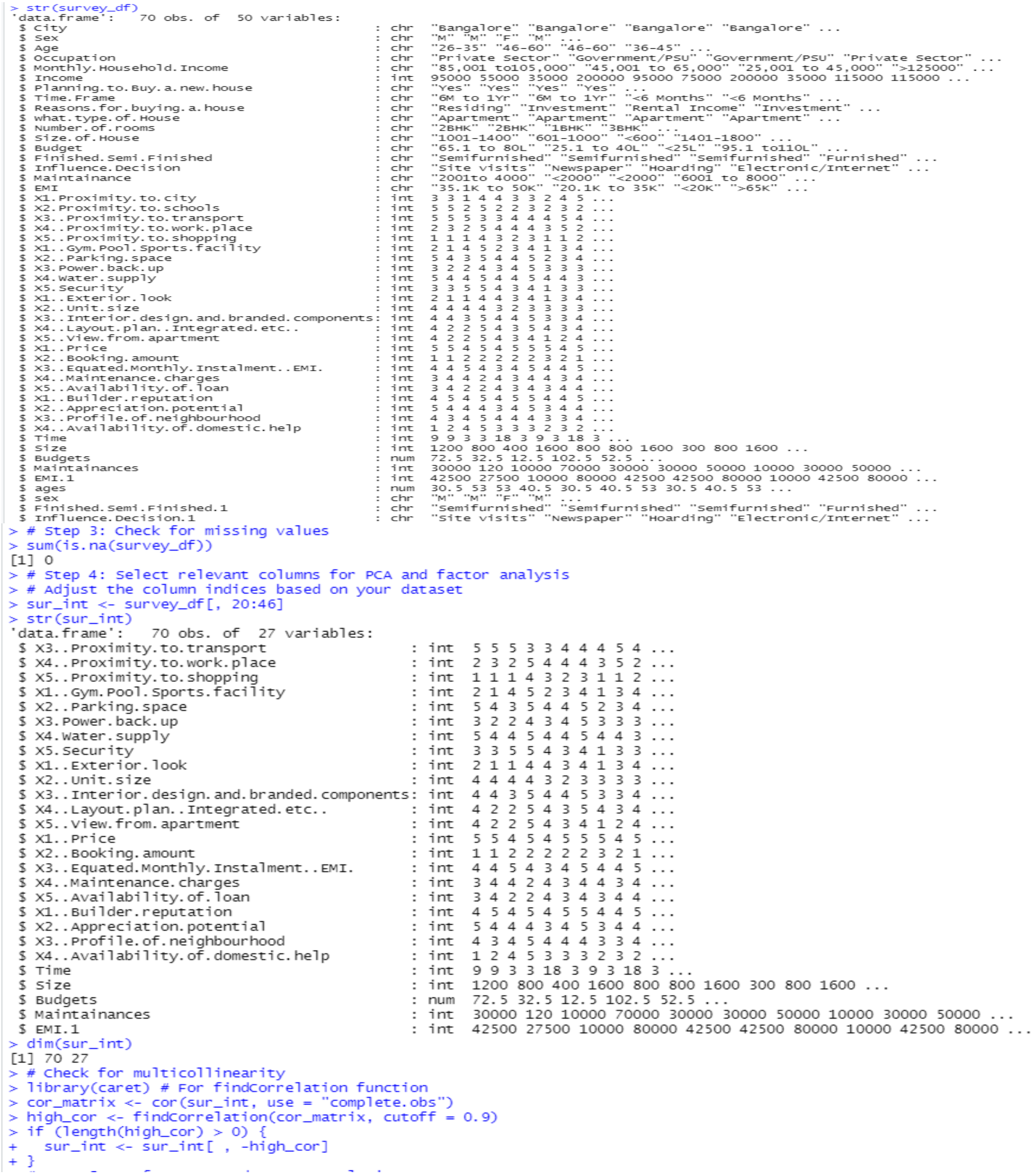
R code results:

1.PCA and Factor

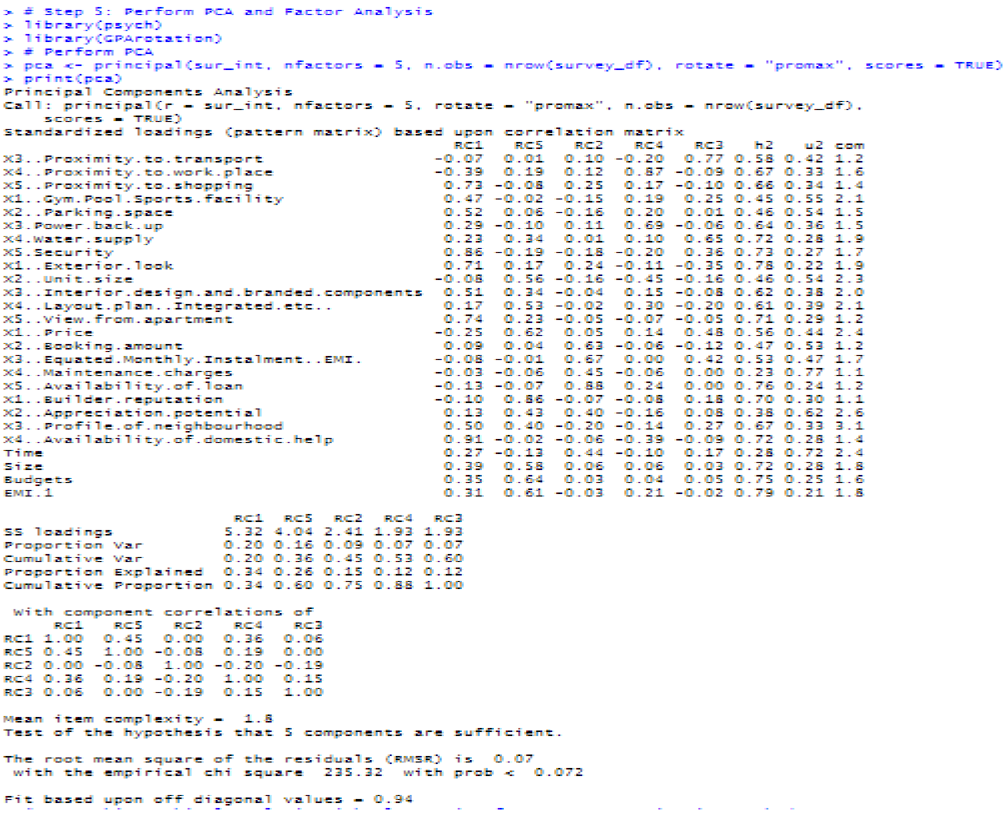
Analysis



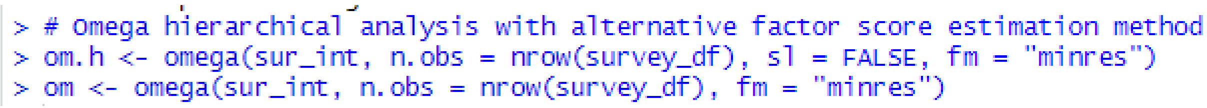
```
> # Function to auto-install and load packages
> install_and_load <- function(packages) {
+   for (package in packages) {
+     if (!require(package, character.only = TRUE)) {
+       install.packages(package, dependencies = TRUE)
+     }
+     library(package, character.only = TRUE)
+   }
+ }
> # List of packages to install and load
> packages <- c("dplyr", "psych", "tidyr", "GPArotation", "FactoMineR", "factoextra", "ggplot2", "ggrepel", "caret", "pheatmap")
> # Call the function
> install_and_load(packages)
> # Step 1: Read the dataset
> survey_df <- read.csv("C:/Users/Aakash/Desktop/SCMA/Survey.csv", header = TRUE)
> # Step 2: Inspect the dataset
> dim(survey_df)
[1] 70 50
> names(survey_df)
 [1] "City"                                       "Sex"                                        "Age"
 [4] "Occupation"                                 "Monthly.Household.Income"                   "Income"
 [7] "Planning.to.Buy.a.new.house"                "Time.Frame"                                 "Reasons.for.buying.a.house"
[10] "what.type.of.House"                         "Number.of.rooms"                            "Size.of.House"
[13] "Budget"                                     "Finished.Semi.Finished"                     "Influence.Decision"
[16] "Maintainance"                               "EMI"                                        "X1.Proximity.to.city"
[19] "X2.Proximity.to.schools"                    "X3..Proximity.to.transport"                 "X4..Proximity.to.work.place"
[22] "X5..Proximity.to.shopping"                  "X1..Gym.Pool.Sports.facility"               "X2..Parking.space"
[25] "X3.Power.back.up"                           "X4.Water.supply"                            "X5.Security"
[28] "X1..Exterior.look"                          "X2..Unit.size"                              "X3..Interior.design.and.branded.components"
[31] "X4..Layout.plan..Integrated.etc.."          "X5..View.from.apartment"                    "X1..Price"
[34] "X2..Booking.amount"                         "X3..Equated.Monthly.Instalment..EMI."       "X4..Maintenance.charges"
[37] "X5..Availability.of.loan"                   "X1..Builder.reputation"                     "X2..Appreciation.potential"
[40] "X3..Profile.of.neighbourhood"               "X4..Availability.of.domestic.help"          "Time"
[43] "Size"                                       "Budgets"                                    "Maintainances"
[46] "EMI.1"                                      "ages"                                       "sex"
[49] "Finished.Semi.Finished.1"                   "Influence.Decision.1"
> head(survey_df)
       City Sex   Age     Occupation Monthly.Household.Income Income Planning.to.Buy.a.new.house Time.Frame Reasons.for.buying.a.house what.type.of.House
1 Bangalore   M 26-35 Private Sector         85,001 to105,000  95000                         Yes  6M to 1Yr                   Residing          Apartment
2 Bangalore   M 46-60 Government/PSU         45,001 to 65,000  55000                         Yes  6M to 1Yr                 Investment          Apartment
3 Bangalore   F 46-60 Government/PSU         25,001 to 45,000  35000                         Yes  <6 Months              Rental Income          Apartment
4 Bangalore   M 36-45 Private Sector                  >125000 200000                         Yes  <6 Months                 Investment          Apartment
5 Bangalore   M 26-35  Self Employed         85,001 to105,000  95000                         Yes     1-2 Yr                   Residing          Apartment
6 Bangalore   F 36-45 Private Sector        65,0001 to 85,000  75000                         Yes  <6 Months                 Investment          Apartment
  Number.of.rooms Size.of.House      Budget Finished.Semi.Finished  Influence.Decision Maintainance          EMI X1.Proximity.to.city X2.Proximity.to.schools
1            2BHK     1001-1400 65.1 to 80L          Semifurnished         Site visits  2001to 4000 35.1K to 50K                    3                       5
2            2BHK      601-1000 25.1 to 40L          Semifurnished           Newspaper        <2000 20.1K to 35K                    3                       5
3            1BHK          <600        <25L          Semifurnished            Hoarding        <2000         <20K                    1                       2
4            3BHK     1401-1800 95.1 to110L              Furnished Electronic/Internet 6001 to 8000         >65K                    4                       5
5            2BHK      601-1000 40.1 to 65L          Semifurnished Electronic/Internet  2001to 4000 35.1K to 50K                    4                       2
6            2BHK      601-1000 40.1 to 65L             Customized         Site visits  2001to 4000 35.1K to 50K                    3                       2
  X3..Proximity.to.transport X4..Proximity.to.work.place X5..Proximity.to.shopping X1..Gym.Pool.Sports.facility X2..Parking.space X3.Power.back.up
1                          5                           2                         1                            2                 5                3
2                          5                           3                         1                            1                 4                2
3                          5                           2                         1                            4                 3                2
4                          3                           5                         4                            5                 5                4
5                          3                           4                         3                            2                 4                3
6                          4                           4                         2                            3                 4                4
  X4.Water.supply X5.Security X1..Exterior.look X2..Unit.size X3..Interior.design.and.branded.components X4..Layout.plan..Integrated.etc..
1               5           3                 2             4                                          4                                 4
2               4           3                 1             4                                          4                                 2
3               4           5                 1             4                                          3                                 2
4               5           5                 4             4                                          5                                 5
5               4           4                 4             3                                          4                                 4
6               4           3                 3             2                                          4                                 3
  X5..View.from.apartment X1..Price X2..Booking.amount X3..Equated.Monthly.Instalment..EMI. X4..Maintenance.charges X5..Availability.of.loan
1                       4         5                  1                                    4                       3                        3
2                       2         5                  1                                    4                       4                        4
3                       2         4                  2                                    5                       4                        2
4                       5         5                  2                                    4                       2                        2
5                       4         4                  2                                    3                       4                        4
6                       3         5                  2                                    4                       3                        3
  X1..Builder.reputation X2..Appreciation.potential X3..Profile.of.neighbourhood X4..Availability.of.domestic.help Time Size Budgets Maintainances EMI.1 ages sex
1                      4                          5                            4                                 1    9 1200    72.5         30000 42500 30.5   M
2                      5                          4                            3                                 2    9  800    32.5           120 27500 53.0   M
3                      4                          4                            4                                 4    3  400    12.5         10000 10000 53.0   F
4                      5                          4                            5                                 5    3 1600   102.5         70000 80000 40.5   M
5                      4                          3                            4                                 3   18  800    52.5         30000 42500 30.5   M
6                      5                          4                            4                                 3    3  800    52.5         30000 42500 40.5   F
  Finished.Semi.Finished.1 Influence.Decision.1
1            Semifurnished          Site visits
2            Semifurnished            Newspaper
3            Semifurnished             Hoarding
4                Furnished  Electronic/Internet
5            Semifurnished  Electronic/Internet
6               Customized          Site visits
```



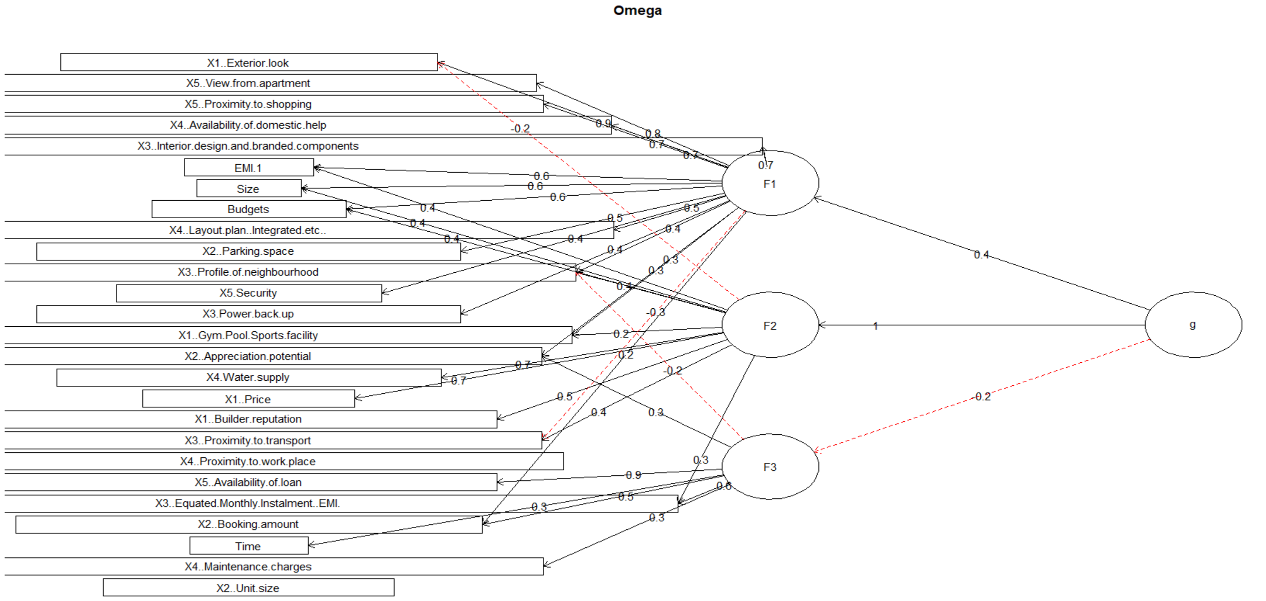
```
> str(survey_df)
'data.frame':	70 obs. of  50 variables:
 $ City                                    : chr  "Bangalore" "Bangalore" "Bangalore" "Bangalore" ...
 $ Sex                                     : chr  "M" "M" "F" "M" ...
 $ Age                                     : chr  "26-35" "46-60" "46-60" "36-45" ...
 $ Occupation                              : chr  "Private Sector" "Government/PSU" "Government/PSU" "Private Sector" ...
 $ Monthly.Household.Income                : chr  "85,001 to105,000" "45,001 to 65,000" "25,001 to 45,000" ">125000" ...
 $ Income                                  : int  95000 55000 35000 200000 95000 75000 200000 35000 115000 115000 ...
 $ Planning.to.Buy.a.new.house             : chr  "Yes" "Yes" "Yes" "Yes" ...
 $ Time.Frame                              : chr  "6M to 1Yr" "6M to 1Yr" "<6 Months" "<6 Months" ...
 $ Reasons.for.buying.a.house              : chr  "Residing" "Investment" "Rental Income" "Investment" ...
 $ what.type.of.House                      : chr  "Apartment" "Apartment" "Apartment" "Apartment" ...
 $ Number.of.rooms                         : chr  "2BHK" "2BHK" "1BHK" "3BHK" ...
 $ Size.of.House                           : chr  "1001-1400" "601-1000" "<600" "1401-1800" ...
 $ Budget                                  : chr  "65.1 to 80L" "25.1 to 40L" "<25L" "95.1 to110L" ...
 $ Finished.Semi.Finished                  : chr  "Semifurnished" "Semifurnished" "Semifurnished" "Furnished" ...
 $ Influence.Decision                      : chr  "Site visits" "Newspaper" "Hoarding" "Electronic/Internet" ...
 $ Maintainance                            : chr  "2001to 4000" "<2000" "<2000" "6001 to 8000" ...
 $ EMI                                     : chr  "35.1K to 50K" "20.1K to 35K" "<20K" ">65K" ...
 $ X1.Proximity.to.city                    : int  3 3 1 4 4 3 3 2 4 5 ...
 $ X2.Proximity.to.schools                 : int  5 5 2 5 2 2 3 2 3 2 ...
 $ X3..Proximity.to.transport              : int  5 5 5 3 3 4 4 4 5 4 ...
 $ X4..Proximity.to.work.place             : int  2 3 2 5 4 4 4 3 5 2 ...
 $ X5..Proximity.to.shopping               : int  1 1 1 4 3 2 3 1 1 2 ...
 $ X1..Gym.Pool.Sports.facility            : int  2 1 4 5 2 3 4 1 3 4 ...
 $ X2..Parking.space                       : int  5 4 3 5 4 4 5 2 3 4 ...
 $ X3.Power.back.up                        : int  3 2 2 4 3 4 5 3 3 3 ...
 $ X4.Water.supply                         : int  5 4 4 5 4 4 5 4 4 3 ...
 $ X5.Security                             : int  3 3 5 5 4 3 4 1 3 3 ...
 $ X1..Exterior.look                       : int  2 1 1 4 4 3 4 1 3 4 ...
 $ X2..Unit.size                           : int  4 4 4 4 3 2 3 3 3 3 ...
 $ X3..Interior.design.and.branded.components: int  4 4 3 5 4 4 5 3 3 4 ...
 $ X4..Layout.plan..Integrated.etc..       : int  4 2 2 5 4 3 5 4 3 4 ...
 $ X5..View.from.apartment                 : int  4 2 2 5 4 3 4 1 2 4 ...
 $ X1..Price                               : int  5 5 4 5 4 5 5 5 4 5 ...
 $ X2..Booking.amount                      : int  1 1 2 2 2 2 2 3 2 1 ...
 $ X3..Equated.Monthly.Instalment..EMI.    : int  4 4 5 4 3 4 5 4 4 5 ...
 $ X4..Maintenance.charges                 : int  3 4 4 2 4 3 4 4 3 4 ...
 $ X5..Availability.of.loan                : int  3 4 2 2 4 3 4 3 4 4 ...
 $ X1..Builder.reputation                  : int  4 5 4 5 4 5 5 4 4 5 ...
 $ X2..Appreciation.potential              : int  5 4 4 4 3 4 5 3 4 4 ...
 $ X3..Profile.of.neighbourhood            : int  4 3 4 5 4 4 4 3 3 4 ...
 $ X4..Availability.of.domestic.help       : int  1 2 4 5 3 3 3 2 3 2 ...
 $ Time                                    : int  9 9 3 3 18 3 9 3 18 3 ...
 $ Size                                    : int  1200 800 400 1600 800 800 1600 300 800 1600 ...
 $ Budgets                                 : num  72.5 32.5 12.5 102.5 52.5 ...
 $ Maintainances                           : int  30000 120 10000 70000 30000 30000 50000 10000 30000 50000 ...
 $ EMI.1                                   : int  42500 27500 10000 80000 42500 42500 80000 10000 42500 80000 ...
 $ ages                                    : num  30.5 53 53 40.5 30.5 40.5 53 30.5 40.5 53 ...
 $ sex                                     : chr  "M" "M" "F" "M" ...
 $ Finished.Semi.Finished.1                : chr  "Semifurnished" "Semifurnished" "Semifurnished" "Furnished" ...
 $ Influence.Decision.1                    : chr  "Site visits" "Newspaper" "Hoarding" "Electronic/Internet" ...
> # Step 3: Check for missing values
> sum(is.na(survey_df))
[1] 0
> # Step 4: Select relevant columns for PCA and factor analysis
> # Adjust the column indices based on your dataset
> sur_int <- survey_df[, 20:46]
> str(sur_int)
'data.frame':	70 obs. of  27 variables:
 $ X3..Proximity.to.transport              : int  5 5 5 3 3 4 4 4 5 4 ...
 $ X4..Proximity.to.work.place             : int  2 3 2 5 4 4 4 3 5 2 ...
 $ X5..Proximity.to.shopping               : int  1 1 1 4 3 2 3 1 1 2 ...
 $ X1..Gym.Pool.Sports.facility            : int  2 1 4 5 2 3 4 1 3 4 ...
 $ X2..Parking.space                       : int  5 4 3 5 4 4 5 2 3 4 ...
 $ X3.Power.back.up                        : int  3 2 2 4 3 4 5 3 3 3 ...
 $ X4.Water.supply                         : int  5 4 4 5 4 4 5 4 4 3 ...
 $ X5.Security                             : int  3 3 5 5 4 3 4 1 3 3 ...
 $ X1..Exterior.look                       : int  2 1 1 4 4 3 4 1 3 4 ...
 $ X2..Unit.size                           : int  4 4 4 4 3 2 3 3 3 3 ...
 $ X3..Interior.design.and.branded.components: int  4 4 3 5 4 4 5 3 3 4 ...
 $ X4..Layout.plan..Integrated.etc..       : int  4 2 2 5 4 3 5 4 3 4 ...
 $ X5..View.from.apartment                 : int  4 2 2 5 4 3 4 1 2 4 ...
 $ X1..Price                               : int  5 5 4 5 4 5 5 5 4 5 ...
 $ X2..Booking.amount                      : int  1 1 2 2 2 2 2 3 2 1 ...
 $ X3..Equated.Monthly.Instalment..EMI.    : int  4 4 5 4 3 4 5 4 4 5 ...
 $ X4..Maintenance.charges                 : int  3 4 4 2 4 3 4 4 3 4 ...
 $ X5..Availability.of.loan                : int  3 4 2 2 4 3 4 3 4 4 ...
 $ X1..Builder.reputation                  : int  4 5 4 5 4 5 5 4 4 5 ...
 $ X2..Appreciation.potential              : int  5 4 4 4 3 4 5 3 4 4 ...
 $ X3..Profile.of.neighbourhood            : int  4 3 4 5 4 4 4 3 3 4 ...
 $ X4..Availability.of.domestic.help       : int  1 2 4 5 3 3 3 2 3 2 ...
 $ Time                                    : int  9 9 3 3 18 3 9 3 18 3 ...
 $ Size                                    : int  1200 800 400 1600 800 800 1600 300 800 1600 ...
 $ Budgets                                 : num  72.5 32.5 12.5 102.5 52.5 ...
 $ Maintainances                           : int  30000 120 10000 70000 30000 30000 50000 10000 30000 50000 ...
 $ EMI.1                                   : int  42500 27500 10000 80000 42500 42500 80000 10000 42500 80000 ...
> dim(sur_int)
[1] 70 27
> # Check for multicollinearity
> library(caret) # For findCorrelation function
> cor_matrix <- cor(sur_int, use = "complete.obs")
> high_cor <- findCorrelation(cor_matrix, cutoff = 0.9)
> if (length(high_cor) > 0) {
+   sur_int <- sur_int[ , -high_cor]
+ }
```

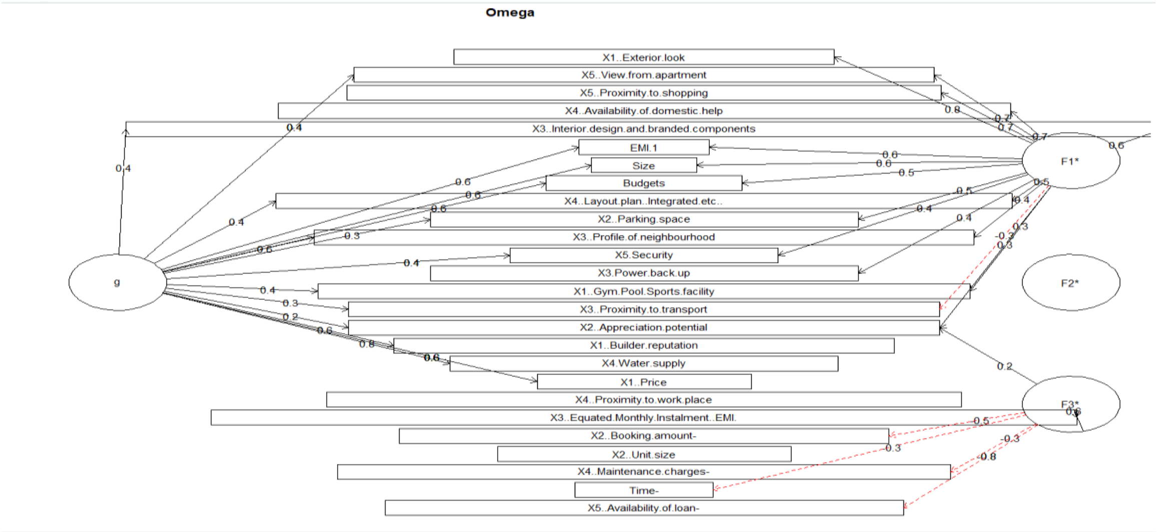


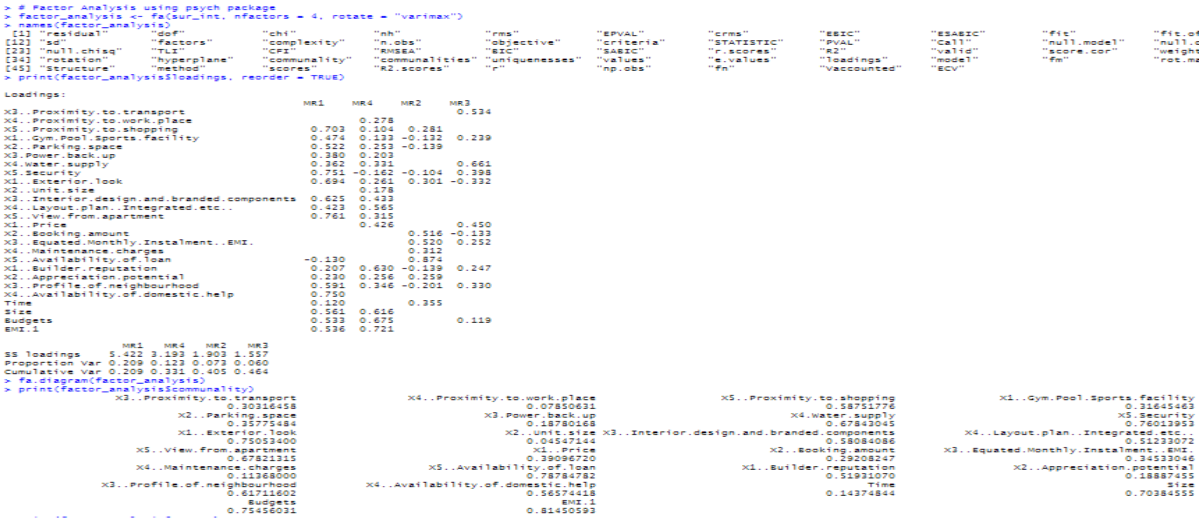
```
> # Step 5: Perform PCA and Factor Analysis
> library(psych)
> library(GPArotation)
> # Perform PCA
> pca <- principal(sur_int, nfactors = 5, n.obs = nrow(survey_df), rotate = "promax", scores = TRUE)
> print(pca)
Principal Components Analysis
Call: principal(r = sur_int, nfactors = 5, rotate = "promax", n.obs = nrow(survey_df),
    scores = TRUE)
Standardized loadings (pattern matrix) based upon correlation matrix
                                               RC1   RC5   RC2   RC4   RC3   h2   u2 com
X3..Proximity.to.transport                   -0.07  0.01  0.10 -0.20  0.77 0.58 0.42 1.2
X4..Proximity.to.work.place                  -0.39  0.19  0.12  0.87 -0.09 0.67 0.33 1.6
X5..Proximity.to.shopping                     0.73 -0.08  0.25  0.17 -0.10 0.66 0.34 1.4
X1..Gym.Pool.Sports.facility                  0.47 -0.02 -0.15  0.19  0.25 0.45 0.55 2.1
X2..Parking.space                             0.52  0.06 -0.16  0.20  0.01 0.46 0.54 1.5
X3.Power.back.up                              0.29 -0.10  0.11  0.69 -0.06 0.64 0.36 1.5
X4.Water.supply                               0.23  0.34  0.01  0.10  0.65 0.72 0.28 1.9
X5.Security                                   0.86 -0.19 -0.18 -0.20  0.36 0.73 0.27 1.7
X1..Exterior.look                             0.71  0.17  0.24 -0.11 -0.35 0.78 0.22 1.9
X2..Unit.size                                -0.08  0.56 -0.16 -0.45 -0.16 0.46 0.54 2.3
X3..Interior.design.and.branded.components    0.51  0.34 -0.04  0.15 -0.08 0.62 0.38 2.0
X4..Layout.plan..Integrated.etc..             0.17  0.53 -0.02  0.30 -0.20 0.61 0.39 2.1
X5..View.from.apartment                       0.74  0.23 -0.05 -0.07 -0.05 0.71 0.29 1.2
X1..Price                                    -0.25  0.62  0.05  0.14  0.48 0.56 0.44 2.4
X2..Booking.amount                            0.09  0.04  0.63 -0.06 -0.12 0.47 0.53 1.2
X3..Equated.Monthly.Instalment..EMI.         -0.08 -0.01  0.67  0.00  0.42 0.53 0.47 1.7
X4..Maintenance.charges                      -0.03 -0.06  0.45 -0.06  0.00 0.23 0.77 1.1
X5..Availability.of.loan                     -0.13 -0.07  0.88  0.24  0.00 0.76 0.24 1.2
X1..Builder.reputation                       -0.10  0.86 -0.07 -0.08  0.18 0.70 0.30 1.1
X2..Appreciation.potential                    0.13  0.43  0.40 -0.16  0.08 0.38 0.62 2.6
X3..Profile.of.neighbourhood                  0.50  0.40 -0.20 -0.14  0.27 0.67 0.33 3.1
X4..Availability.of.domestic.help             0.91 -0.02 -0.06 -0.39 -0.09 0.72 0.28 1.4
Time                                          0.27 -0.13  0.44 -0.10  0.17 0.28 0.72 2.4
Size                                          0.39  0.58  0.06  0.06  0.03 0.72 0.28 1.8
Budgets                                       0.35  0.64  0.03  0.04  0.05 0.75 0.25 1.6
EMI.1                                         0.31  0.61 -0.03  0.21 -0.02 0.79 0.21 1.8

                       RC1  RC5  RC2  RC4  RC3
SS loadings           5.32 4.04 2.41 1.93 1.93
Proportion Var        0.20 0.16 0.09 0.07 0.07
Cumulative Var        0.20 0.36 0.45 0.53 0.60
Proportion Explained  0.34 0.26 0.15 0.12 0.12
Cumulative Proportion 0.34 0.60 0.75 0.88 1.00

 With component correlations of 
     RC1   RC5   RC2   RC4   RC3
RC1 1.00  0.45  0.00  0.36  0.06
RC5 0.45  1.00 -0.08  0.19  0.00
RC2 0.00 -0.08  1.00 -0.20 -0.19
RC4 0.36  0.19 -0.20  1.00  0.15
RC3 0.06  0.00 -0.19  0.15  1.00

Mean item complexity =  1.8
Test of the hypothesis that 5 components are sufficient.

The root mean square of the residuals (RMSR) is  0.07 
 with the empirical chi square  235.32  with prob <  0.072 

Fit based upon off diagonal values = 0.94
```



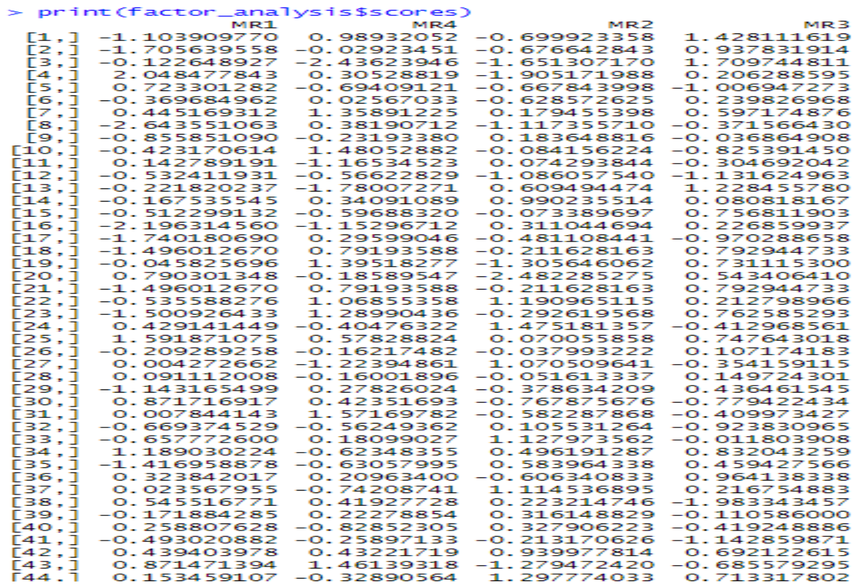
```
> # Omega hierarchical analysis with alternative factor score estimation method
> om.h <- omega(sur_int, n.obs = nrow(survey_df), sl = FALSE, fm = "minres")
> om <- omega(sur_int, n.obs = nrow(survey_df), fm = "minres")
```


Omega
X1..Exterior.look
X5..View.from.apartment
X5..Proximity.to.shopping
X4..Availability.of.domestic.help
X3..Interior.design.and.branded.components
EMI.1
Size
Budgets
X4..Layout.plan..Integrated.etc..
X2..Parking.space
X3..Profile.of.neighbourhood
X5.Security
X3.Power.back.up
X1..Gym.Pool.Sports.facility
X2..Appreciation.potential
X4.Water.supply
X1..Price
X1..Builder.reputation
X3..Proximity.to.transport
X4..Proximity.to.work.place
X5..Availability.of.loan
X3..Equated.Monthly.Instalment..EMI.
X2..Booking.amount
Time
X4..Maintenance.charges
X2..Unit.size
F1
F2
F3
g
0.4
1
-0.2

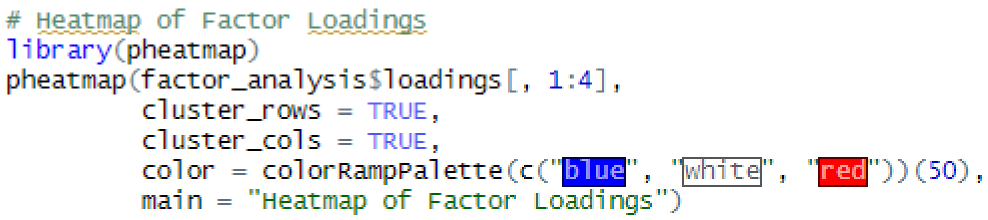

Omega
X1..Exterior.look
X5..View.from.apartment
X5..Proximity.to.shopping
X4..Availability.of.domestic.help
X3..Interior.design.and.branded.components
EMI.1
Size
Budgets
X4..Layout.plan..Integrated.etc..
X2..Parking.space
X3..Profile.of.neighbourhood
X5.Security
X3.Power.back.up
X1..Gym.Pool.Sports.facility
X3..Proximity.to.transport
X2..Appreciation.potential
X1..Builder.reputation
X4.Water.supply
X1..Price
X4..Proximity.to.work.place
X3..Equated.Monthly.Instalment..EMI.
X2..Booking.amount-
X2..Unit.size
X4..Maintenance.charges-
Time-
X5..Availability.of.loan-
g
F1*
F2*
F3*



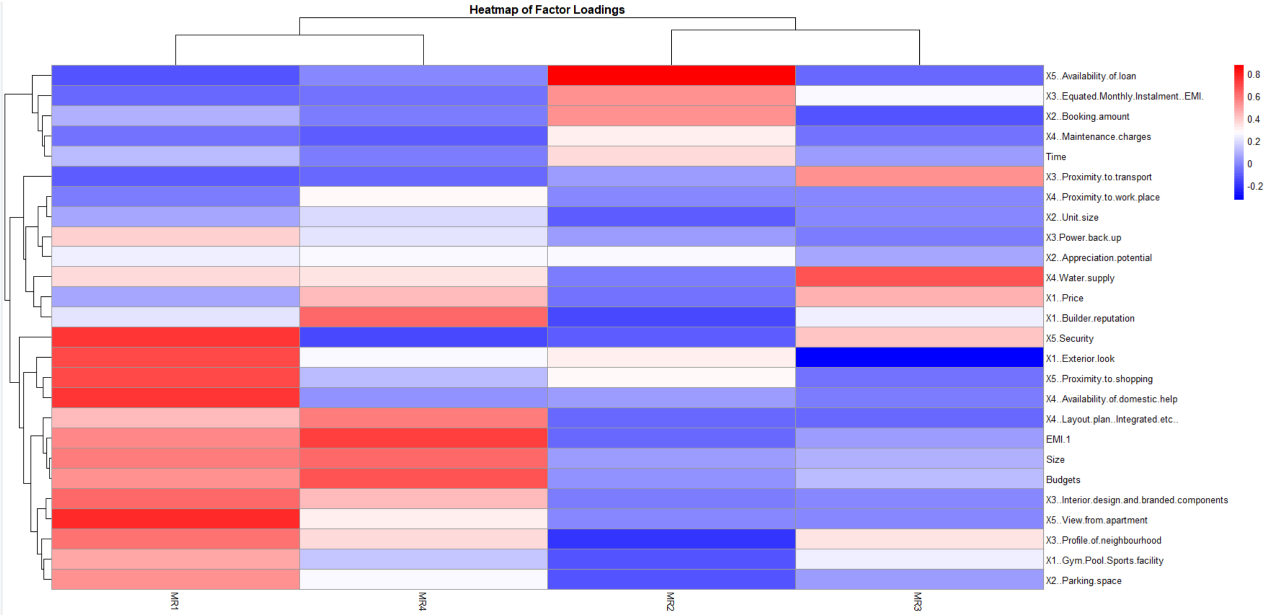
```
> # Factor Analysis using psych package
> factor_analysis <- fa(sur_int, nfactors = 4, rotate = "varimax")
> names(factor_analysis)
 [1] "residual"      "dof"           "chi"           "nh"            "rms"           "EPVAL"         "crms"          "EBIC"          "ESABIC"        "fit"           "fit.of
[12] "sd"            "factors"       "complexity"    "n.obs"         "objective"     "criteria"      "STATISTIC"     "PVAL"          "Call"          "null.model"    "null.c
[23] "null.chisq"    "TLI"           "CFI"           "RMSEA"         "BIC"           "SABIC"         "r.scores"      "R2"            "valid"         "score.cor"     "weight
[34] "rotation"      "hyperplane"    "communality"   "communalities" "uniquenesses"  "values"        "e.values"      "loadings"      "model"         "fm"            "rot.ma
[45] "Structure"     "method"        "scores"        "R2.scores"     "r"             "np.obs"        "fn"            "Vaccounted"    "ECV"
> print(factor_analysis$loadings, reorder = TRUE)

Loadings:
                                           MR1    MR4    MR2    MR3
X3..Proximity.to.transport                                      0.534
X4..Proximity.to.work.place                        0.278
X5..Proximity.to.shopping                  0.703  0.104  0.281
X1..Gym.Pool.Sports.facility               0.474  0.133 -0.132  0.239
X2..Parking.space                          0.522  0.253 -0.139
X3.Power.back.up                           0.380  0.203
X4.Water.supply                            0.362  0.331         0.661
X5.Security                                0.751 -0.162 -0.104  0.398
X1..Exterior.look                          0.694  0.261  0.301 -0.332
X2..Unit.size                                     0.178
X3..Interior.design.and.branded.components 0.625  0.433
X4..Layout.plan..Integrated.etc..          0.423  0.565
X5..View.from.apartment                    0.761  0.315
X1..Price                                         0.426         0.450
X2..Booking.amount                                       0.516 -0.133
X3..Equated.Monthly.Instalment..EMI.                     0.520  0.252
X4..Maintenance.charges                                  0.312
X5..Availability.of.loan                  -0.130         0.874
X1..Builder.reputation                     0.207  0.630 -0.139  0.247
X2..Appreciation.potential                 0.230  0.256  0.259
X3..Profile.of.neighbourhood               0.591  0.346 -0.201  0.330
X4..Availability.of.domestic.help          0.750
Time                                       0.120         0.355
Size                                       0.561  0.616
Budgets                                    0.533  0.675         0.119
EMI.1                                      0.536  0.721

                 MR1   MR4   MR2   MR3
SS loadings    5.422 3.193 1.903 1.557
Proportion Var 0.209 0.123 0.073 0.060
Cumulative Var 0.209 0.331 0.405 0.464
> fa.diagram(factor_analysis)
> print(factor_analysis$communality)
                X3..Proximity.to.transport                X4..Proximity.to.work.place                  X5..Proximity.to.shopping               X1..Gym.Pool.Sports.facility
                                0.30316458                                 0.07850631                                 0.58751776                                 0.31645463
                         X2..Parking.space                           X3.Power.back.up                            X4.Water.supply                                X5.Security
                                0.35775484                                 0.18780168                                 0.67843045                                 0.76013953
                         X1..Exterior.look                              X2..Unit.size X3..Interior.design.and.branded.components          X4..Layout.plan..Integrated.etc..
                                0.75053400                                 0.04547144                                 0.58084086                                 0.51233072
                   X5..View.from.apartment                                  X1..Price                         X2..Booking.amount       X3..Equated.Monthly.Instalment..EMI.
                                0.67821315                                 0.39096720                                 0.29208247                                 0.34533046
                   X4..Maintenance.charges                   X5..Availability.of.loan                     X1..Builder.reputation                 X2..Appreciation.potential
                                0.11368000                                 0.78784782                                 0.51931070                                 0.18887455
              X3..Profile.of.neighbourhood          X4..Availability.of.domestic.help                                       Time                                       Size
                                0.61711602                                 0.56574418                                 0.14374844                                 0.70384555
                                   Budgets                                      EMI.1
                                0.75456031                                 0.81450593
```

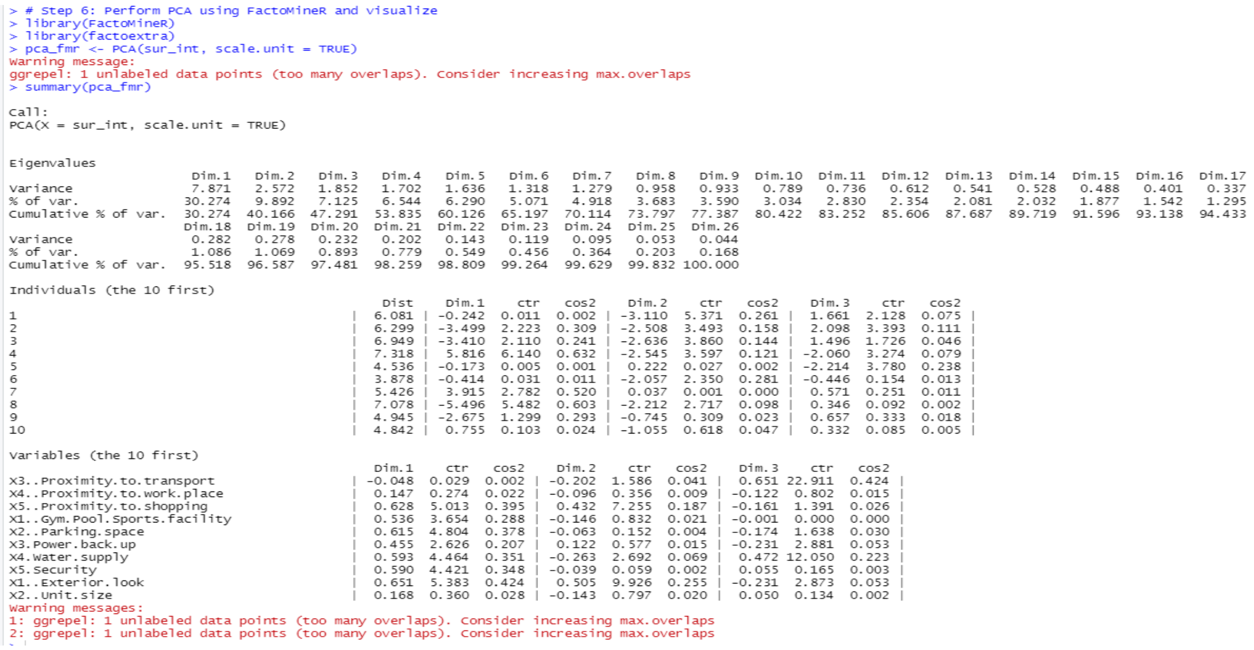


```
> print(factor_analysis$scores)
               MR1         MR4          MR2          MR3
 [1,] -1.103909770  0.98932052 -0.699923358  1.428111619
 [2,] -1.705639558 -0.02923451 -0.676642843  0.937831914
 [3,] -0.122648927 -2.43623946 -1.651307170  1.709744811
 [4,]  2.048477843  0.30528819 -1.905171988  0.206288595
 [5,]  0.723301282 -0.69409121 -0.667843998 -1.006947273
 [6,] -0.369684962  0.02567033 -0.628572625  0.239826968
 [7,]  0.445169312  1.35891225  0.179455398  0.597174876
 [8,] -2.643551063  0.38190712 -1.117355710 -0.371566430
 [9,] -0.855851090 -0.23193380  0.183648816 -0.036864908
[10,] -0.423170614  1.48052882 -0.084156224 -0.825391450
[11,]  0.142789191 -1.16534523  0.074293844 -0.304692042
[12,] -0.532411931 -0.56622829 -1.086057540 -1.131624963
[13,] -0.221820237 -1.78007271  0.609494474  1.228455780
[14,] -0.167535545  0.34091089  0.990235514  0.080818167
[15,] -0.512299132 -0.59688320 -0.073389697  0.756811903
[16,] -2.196314560 -1.15296712  0.311044694  0.226859937
[17,] -1.740180690  0.29599046 -0.481108441 -0.970288658
[18,] -1.496012670  0.79193588 -0.211628163  0.792944733
[19,] -0.045825696  1.39518277 -1.305646062  0.731115300
[20,]  0.790301348 -0.18589547 -2.482285275  0.543406410
[21,] -1.496012670  0.79193588 -0.211628163  0.792944733
[22,] -0.535588276  1.06855358  1.190965115  0.212798966
[23,] -1.500926433  1.28990436 -0.292619568  0.762585293
[24,]  0.429141449 -0.40476322  1.475181357 -0.412968561
[25,]  1.591871075  0.57828824  0.070055858  0.747643018
[26,] -0.209289258 -0.16217482 -0.037993222  0.107174183
[27,]  0.004272662 -1.22394861  1.070509641 -0.354159115
[28,]  0.091112008 -0.16001896 -0.051613337  0.149724301
[29,] -1.143165499  0.27826024 -0.378634209  0.436461545
[30,]  0.871716917  0.42351693 -0.767875676 -0.779422434
[31,]  0.007844143  1.57169782 -0.582287868 -0.409973427
[32,] -0.669374529 -0.56249362  0.105531264 -0.923830965
[33,] -0.657772600  0.18099027  1.127973562 -0.011803908
[34,]  1.189030224 -0.62348355  0.496191287  0.832043259
[35,] -1.416958878 -0.63057995  0.583964338  0.459427566
[36,]  0.323842017  0.20963400 -0.606340833  0.964138338
[37,]  0.023567955 -0.74208741  1.114536895  0.216754883
[38,]  0.545516771  0.41927728  0.223214746 -1.983343457
[39,] -0.171884285  0.22278854  0.316148829 -0.110586000
[40,]  0.258807628 -0.82852305  0.327906223 -0.419248886
[41,] -0.493020882 -0.25897133 -0.213170626 -1.142859871
[42,]  0.439403978  0.43221719  0.939977814  0.692122615
[43,]  0.871471394  1.46139318 -1.279472420 -0.685579295
[44,]  0.153459107 -0.32890564  1.297774033  0.713317802
```

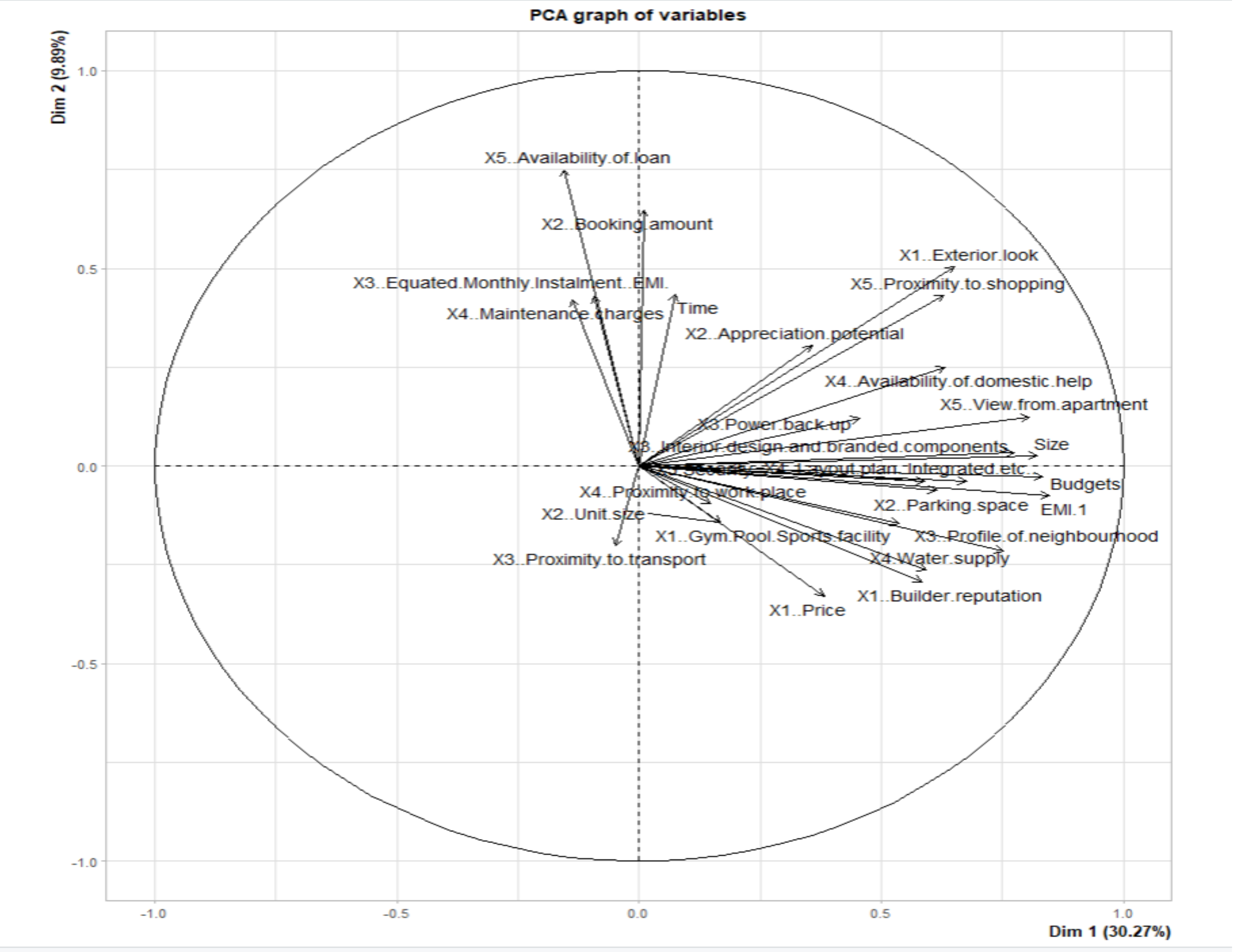


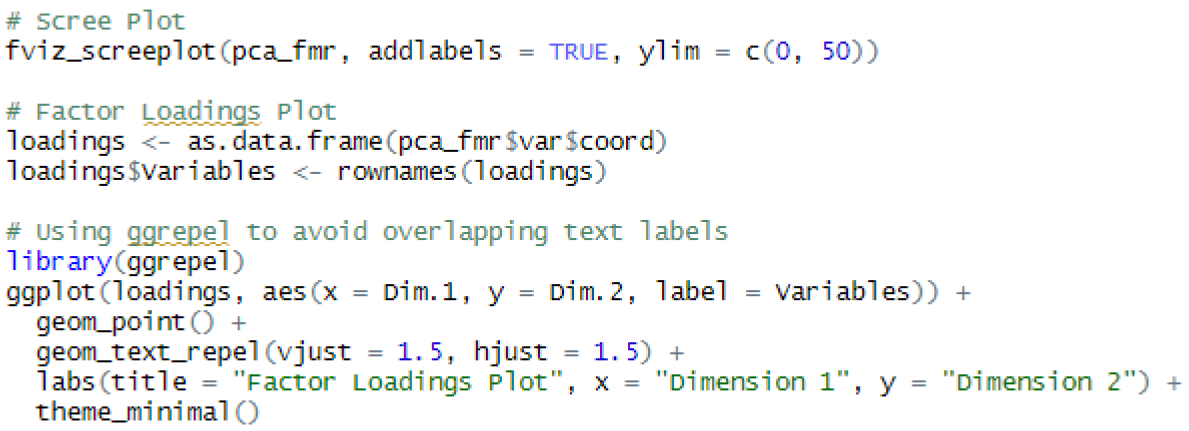
```
# Heatmap of Factor Loadings
library(pheatmap)
pheatmap(factor_analysis$loadings[, 1:4],
         cluster_rows = TRUE,
         cluster_cols = TRUE,
         color = colorRampPalette(c("blue", "white", "red"))(50),
         main = "Heatmap of Factor Loadings")
```



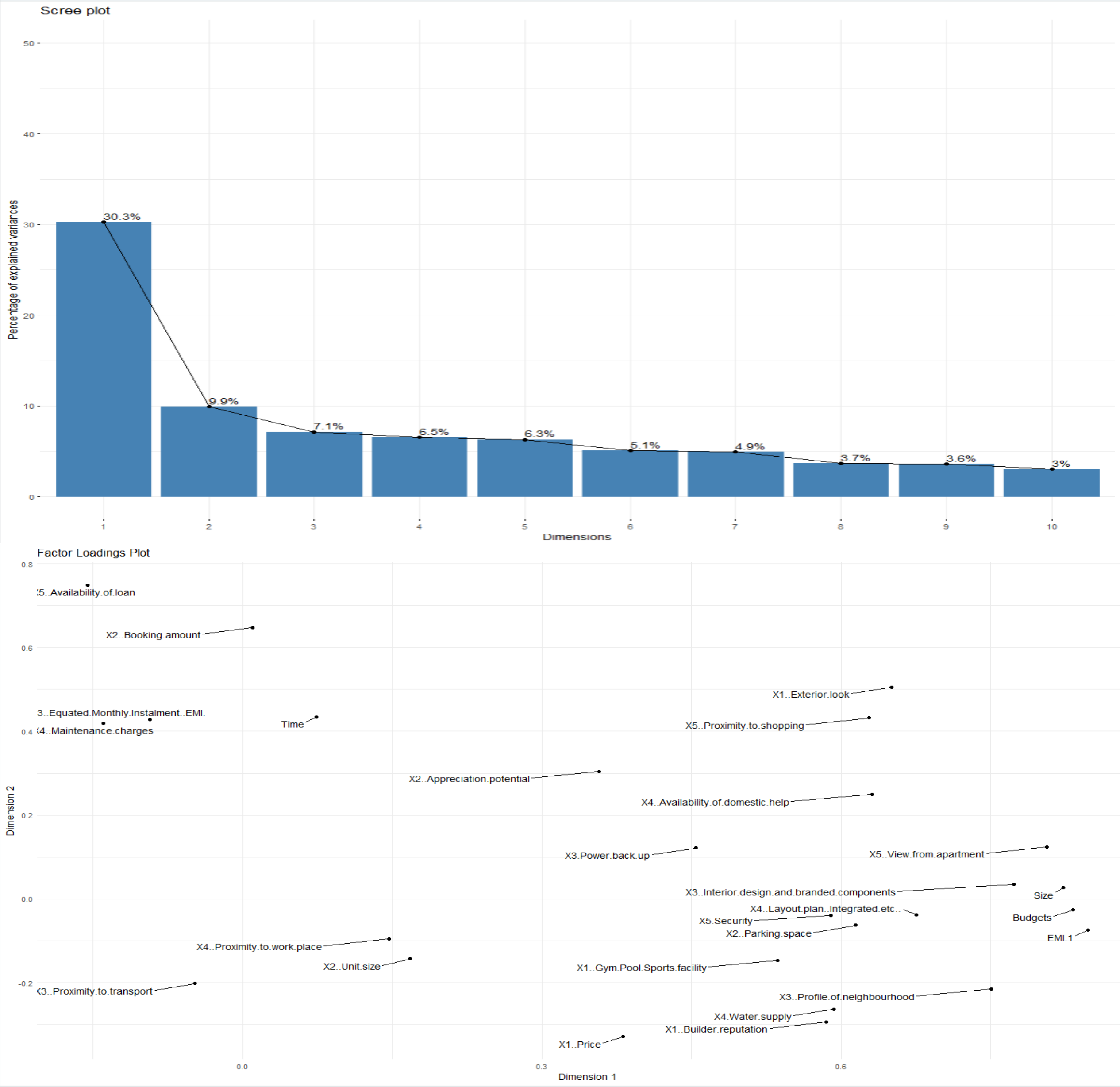


```
> # Step 6: Perform PCA using FactoMineR and visualize
> library(FactoMineR)
> library(factoextra)
> pca_fmr <- PCA(sur_int, scale.unit = TRUE)
Warning message:
ggrepel: 1 unlabeled data points (too many overlaps). Consider increasing max.overlaps
> summary(pca_fmr)

Call:
PCA(X = sur_int, scale.unit = TRUE)


Eigenvalues
                       Dim.1   Dim.2   Dim.3   Dim.4   Dim.5   Dim.6   Dim.7   Dim.8   Dim.9  Dim.10  Dim.11  Dim.12  Dim.13  Dim.14  Dim.15  Dim.16  Dim.17
Variance               7.871   2.572   1.852   1.702   1.636   1.318   1.279   0.958   0.933   0.789   0.736   0.612   0.541   0.528   0.488   0.401   0.337
% of var.             30.274   9.892   7.125   6.544   6.290   5.071   4.918   3.683   3.590   3.034   2.830   2.354   2.081   2.032   1.877   1.542   1.295
Cumulative % of var.  30.274  40.166  47.291  53.835  60.126  65.197  70.114  73.797  77.387  80.422  83.252  85.606  87.687  89.719  91.596  93.138  94.433
                      Dim.18  Dim.19  Dim.20  Dim.21  Dim.22  Dim.23  Dim.24  Dim.25  Dim.26
Variance               0.282   0.278   0.232   0.202   0.143   0.119   0.095   0.053   0.044
% of var.              1.086   1.069   0.893   0.779   0.549   0.456   0.364   0.203   0.168
Cumulative % of var.  95.518  96.587  97.481  98.259  98.809  99.264  99.629  99.832 100.000

Individuals (the 10 first)
                                 Dist    Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr   cos2  
1                            |  6.081 | -0.242  0.011  0.002 | -3.110  5.371  0.261 |  1.661  2.128  0.075 |
2                            |  6.299 | -3.499  2.223  0.309 | -2.508  3.493  0.158 |  2.098  3.393  0.111 |
3                            |  6.949 | -3.410  2.110  0.241 | -2.636  3.860  0.144 |  1.496  1.726  0.046 |
4                            |  7.318 |  5.816  6.140  0.632 | -2.545  3.597  0.121 | -2.060  3.274  0.079 |
5                            |  4.536 | -0.173  0.005  0.001 |  0.222  0.027  0.002 | -2.214  3.780  0.238 |
6                            |  3.878 | -0.414  0.031  0.011 | -2.057  2.350  0.281 | -0.446  0.154  0.013 |
7                            |  5.426 |  3.915  2.782  0.520 |  0.037  0.001  0.000 |  0.571  0.251  0.011 |
8                            |  7.078 | -5.496  5.482  0.603 | -2.212  2.717  0.098 |  0.346  0.092  0.002 |
9                            |  4.945 | -2.675  1.299  0.293 | -0.745  0.309  0.023 |  0.657  0.333  0.018 |
10                           |  4.842 |  0.755  0.103  0.024 | -1.055  0.618  0.047 |  0.332  0.085  0.005 |

Variables (the 10 first)
                                Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr   cos2  
X3..Proximity.to.transport   | -0.048  0.029  0.002 | -0.202  1.586  0.041 |  0.651 22.911  0.424 |
X4..Proximity.to.work.place  |  0.147  0.274  0.022 | -0.096  0.356  0.009 | -0.122  0.802  0.015 |
X5..Proximity.to.shopping    |  0.628  5.013  0.395 |  0.432  7.255  0.187 | -0.161  1.391  0.026 |
X1..Gym.Pool.Sports.facility |  0.536  3.654  0.288 | -0.146  0.832  0.021 | -0.001  0.000  0.000 |
X2..Parking.space            |  0.615  4.804  0.378 | -0.063  0.152  0.004 | -0.174  1.638  0.030 |
X3.Power.back.up             |  0.455  2.626  0.207 |  0.122  0.577  0.015 | -0.231  2.881  0.053 |
X4.Water.supply              |  0.593  4.464  0.351 | -0.263  2.692  0.069 |  0.472 12.050  0.223 |
X5.Security                  |  0.590  4.421  0.348 | -0.039  0.059  0.002 |  0.055  0.165  0.003 |
X1..Exterior.look            |  0.651  5.383  0.424 |  0.505  9.926  0.255 | -0.231  2.873  0.053 |
X2..Unit.size                |  0.168  0.360  0.028 | -0.143  0.797  0.020 |  0.050  0.134  0.002 |
Warning messages:
1: ggrepel: 1 unlabeled data points (too many overlaps). Consider increasing max.overlaps
2: ggrepel: 1 unlabeled data points (too many overlaps). Consider increasing max.overlaps
```





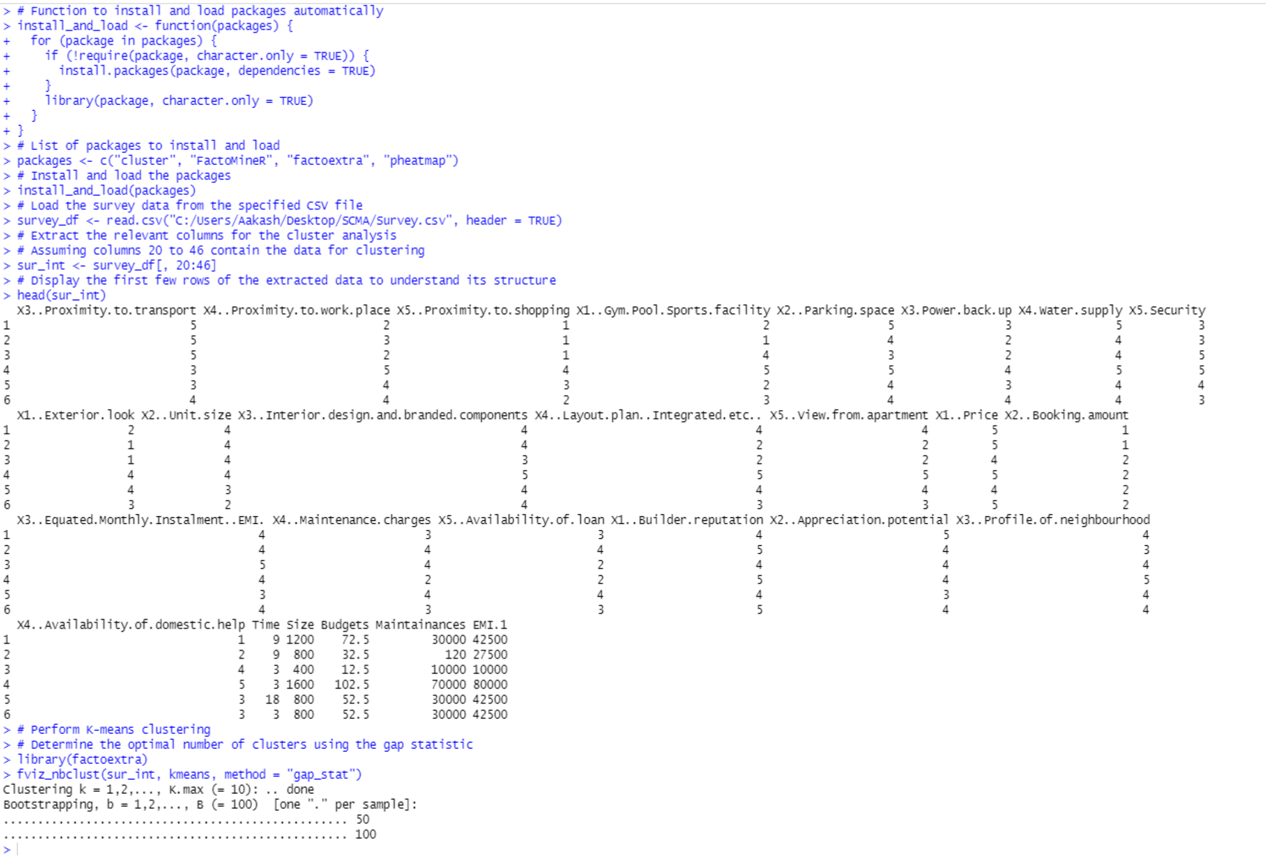
```
# Scree Plot
fviz_screeplot(pca_fmr, addlabels = TRUE, ylim = c(0, 50))

# Factor Loadings Plot
loadings <- as.data.frame(pca_fmr$var$coord)
loadings$variables <- rownames(loadings)

# Using ggrepel to avoid overlapping text labels
library(ggrepel)
ggplot(loadings, aes(x = Dim.1, y = Dim.2, label = variables)) +
  geom_point() +
  geom_text_repel(vjust = 1.5, hjust = 1.5) +
  labs(title = "Factor Loadings Plot", x = "Dimension 1", y = "Dimension 2") +
  theme_minimal()
```



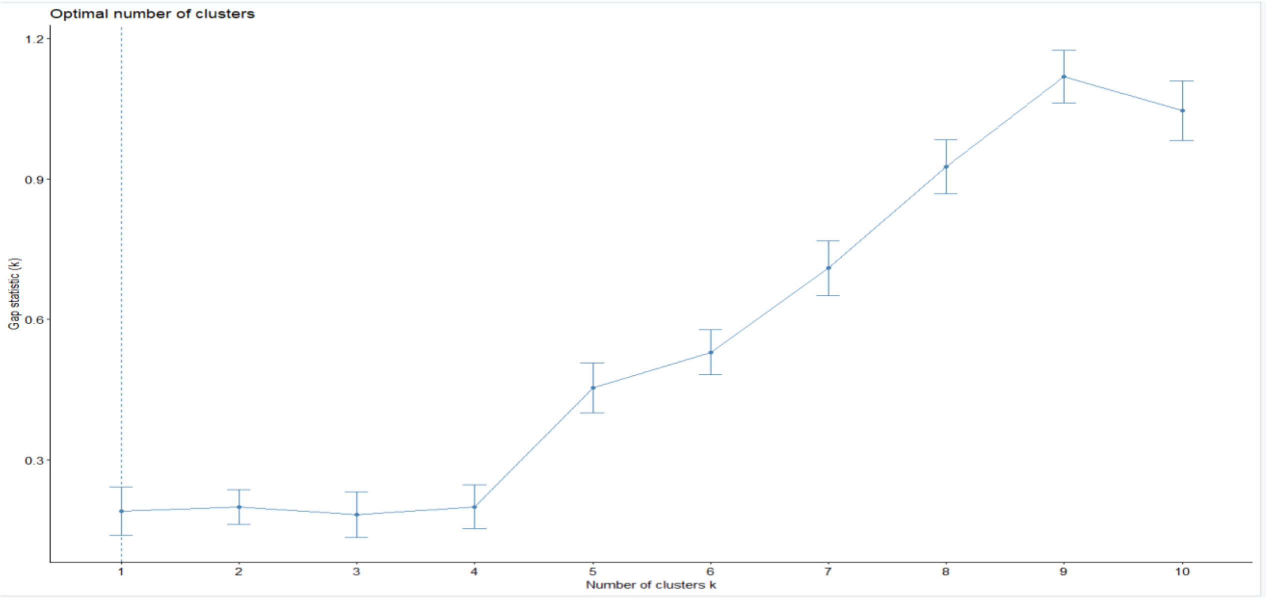
Interpretation:

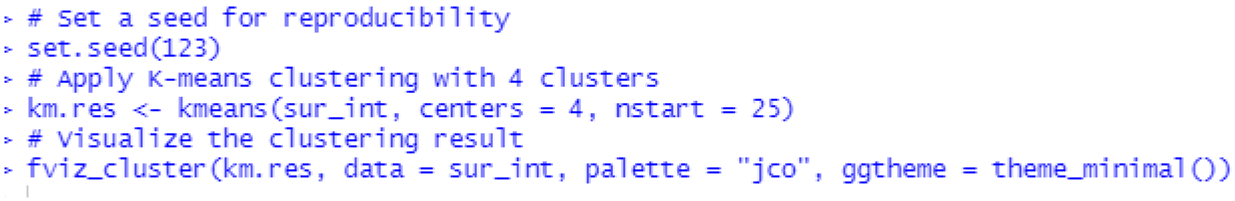
The Principal Components Analysis (PCA) conducted on the selected dataset reveals key patterns by extracting five principal components that capture various aspects of the data. The analysis shows that proximity to shopping and security are highly influential, with significant contributions from price, booking amount, and availability of loans. Specifically, proximity to shopping and the security component load heavily on component 1, while builder reputation and appreciation potential are prominent in component 2. The PCA effectively distills the dataset into a manageable number of dimensions while retaining critical information. To assess the model fit, an off-diagonal value of 0.94 was achieved, and omega hierarchical analysis, using both default and alternative factor score estimation methods from the psych package, was performed to evaluate the hierarchical structure. Factor analysis with a varimax rotation identified four factors (MR1 to MR4), revealing significant loadings from attributes like proximity to transportation, security, and price. Factor analysis results, including communalities and factor scores, were visualized using a heatmap to illustrate variable relationships. Subsequently, PCA using the FactoMineR package detailed eigenvalues, variance explained, and included a scree plot and biplot. The biplot showed the distribution of variables and individuals in the reduced dimensional space, with factor loadings plotted using ggrepel to manage label overlaps.

2.Cluster Analysis

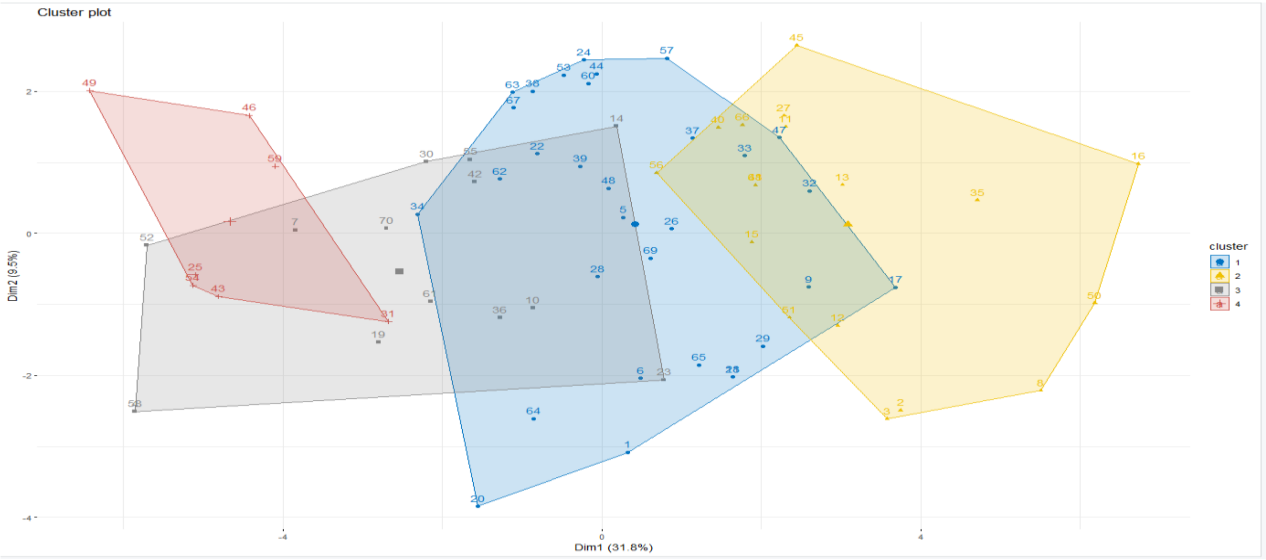


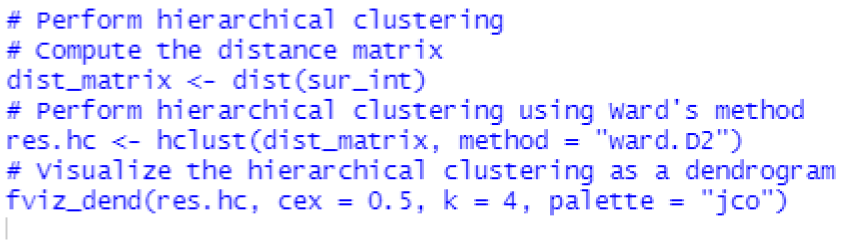
```
> # Function to install and load packages automatically
> install_and_load <- function(packages) {
+   for (package in packages) {
+     if (!require(package, character.only = TRUE)) {
+       install.packages(package, dependencies = TRUE)
+     }
+     library(package, character.only = TRUE)
+   }
+ }
> # List of packages to install and load
> packages <- c("cluster", "FactoMineR", "factoextra", "pheatmap")
> # Install and load the packages
> install_and_load(packages)
> # Load the survey data from the specified CSV file
> survey_df <- read.csv("C:/Users/Aakash/Desktop/SCMA/Survey.csv", header = TRUE)
> # Extract the relevant columns for the cluster analysis
> # Assuming columns 20 to 46 contain the data for clustering
> sur_int <- survey_df[, 20:46]
> # Display the first few rows of the extracted data to understand its structure
> head(sur_int)
  X3..Proximity.to.transport X4..Proximity.to.work.place X5..Proximity.to.shopping X1..Gym.Pool.Sports.facility X2..Parking.space X3.Power.back.up X4.Water.supply X5.Security
1                          5                           2                         1                            2                 5                3               5           3
2                          5                           3                         1                            1                 4                2               4           3
3                          5                           2                         1                            4                 3                2               4           5
4                          3                           5                         4                            5                 5                4               5           5
5                          3                           4                         3                            2                 4                3               4           4
6                          4                           4                         2                            3                 4                4               4           3
  X1..Exterior.look X2..Unit.size X3..Interior.design.and.branded.components X4..Layout.plan..Integrated.etc.. X5..View.from.apartment X1..Price X2..Booking.amount
1                 2             4                                          4                                 4                       4         5                  1
2                 1             4                                          4                                 2                       2         5                  1
3                 1             4                                          3                                 2                       2         4                  2
4                 4             4                                          5                                 5                       5         5                  2
5                 4             3                                          4                                 4                       4         4                  2
6                 3             2                                          4                                 3                       3         5                  2
  X3..Equated.Monthly.Instalment..EMI. X4..Maintenance.charges X5..Availability.of.loan X1..Builder.reputation X2..Appreciation.potential X3..Profile.of.neighbourhood
1                                    4                       3                        3                      4                          5                            4
2                                    4                       4                        4                      5                          4                            3
3                                    5                       4                        2                      4                          4                            4
4                                    4                       2                        2                      5                          4                            5
5                                    3                       4                        4                      4                          3                            4
6                                    4                       3                        3                      5                          4                            4
  X4..Availability.of.domestic.help Time Size Budgets Maintainances EMI.1
1                                 1    9 1200    72.5         30000 42500
2                                 2    9  800    32.5           120 27500
3                                 4    3  400    12.5         10000 10000
4                                 5    3 1600   102.5         70000 80000
5                                 3   18  800    52.5         30000 42500
6                                 3    3  800    52.5         30000 42500
> # Perform K-means clustering
> # Determine the optimal number of clusters using the gap statistic
> library(factoextra)
> fviz_nbclust(sur_int, kmeans, method = "gap_stat")
Clustering k = 1,2,..., K.max (= 10): .. done
Bootstrapping, b = 1,2,..., B (= 100)  [one "." per sample]:
.................................................. 50 
.................................................. 100 
> 
```



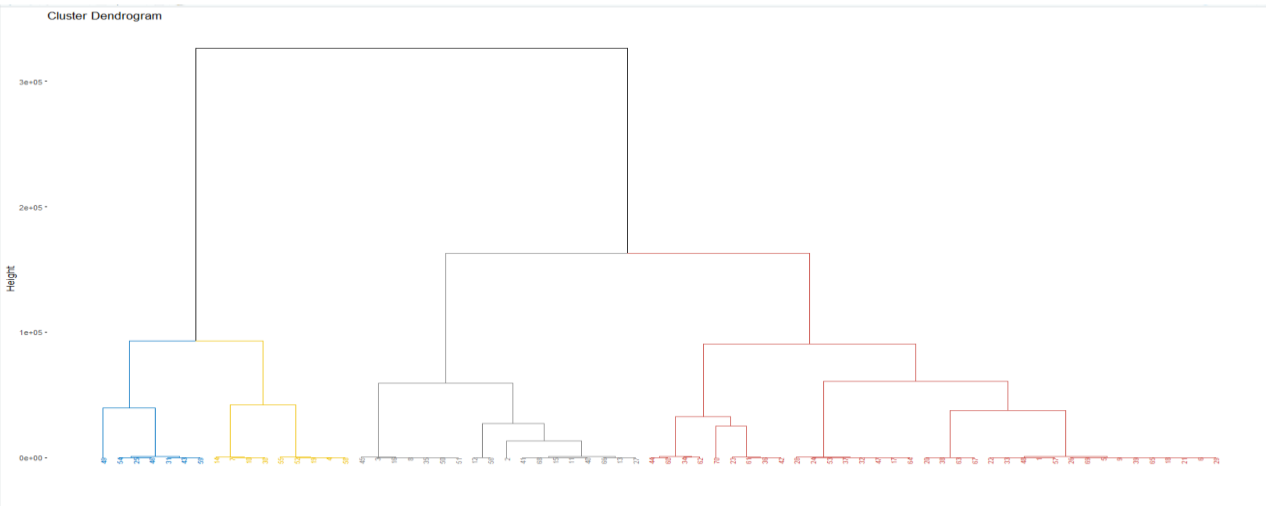


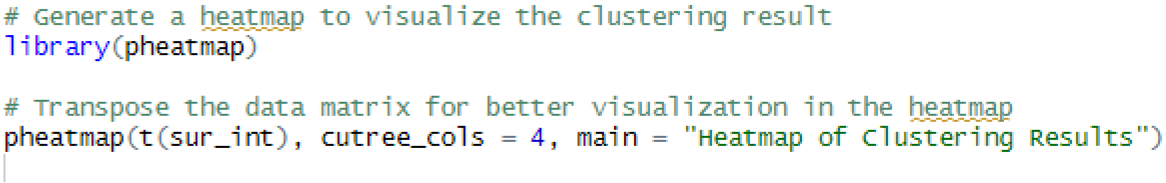
```
> # Set a seed for reproducibility
> set.seed(123)
> # Apply K-means clustering with 4 clusters
> km.res <- kmeans(sur_int, centers = 4, nstart = 25)
> # Visualize the clustering result
> fviz_cluster(km.res, data = sur_int, palette = "jco", ggtheme = theme_minimal())
```



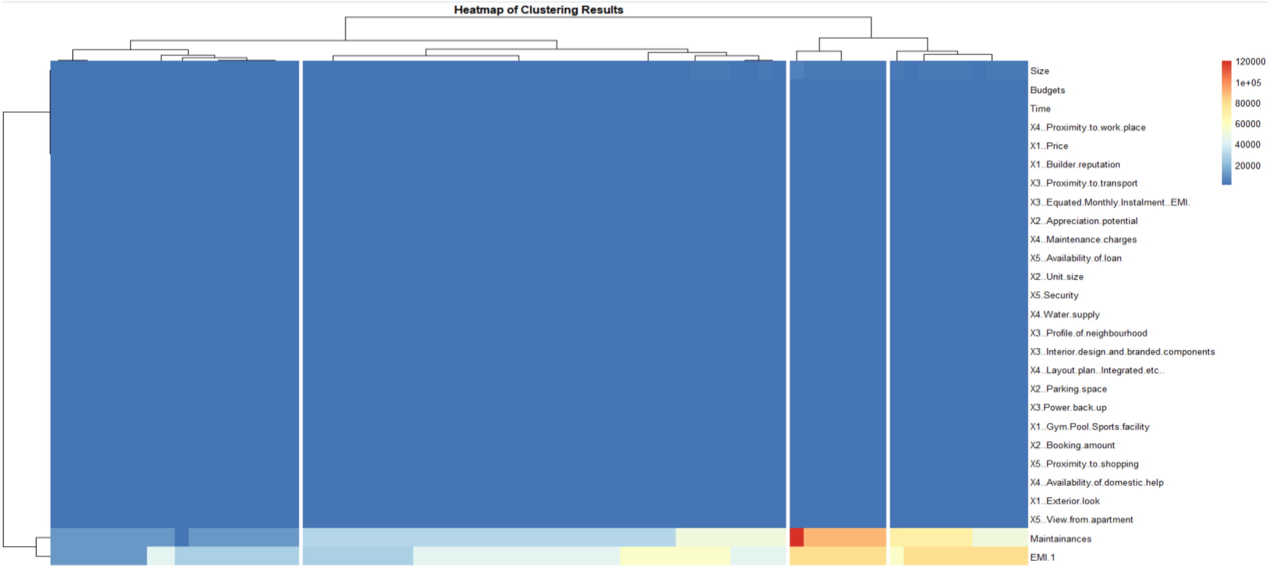


```
# Perform hierarchical clustering
# Compute the distance matrix
dist_matrix <- dist(sur_int)
# Perform hierarchical clustering using Ward's method
res.hc <- hclust(dist_matrix, method = "ward.D2")
# Visualize the hierarchical clustering as a dendrogram
fviz_dend(res.hc, cex = 0.5, k = 4, palette = "jco")
```





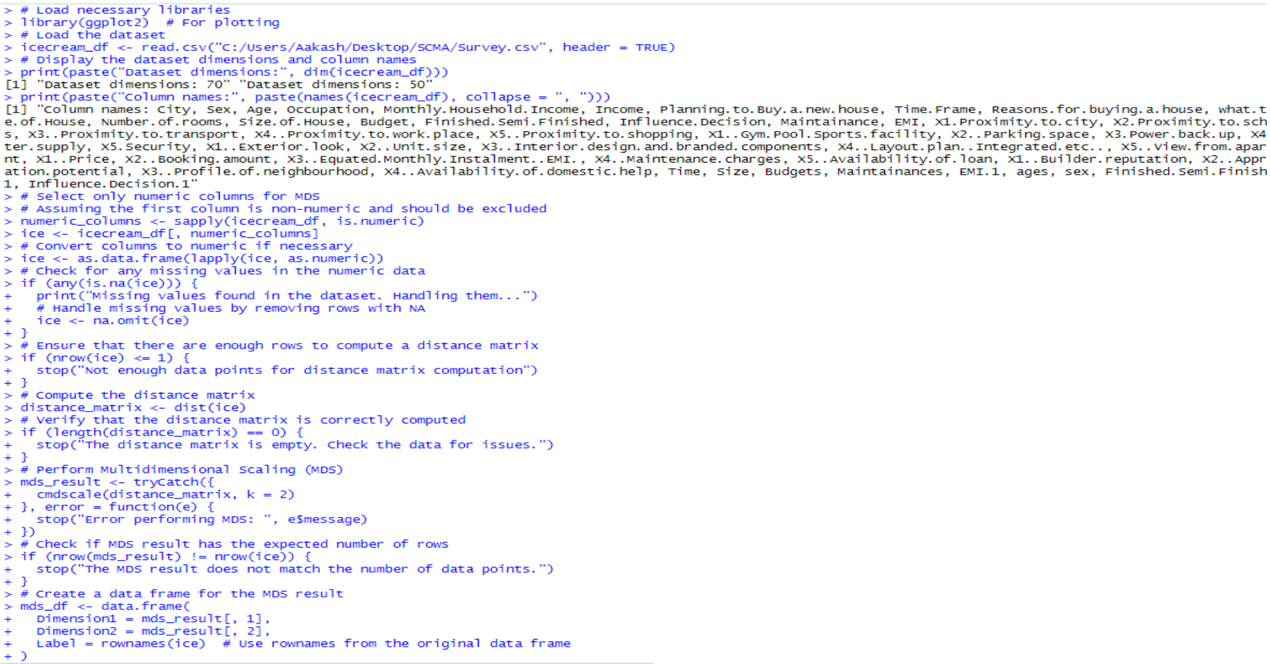
```
# Generate a heatmap to visualize the clustering result
library(pheatmap)

# Transpose the data matrix for better visualization in the heatmap
pheatmap(t(sur_int), cutree_cols = 4, main = "Heatmap of Clustering Results")
```



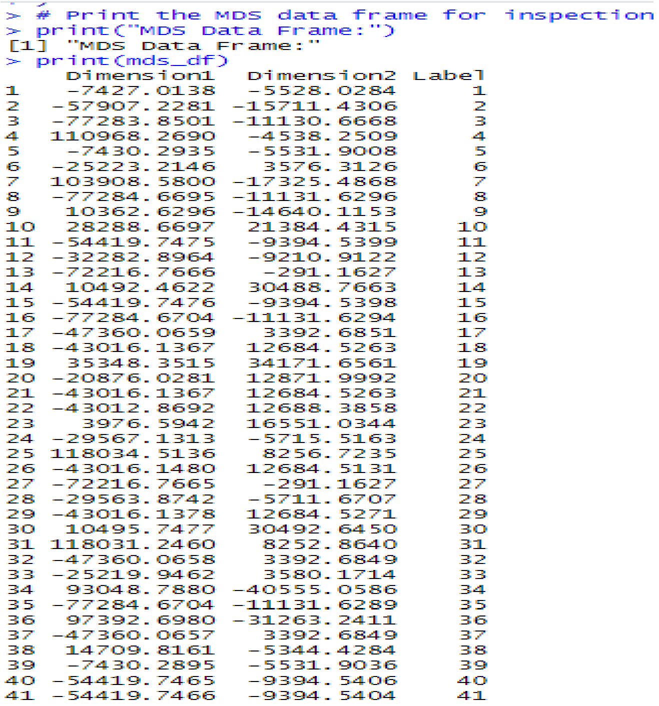
Interpretation:

The clustering analysis on your survey data involved several steps to uncover patterns and groupings within the dataset. Initially, a subset of columns relevant to housing features was extracted for analysis. K-means clustering, optimized to 4 clusters using the gap statistic method, revealed distinct groupings among the data points, with the results visualized to show cluster distributions. Hierarchical clustering, performed with Ward's method, was used to create a dendrogram illustrating the hierarchical structure of the clusters, which aligned with the K-means results. Additionally, a heatmap was generated to visualize the clustering results in a matrix format, revealing patterns and relationships between attributes and samples. Collectively, these methods provided a comprehensive view of the data’s underlying structure, identifying key groupings and relationships that offer insights into the survey data.

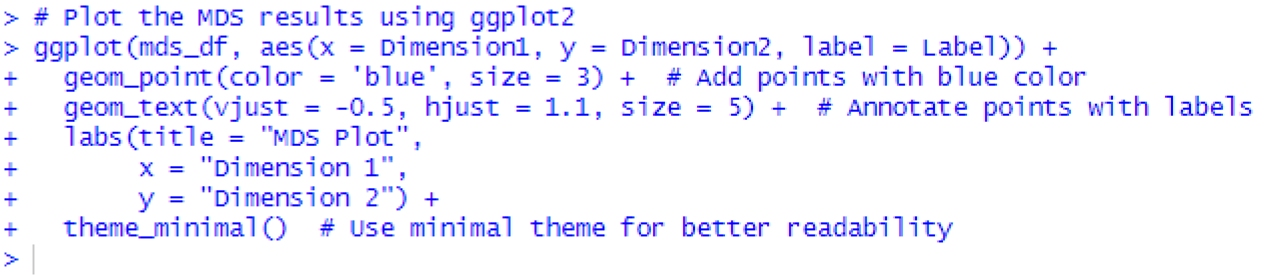
3.MDS



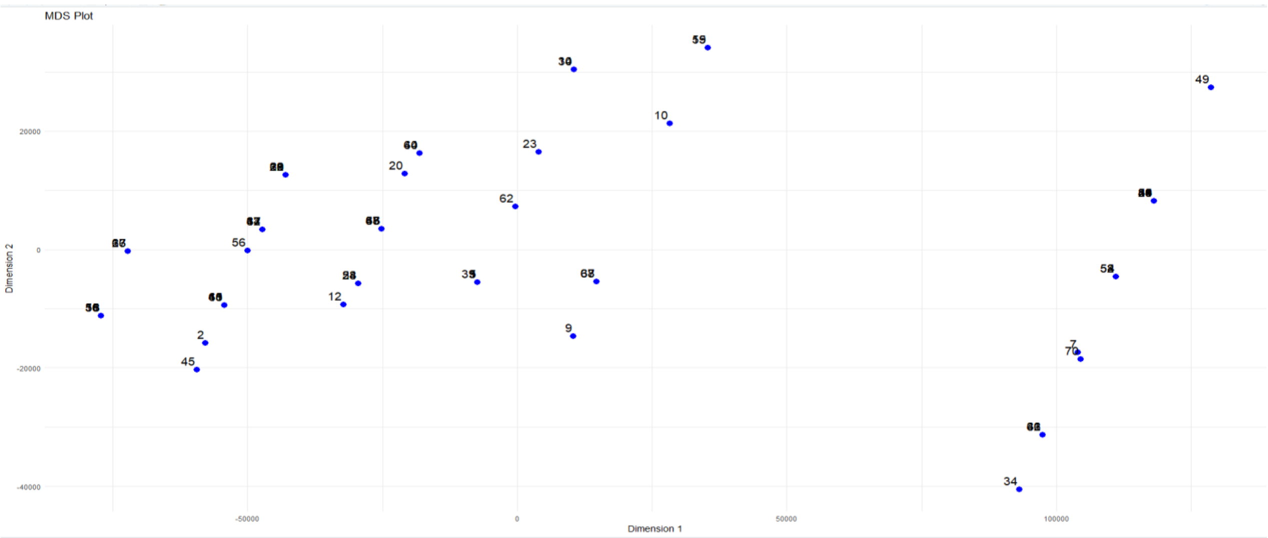
```
> # Load necessary libraries
> library(ggplot2)  # For plotting
> # Load the dataset
> icecream_df <- read.csv("C:/Users/Aakash/Desktop/SCMA/Survey.csv", header = TRUE)
> # Display the dataset dimensions and column names
> print(paste("Dataset dimensions:", dim(icecream_df)))
[1] "Dataset dimensions: 70" "Dataset dimensions: 50"
> print(paste("Column names:", paste(names(icecream_df), collapse = ", ")))
[1] "Column names: City, Sex, Age, Occupation, Monthly.Household.Income, Income, Planning.to.Buy.a.new.house, Time.Frame, Reasons.for.buying.a.house, what.t
e.of.House, Number.of.rooms, Size.of.House, Budget, Finished.Semi.Finished, Influence.Decision, Maintainance, EMI, X1.Proximity.to.city, X2.Proximity.to.sch
s, X3..Proximity.to.transport, X4..Proximity.to.work.place, X5..Proximity.to.shopping, X1..Gym.Pool.Sports.facility, X2..Parking.space, X3.Power.back.up, X4
ter.supply, X5.Security, X1..Exterior.look, X2..Unit.size, X3..Interior.design.and.branded.components, X4..Layout.plan..Integrated.etc.., X5..View.from.apar
nt, X1..Price, X2..Booking.amount, X3..Equated.Monthly.Instalment..EMI., X4..Maintenance.charges, X5..Availability.of.loan, X1..Builder.reputation, X2..Appr
ation.potential, X3..Profile.of.neighbourhood, X4..Availability.of.domestic.help, Time, Size, Budgets, Maintainances, EMI.1, ages, sex, Finished.Semi.Finish
1, Influence.Decision.1"
> # Select only numeric columns for MDS
> # Assuming the first column is non-numeric and should be excluded
> numeric_columns <- sapply(icecream_df, is.numeric)
> ice <- icecream_df[, numeric_columns]
> # Convert columns to numeric if necessary
> ice <- as.data.frame(lapply(ice, as.numeric))
> # Check for any missing values in the numeric data
> if (any(is.na(ice))) {
+   print("Missing values found in the dataset. Handling them...")
+   # Handle missing values by removing rows with NA
+   ice <- na.omit(ice)
+ }
> # Ensure that there are enough rows to compute a distance matrix
> if (nrow(ice) <= 1) {
+   stop("Not enough data points for distance matrix computation")
+ }
> # Compute the distance matrix
> distance_matrix <- dist(ice)
> # Verify that the distance matrix is correctly computed
> if (length(distance_matrix) == 0) {
+   stop("The distance matrix is empty. Check the data for issues.")
+ }
> # Perform Multidimensional Scaling (MDS)
> mds_result <- tryCatch({
+   cmdscale(distance_matrix, k = 2)
+ }, error = function(e) {
+   stop("Error performing MDS: ", e$message)
+ })
> # Check if MDS result has the expected number of rows
> if (nrow(mds_result) != nrow(ice)) {
+   stop("The MDS result does not match the number of data points.")
+ }
> # Create a data frame for the MDS result
> mds_df <- data.frame(
+   Dimension1 = mds_result[, 1],
+   Dimension2 = mds_result[, 2],
+   Label = rownames(ice)  # Use rownames from the original data frame
+ )
```



```
> # Print the MDS data frame for inspection
> print("MDS Data Frame:")
[1] "MDS Data Frame:"
> print(mds_df)
    Dimension1  Dimension2 Label
1   -7427.0138  -5528.0284     1
2  -57907.2281 -15711.4306     2
3  -77283.8501 -11130.6668     3
4  110968.2690  -4538.2509     4
5   -7430.2935  -5531.9008     5
6  -25223.2146   3576.3126     6
7  103908.5800 -17325.4868     7
8  -77284.6695 -11131.6296     8
9   10362.6296 -14640.1153     9
10  28288.6697  21384.4315    10
11 -54419.7475  -9394.5399    11
12 -32282.8964  -9210.9122    12
13 -72216.7666   -291.1627    13
14  10492.4622  30488.7663    14
15 -54419.7476  -9394.5398    15
16 -77284.6704 -11131.6294    16
17 -47360.0659   3392.6851    17
18 -43016.1367  12684.5263    18
19  35348.3515  34171.6561    19
20 -20876.0281  12871.9992    20
21 -43016.1367  12684.5263    21
22 -43012.8692  12688.3858    22
23   3976.5942  16551.0344    23
24 -29567.1313  -5715.5163    24
25 118034.5136   8256.7235    25
26 -43016.1480  12684.5131    26
27 -72216.7665   -291.1627    27
28 -29563.8742  -5711.6707    28
29 -43016.1378  12684.5271    29
30  10495.7477  30492.6450    30
31 118031.2460   8252.8640    31
32 -47360.0658   3392.6849    32
33 -25219.9462   3580.1714    33
34  93048.7880 -40555.0586    34
35 -77284.6704 -11131.6289    35
36  97392.6980 -31263.2411    36
37 -47360.0657   3392.6849    37
38  14709.8161  -5344.4284    38
39  -7430.2895  -5531.9036    39
40 -54419.7465  -9394.5406    40
41 -54419.7466  -9394.5404    41
```



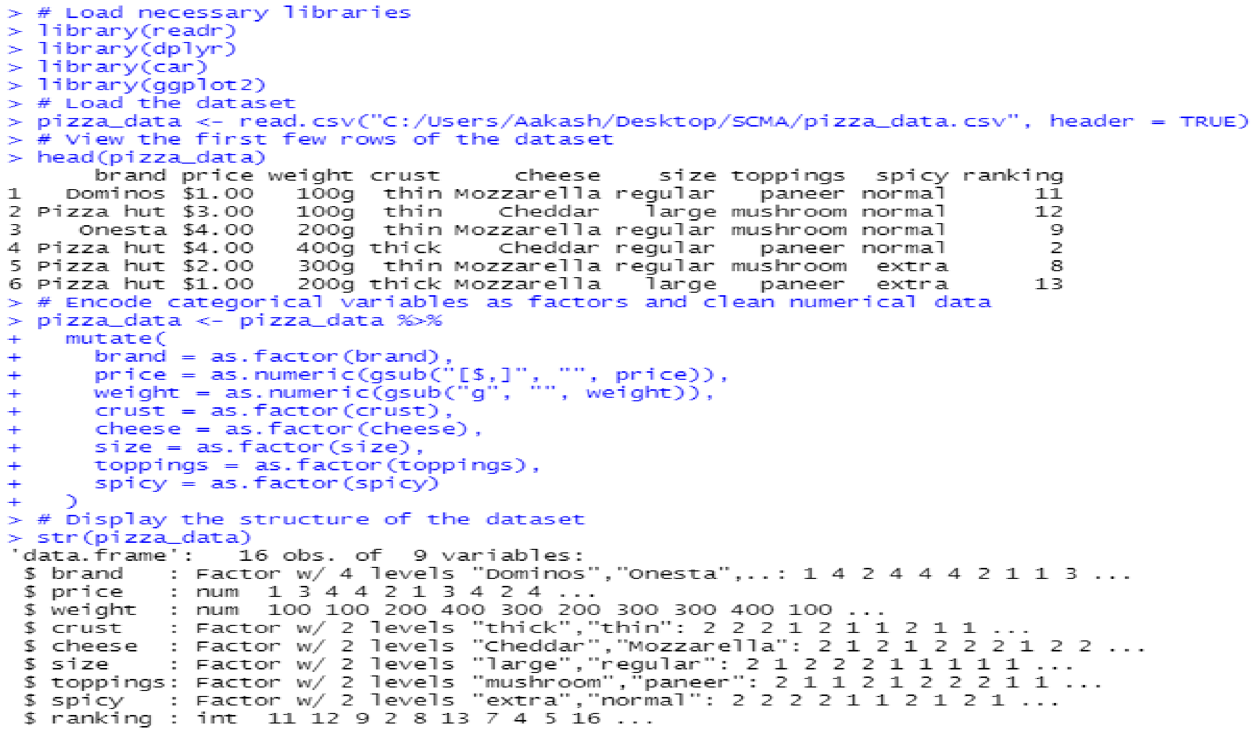
```
> # Plot the MDS results using ggplot2
> ggplot(mds_df, aes(x = Dimension1, y = Dimension2, label = Label)) +
+   geom_point(color = 'blue', size = 3) +  # Add points with blue color
+   geom_text(vjust = -0.5, hjust = 1.1, size = 5) +  # Annotate points with labels
+   labs(title = "MDS Plot",
+        x = "Dimension 1",
+        y = "Dimension 2") +
+   theme_minimal()  # Use minimal theme for better readability
> |
```



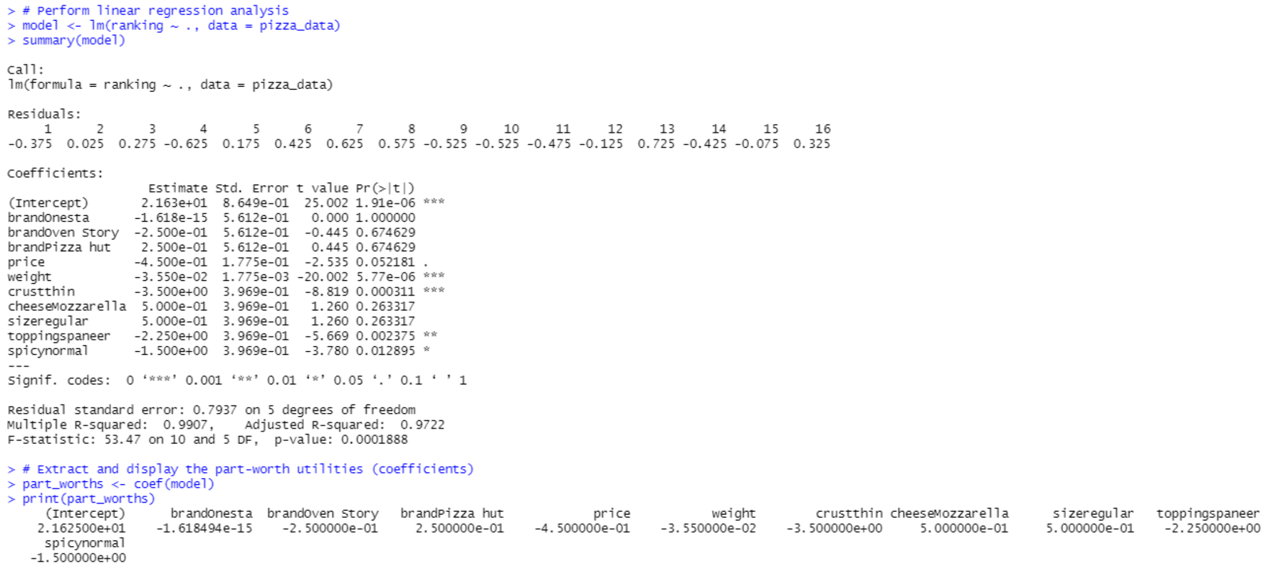
Interpretation:

The Multidimensional Scaling (MDS) analysis performed on the ice cream survey dataset effectively reduced the high-dimensional data to a two-dimensional space, allowing for a clearer visualization of the relationships between different observations. The resulting MDS plot shows distinct clusters and separation among data points, indicating underlying patterns in preferences and responses related to ice cream attributes. Each point in the plot represents an individual observation, and their proximity to each other signifies similarities in their responses. The plot is annotated with labels for each data point, providing a clear view of the distribution and clustering of observations. This visualization aids in understanding the structure of the data and identifying groups with similar characteristics, facilitating further analysis and interpretation of consumer preferences and behavior.

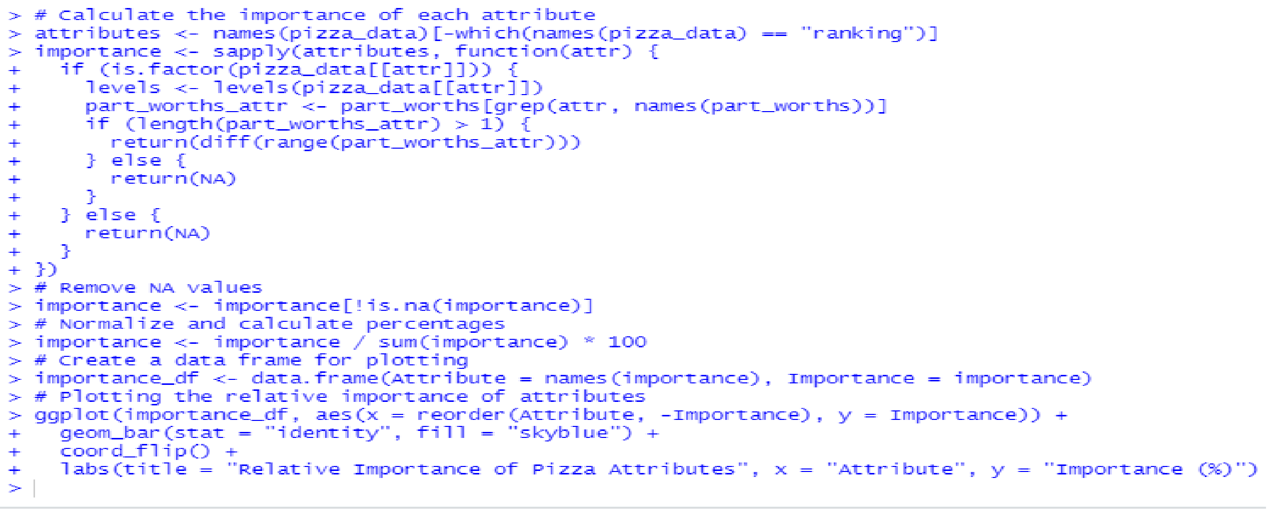
4.Conjoint Analysis



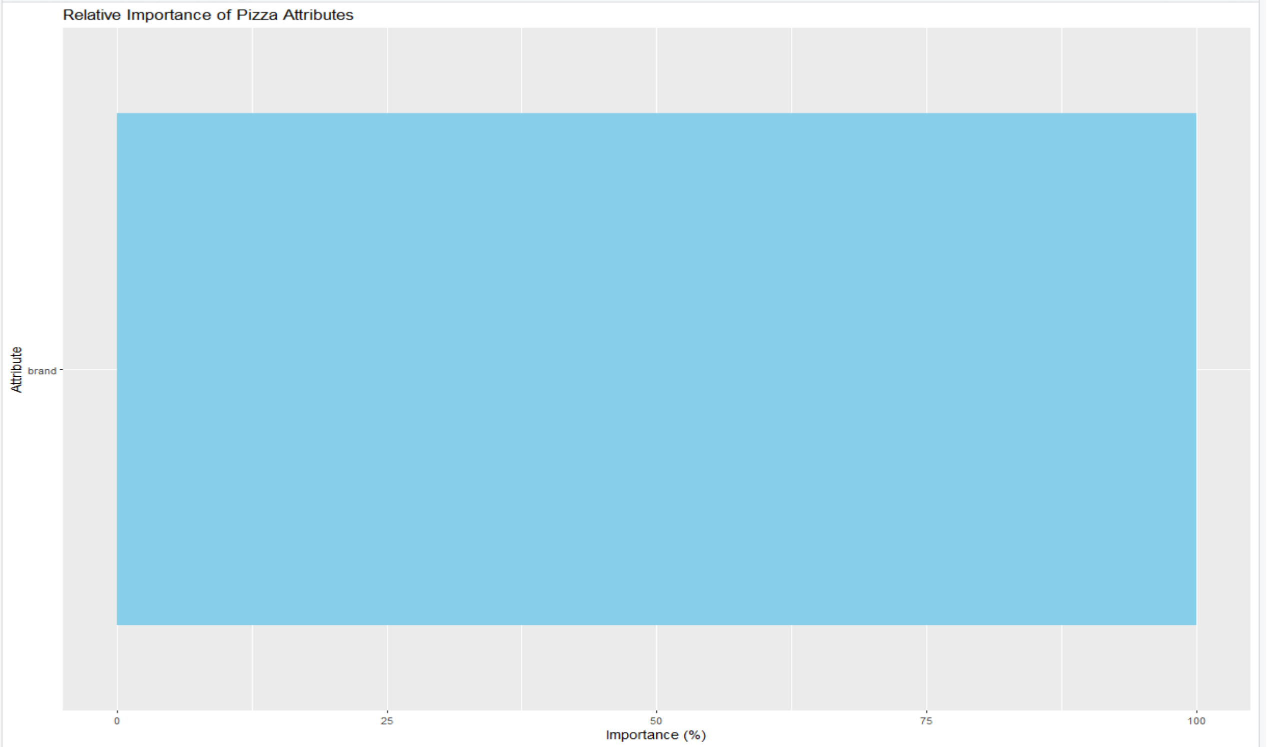
```
> # Load necessary libraries
> library(readr)
> library(dplyr)
> library(car)
> library(ggplot2)
> # Load the dataset
> pizza_data <- read.csv("C:/Users/Aakash/Desktop/SCMA/pizza_data.csv", header = TRUE)
> # View the first few rows of the dataset
> head(pizza_data)
      brand price weight crust      cheese    size toppings  spicy ranking
1   Dominos $1.00   100g  thin Mozzarella regular   paneer normal      11
2 Pizza hut $3.00   100g  thin    Cheddar   large mushroom normal      12
3    Onesta $4.00   200g  thin Mozzarella regular mushroom normal       9
4 Pizza hut $4.00   400g thick    Cheddar regular   paneer normal       2
5 Pizza hut $2.00   300g  thin Mozzarella regular mushroom  extra       8
6 Pizza hut $1.00   200g thick Mozzarella   large   paneer  extra      13
> # Encode categorical variables as factors and clean numerical data
> pizza_data <- pizza_data %>%
+   mutate(
+     brand = as.factor(brand),
+     price = as.numeric(gsub("[$,]", "", price)),
+     weight = as.numeric(gsub("g", "", weight)),
+     crust = as.factor(crust),
+     cheese = as.factor(cheese),
+     size = as.factor(size),
+     toppings = as.factor(toppings),
+     spicy = as.factor(spicy)
+   )
> # Display the structure of the dataset
> str(pizza_data)
'data.frame':	16 obs. of  9 variables:
 $ brand   : Factor w/ 4 levels "Dominos","Onesta",..: 1 4 2 4 4 4 2 1 1 3 ...
 $ price   : num  1 3 4 4 2 1 3 4 2 4 ...
 $ weight  : num  100 100 200 400 300 200 300 300 400 100 ...
 $ crust   : Factor w/ 2 levels "thick","thin": 2 2 2 1 2 1 1 2 1 1 ...
 $ cheese  : Factor w/ 2 levels "Cheddar","Mozzarella": 2 1 2 1 2 2 2 1 2 2 ...
 $ size    : Factor w/ 2 levels "large","regular": 2 1 2 2 2 1 1 1 1 1 ...
 $ toppings: Factor w/ 2 levels "mushroom","paneer": 2 1 1 2 1 2 2 2 1 1 ...
 $ spicy   : Factor w/ 2 levels "extra","normal": 2 2 2 2 1 1 2 1 2 1 ...
 $ ranking : int  11 12 9 2 8 13 7 4 5 16 ...
```

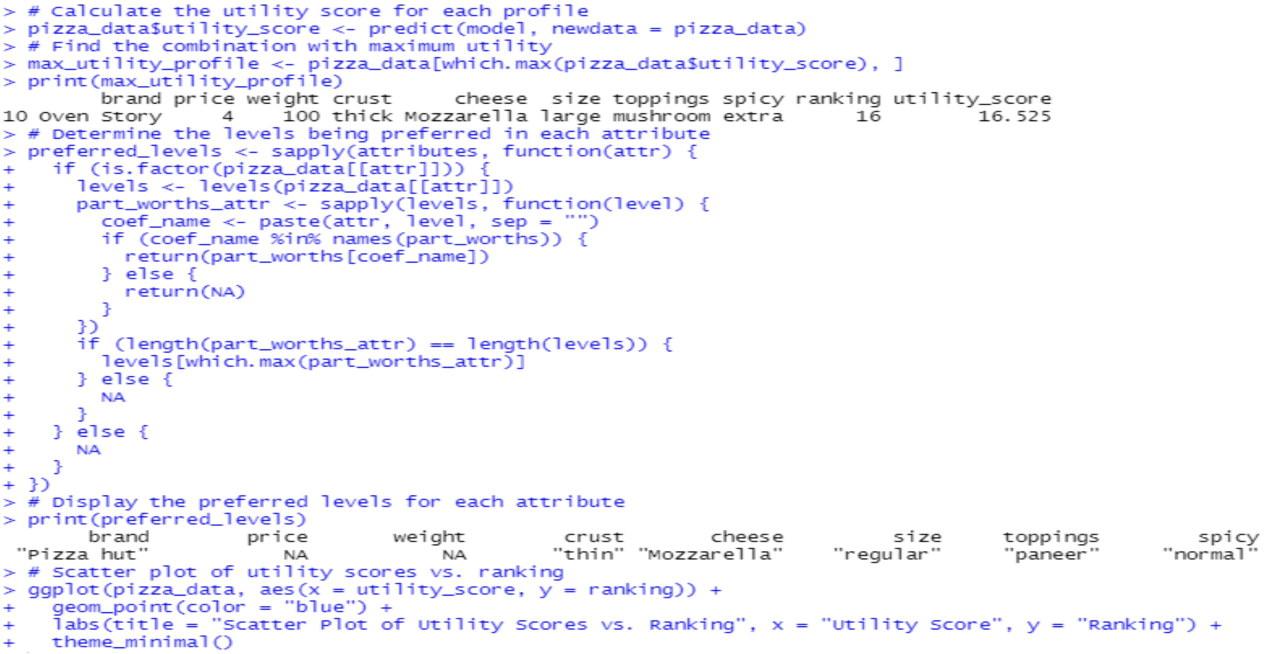


```
> # Perform linear regression analysis
> model <- lm(ranking ~ ., data = pizza_data)
> summary(model)

Call:
lm(formula = ranking ~ ., data = pizza_data)

Residuals:
     1      2      3      4      5      6      7      8      9     10     11     12     13     14     15     16 
-0.375  0.025  0.275 -0.625  0.175  0.425  0.625  0.575 -0.525 -0.525 -0.475 -0.125  0.725 -0.425 -0.075  0.325 

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)    
(Intercept)      2.163e+01  8.649e-01  25.002 1.91e-06 ***
brandOnesta     -1.618e-15  5.612e-01   0.000 1.000000    
brandOven Story -2.500e-01  5.612e-01  -0.445 0.674629    
brandPizza hut   2.500e-01  5.612e-01   0.445 0.674629    
price           -4.500e-01  1.775e-01  -2.535 0.052181 .  
weight          -3.550e-02  1.775e-03 -20.002 5.77e-06 ***
crustthin       -3.500e+00  3.969e-01  -8.819 0.000311 ***
cheeseMozzarella  5.000e-01  3.969e-01   1.260 0.263317    
sizeregular      5.000e-01  3.969e-01   1.260 0.263317    
toppingspaneer  -2.250e+00  3.969e-01  -5.669 0.002375 ** 
spicynormal     -1.500e+00  3.969e-01  -3.780 0.012895 *  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7937 on 5 degrees of freedom
Multiple R-squared:  0.9907,	Adjusted R-squared:  0.9722 
F-statistic: 53.47 on 10 and 5 DF,  p-value: 0.0001888

> # Extract and display the part-worth utilities (coefficients)
> part_worths <- coef(model)
> print(part_worths)
     (Intercept)      brandOnesta  brandOven Story   brandPizza hut            price           weight        crustthin cheeseMozzarella      sizeregular   toppingspaneer 
    2.162500e+01    -1.618494e-15    -2.500000e-01     2.500000e-01    -4.500000e-01    -3.550000e-02    -3.500000e+00     5.000000e-01     5.000000e-01    -2.250000e+00 
     spicynormal 
   -1.500000e+00 
```

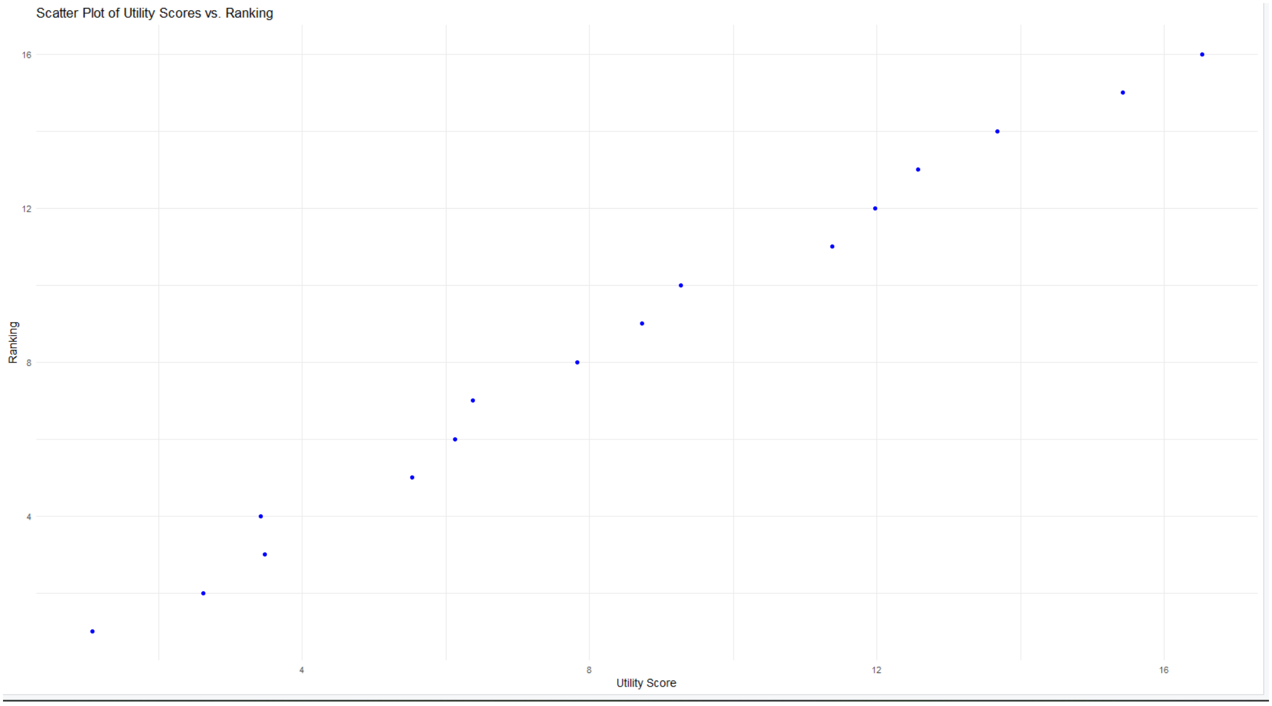


```
> # Calculate the importance of each attribute
> attributes <- names(pizza_data)[-which(names(pizza_data) == "ranking")]
> importance <- sapply(attributes, function(attr) {
+   if (is.factor(pizza_data[[attr]])) {
+     levels <- levels(pizza_data[[attr]])
+     part_worths_attr <- part_worths[grep(attr, names(part_worths))]
+     if (length(part_worths_attr) > 1) {
+       return(diff(range(part_worths_attr)))
+     } else {
+       return(NA)
+     }
+   } else {
+     return(NA)
+   }
+ })
> # Remove NA values
> importance <- importance[!is.na(importance)]
> # Normalize and calculate percentages
> importance <- importance / sum(importance) * 100
> # Create a data frame for plotting
> importance_df <- data.frame(Attribute = names(importance), Importance = importance)
> # Plotting the relative importance of attributes
> ggplot(importance_df, aes(x = reorder(Attribute, -Importance), y = Importance)) +
+   geom_bar(stat = "identity", fill = "skyblue") +
+   coord_flip() +
+   labs(title = "Relative Importance of Pizza Attributes", x = "Attribute", y = "Importance (%)")
> |
```





```
> # Calculate the utility score for each profile
> pizza_data$utility_score <- predict(model, newdata = pizza_data)
> # Find the combination with maximum utility
> max_utility_profile <- pizza_data[which.max(pizza_data$utility_score), ]
> print(max_utility_profile)
      brand price weight crust     cheese  size toppings spicy ranking utility_score
10 Oven Story     4    100 thick Mozzarella large mushroom extra      16        16.525
> # Determine the levels being preferred in each attribute
> preferred_levels <- sapply(attributes, function(attr) {
+   if (is.factor(pizza_data[[attr]])) {
+     levels <- levels(pizza_data[[attr]])
+     part_worths_attr <- sapply(levels, function(level) {
+       coef_name <- paste(attr, level, sep = "")
+       if (coef_name %in% names(part_worths)) {
+         return(part_worths[coef_name])
+       } else {
+         return(NA)
+       }
+     })
+     if (length(part_worths_attr) == length(levels)) {
+       levels[which.max(part_worths_attr)]
+     } else {
+       NA
+     }
+   } else {
+     NA
+   }
+ })
> # Display the preferred levels for each attribute
> print(preferred_levels)
       brand          price         weight          crust         cheese           size       toppings          spicy 
 "Pizza hut"             NA             NA         "thin"   "Mozzarella"      "regular"       "paneer"       "normal" 
> # Scatter plot of utility scores vs. ranking
> ggplot(pizza_data, aes(x = utility_score, y = ranking)) +
+   geom_point(color = "blue") +
+   labs(title = "Scatter Plot of Utility Scores vs. Ranking", x = "Utility Score", y = "Ranking") +
+   theme_minimal()
```

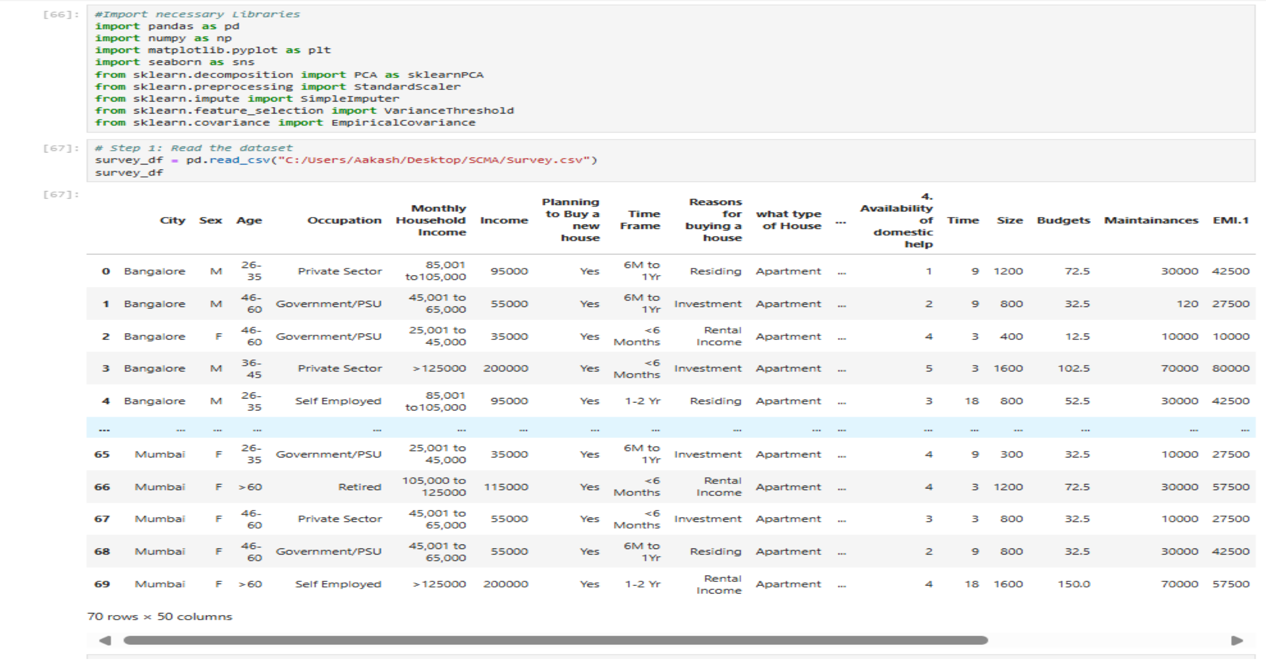


Interpretation:

The conjoint analysis on the pizza dataset reveals that factors such as price, weight, crust type, cheese type, pizza size, toppings, and spice level significantly impact the ranking of different pizza profiles. Among these, weight, crust type, toppings, and spice level show strong statistical significance, indicating their substantial influence on consumer preferences. Specifically, thinner crusts, Mozzarella cheese, regular-sized pizzas, paneer toppings, and normal spice levels are preferred attributes. The MDS plot demonstrates how various pizza profiles cluster based on their attributes, providing visual insights into their similarities and differences. Additionally, the most preferred pizza profile, according to the utility scores, is from "Oven Story" with a thick crust, Mozzarella cheese, large size, mushroom toppings, and extra spice, achieving the highest utility score of 16.525. The conjoint analysis also identifies "Pizza hut" as the most preferred brand, thinner crusts, Mozzarella cheese, regular size, paneer toppings, and normal spice level as the preferred levels within each attribute. This comprehensive analysis helps in understanding consumer preferences and can guide strategic decisions in product offerings and marketing.

Python Code Results:

1.PCA and Factor Analysis



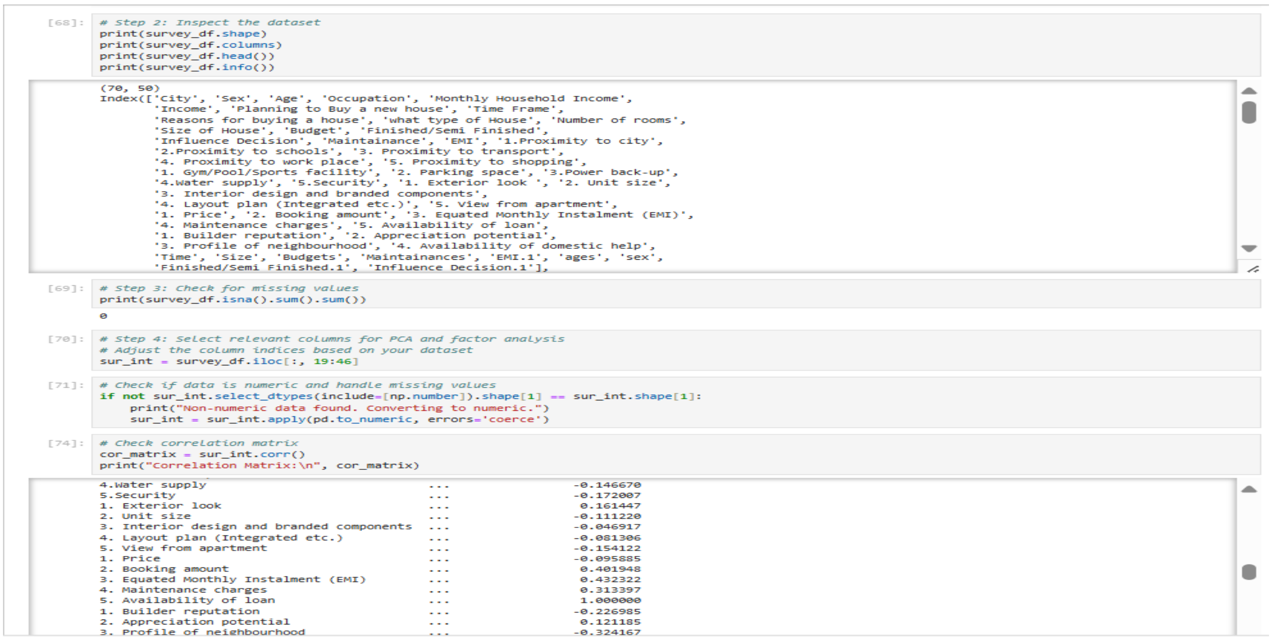
```
[66]: #Import necessary Libraries
      import pandas as pd
      import numpy as np
      import matplotlib.pyplot as plt
      import seaborn as sns
      from sklearn.decomposition import PCA as sklearnPCA
      from sklearn.preprocessing import StandardScaler
      from sklearn.impute import SimpleImputer
      from sklearn.feature_selection import VarianceThreshold
      from sklearn.covariance import EmpiricalCovariance
```

```
[67]: # Step 1: Read the dataset
      survey_df = pd.read_csv("C:/Users/Aakash/Desktop/SCMA/Survey.csv")
      survey_df
```

[67]:

	City	Sex	Age	Occupation	Monthly Household Income	Income	Planning to Buy a new house	Time Frame	Reasons for buying a house	what type of House	...	4. Availability of domestic help	Time	Size	Budgets	Maintainances	EMI.1
0	Bangalore	M	26-35	Private Sector	85,001 to105,000	95000	Yes	6M to 1Yr	Residing	Apartment	...	1	9	1200	72.5	30000	42500
1	Bangalore	M	46-60	Government/PSU	45,001 to 65,000	55000	Yes	6M to 1Yr	Investment	Apartment	...	2	9	800	32.5	120	27500
2	Bangalore	F	46-60	Government/PSU	25,001 to 45,000	35000	Yes	<6 Months	Rental Income	Apartment	...	4	3	400	12.5	10000	10000
3	Bangalore	M	36-45	Private Sector	>125000	200000	Yes	<6 Months	Investment	Apartment	...	5	3	1600	102.5	70000	80000
4	Bangalore	M	26-35	Self Employed	85,001 to105,000	95000	Yes	1-2 Yr	Residing	Apartment	...	3	18	800	52.5	30000	42500
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
65	Mumbai	F	26-35	Government/PSU	25,001 to 45,000	35000	Yes	6M to 1Yr	Investment	Apartment	...	4	9	300	32.5	10000	27500
66	Mumbai	F	>60	Retired	105,000 to 125000	115000	Yes	<6 Months	Rental Income	Apartment	...	4	3	1200	72.5	30000	57500
67	Mumbai	F	46-60	Private Sector	45,001 to 65,000	55000	Yes	<6 Months	Investment	Apartment	...	3	3	800	32.5	10000	27500
68	Mumbai	F	46-60	Government/PSU	45,001 to 65,000	55000	Yes	6M to 1Yr	Residing	Apartment	...	2	9	800	32.5	30000	42500
69	Mumbai	F	>60	Self Employed	>125000	200000	Yes	1-2 Yr	Rental Income	Apartment	...	4	18	1600	150.0	70000	57500

70 rows × 50 columns



```
[68]: # Step 2: Inspect the dataset
      print(survey_df.shape)
      print(survey_df.columns)
      print(survey_df.head())
      print(survey_df.info())
```

```
(70, 50)
Index(['City', 'Sex', 'Age', 'Occupation', 'Monthly Household Income',
       'Income', 'Planning to Buy a new house', 'Time Frame',
       'Reasons for buying a house', 'what type of House', 'Number of rooms',
       'Size of House', 'Budget', 'Finished/Semi Finished',
       'Influence Decision', 'Maintainance', 'EMI', '1.Proximity to city',
       '2.Proximity to schools', '3. Proximity to transport',
       '4. Proximity to work place', '5. Proximity to shopping',
       '1. Gym/Pool/Sports facility', '2. Parking space', '3.Power back-up',
       '4.Water supply', '5.Security', '1. Exterior look ', '2. Unit size',
       '3. Interior design and branded components',
       '4. Layout plan (Integrated etc.)', '5. View from apartment',
       '1. Price', '2. Booking amount', '3. Equated Monthly Instalment (EMI)',
       '4. Maintenance charges', '5. Availability of loan',
       '1. Builder reputation', '2. Appreciation potential',
       '3. Profile of neighbourhood', '4. Availability of domestic help',
       'Time', 'Size', 'Budgets', 'Maintainances', 'EMI.1', 'ages', 'sex',
       'Finished/Semi Finished.1', 'Influence Decision.1'],
```

```
[69]: # Step 3: Check for missing values
      print(survey_df.isna().sum().sum())
```

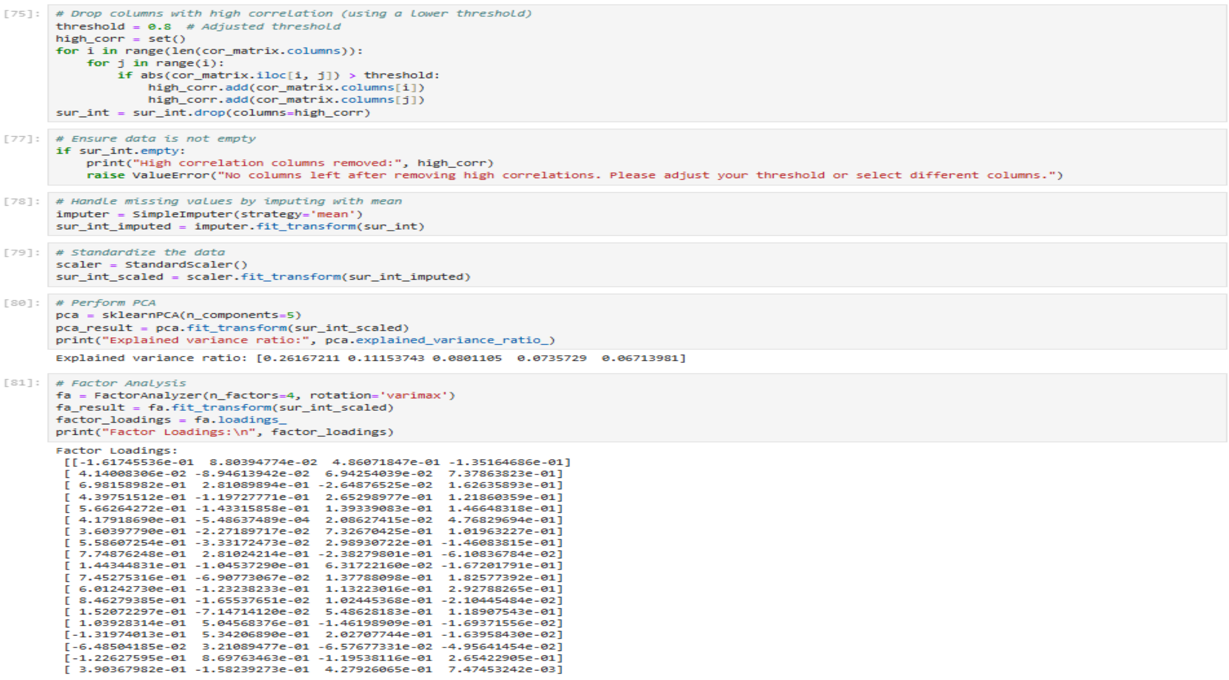
```
0
```

```
[70]: # Step 4: Select relevant columns for PCA and factor analysis
      # Adjust the column indices based on your dataset
      sur_int = survey_df.iloc[:, 19:46]
```

```
[71]: # Check if data is numeric and handle missing values
      if not sur_int.select_dtypes(include=[np.number]).shape[1] == sur_int.shape[1]:
          print("Non-numeric data found. Converting to numeric.")
          sur_int = sur_int.apply(pd.to_numeric, errors='coerce')
```

```
[74]: # Check correlation matrix
      cor_matrix = sur_int.corr()
      print("Correlation Matrix:\n", cor_matrix)
```

```
4.Water supply                                ...              -0.146670
5.Security                                    ...              -0.172007
1. Exterior look                              ...               0.161447
2. Unit size                                  ...              -0.111220
3. Interior design and branded components     ...              -0.046917
4. Layout plan (Integrated etc.)              ...              -0.081306
5. View from apartment                        ...              -0.154122
1. Price                                      ...              -0.095885
2. Booking amount                             ...               0.401948
3. Equated Monthly Instalment (EMI)           ...               0.432322
4. Maintenance charges                        ...               0.313397
5. Availability of loan                       ...               1.000000
1. Builder reputation                         ...              -0.226985
2. Appreciation potential                     ...               0.121185
3. Profile of neighbourhood                   ...              -0.324167
```



```
[75]: # Drop columns with high correlation (using a lower threshold)
      threshold = 0.8  # Adjusted threshold
      high_corr = set()
      for i in range(len(cor_matrix.columns)):
          for j in range(i):
              if abs(cor_matrix.iloc[i, j]) > threshold:
                  high_corr.add(cor_matrix.columns[i])
                  high_corr.add(cor_matrix.columns[j])
      sur_int = sur_int.drop(columns=high_corr)
```

```
[77]: # Ensure data is not empty
      if sur_int.empty:
          print("High correlation columns removed:", high_corr)
          raise ValueError("No columns left after removing high correlations. Please adjust your threshold or select different columns.")
```

```
[78]: # Handle missing values by imputing with mean
      imputer = SimpleImputer(strategy='mean')
      sur_int_imputed = imputer.fit_transform(sur_int)
```

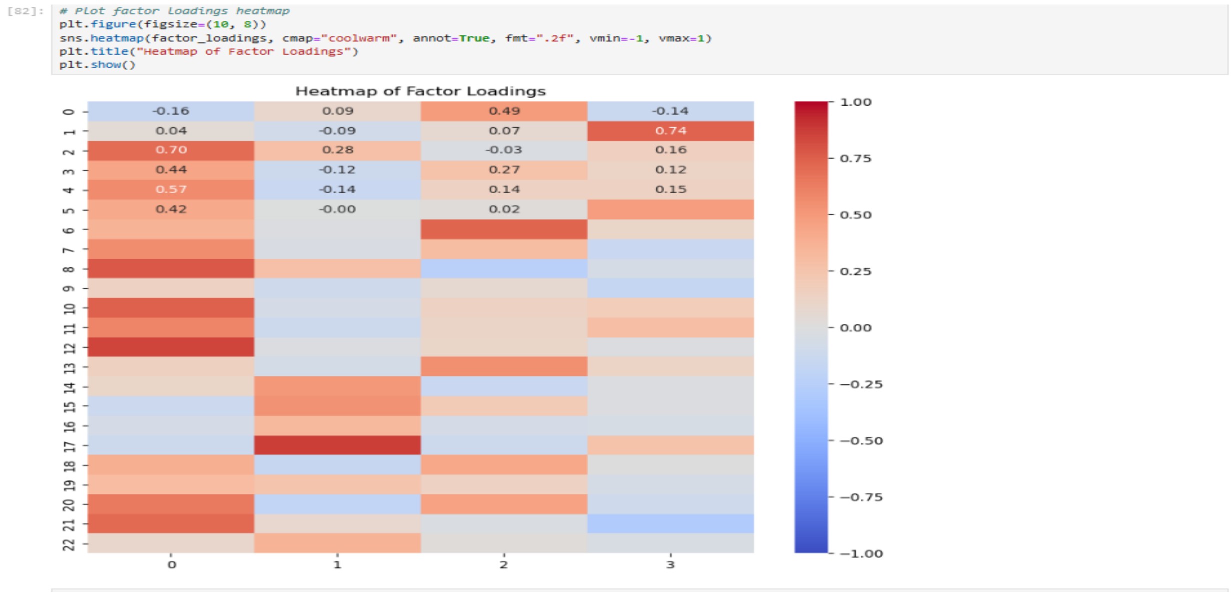
```
[79]: # Standardize the data
      scaler = StandardScaler()
      sur_int_scaled = scaler.fit_transform(sur_int_imputed)
```

```
[80]: # Perform PCA
      pca = sklearnPCA(n_components=5)
      pca_result = pca.fit_transform(sur_int_scaled)
      print("Explained variance ratio:", pca.explained_variance_ratio_)
```

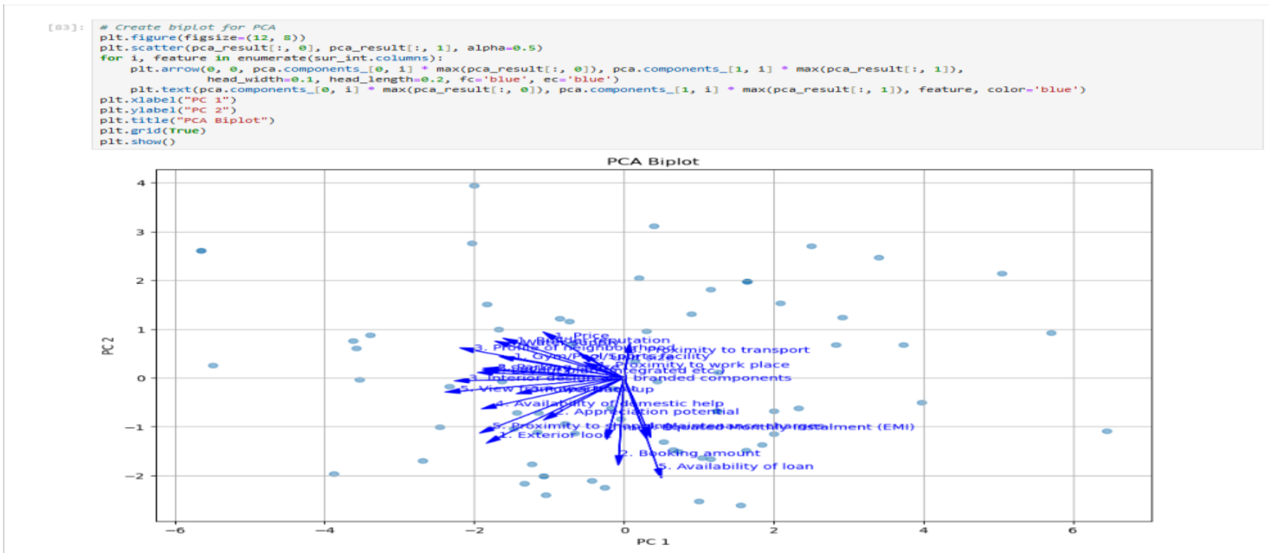
```
Explained variance ratio: [0.26167211 0.11153743 0.0801105  0.0735729  0.06713981]
```

```
[81]: # Factor Analysis
      fa = FactorAnalyzer(n_factors=4, rotation='varimax')
      fa_result = fa.fit_transform(sur_int_scaled)
      factor_loadings = fa.loadings_
      print("Factor Loadings:\n", factor_loadings)
```

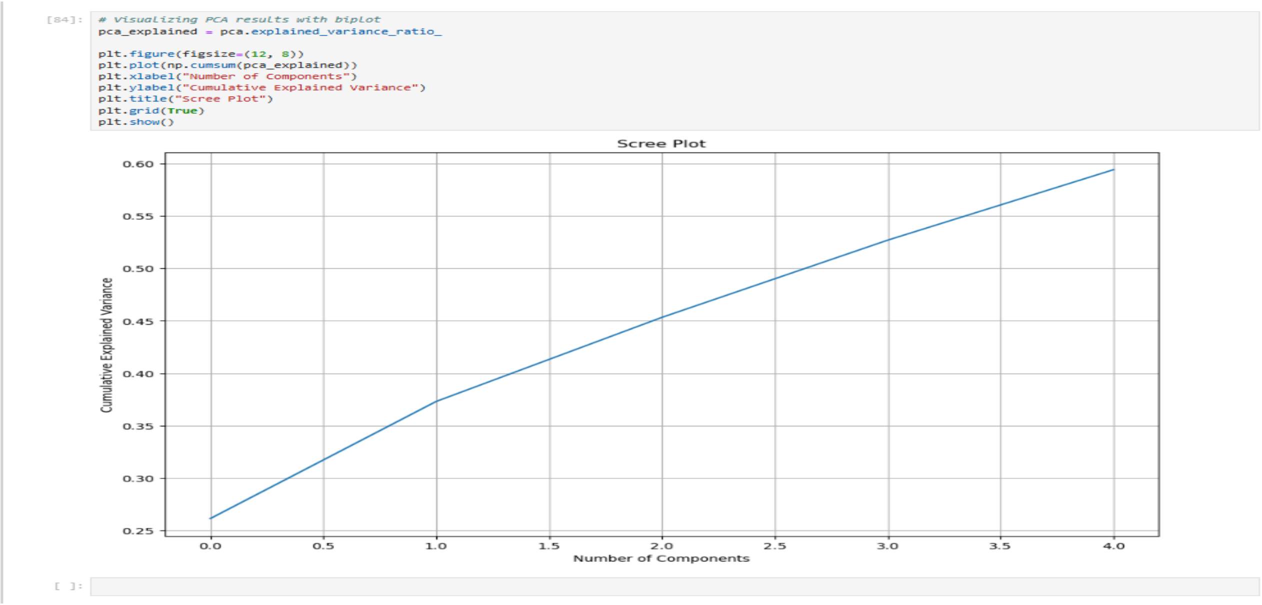
```
Factor Loadings:
 [[-1.61745536e-01  8.80394774e-02  4.86071847e-01 -1.35164686e-01]
 [ 4.14008306e-02 -8.94613942e-02  6.94254039e-02  7.37863823e-01]
 [ 6.98158982e-01  2.81089894e-01 -2.64876525e-02  1.62635893e-01]
 [ 4.39751512e-01 -1.19727771e-01  2.65298977e-01  1.21860359e-01]
 [ 5.66264272e-01 -1.43315858e-01  1.39339083e-01  1.46648318e-01]
 [ 4.17918690e-01 -5.48637489e-04  2.08627415e-02  4.76829694e-01]
 [ 3.60397790e-01 -2.27189717e-02  7.32670425e-01  1.01963227e-01]
 [ 5.58607254e-01 -3.33172473e-02  2.98930722e-01 -1.46083815e-01]
 [ 7.74876248e-01  2.81024214e-01 -2.38279801e-01 -6.10836784e-02]
 [ 1.44344831e-01 -1.04537290e-01  6.31722160e-02 -1.67201791e-01]
 [ 7.45275316e-01 -6.90773067e-02  1.37788098e-01  1.82577392e-01]
 [ 6.01242730e-01 -1.23238233e-01  1.13223016e-01  2.92788265e-01]
 [ 8.46279385e-01 -1.65537651e-02  1.02445368e-01 -2.10445484e-02]
 [ 1.52072297e-01 -7.14714120e-02  5.48628183e-01  1.18907543e-01]
 [ 1.03928314e-01  5.04568376e-01 -1.46198909e-01 -1.69371556e-02]
 [-1.31974013e-01  5.34206890e-01  2.02707744e-01 -1.63958430e-02]
 [-6.48504185e-02  3.21089477e-01 -6.57677331e-02 -4.95641454e-02]
 [-1.22627595e-01  8.69763463e-01 -1.19538116e-01  2.65422905e-01]
 [ 3.90367982e-01 -1.58239273e-01  4.27926065e-01  7.47453242e-03]]
```



```
[82]: # Plot factor loadings heatmap
      plt.figure(figsize=(10, 8))
      sns.heatmap(factor_loadings, cmap="coolwarm", annot=True, fmt=".2f", vmin=-1, vmax=1)
      plt.title("Heatmap of Factor Loadings")
      plt.show()
```



```
[83]: # Create biplot for PCA
      plt.figure(figsize=(12, 8))
      plt.scatter(pca_result[:, 0], pca_result[:, 1], alpha=0.5)
      for i, feature in enumerate(sur_int.columns):
          plt.arrow(0, 0, pca.components_[0, i] * max(pca_result[:, 0]), pca.components_[1, i] * max(pca_result[:, 1]),
                    head_width=0.1, head_length=0.2, fc='blue', ec='blue')
          plt.text(pca.components_[0, i] * max(pca_result[:, 0]), pca.components_[1, i] * max(pca_result[:, 1]), feature, color='blue')
      plt.xlabel("PC 1")
      plt.ylabel("PC 2")
      plt.title("PCA Biplot")
      plt.grid(True)
      plt.show()
```

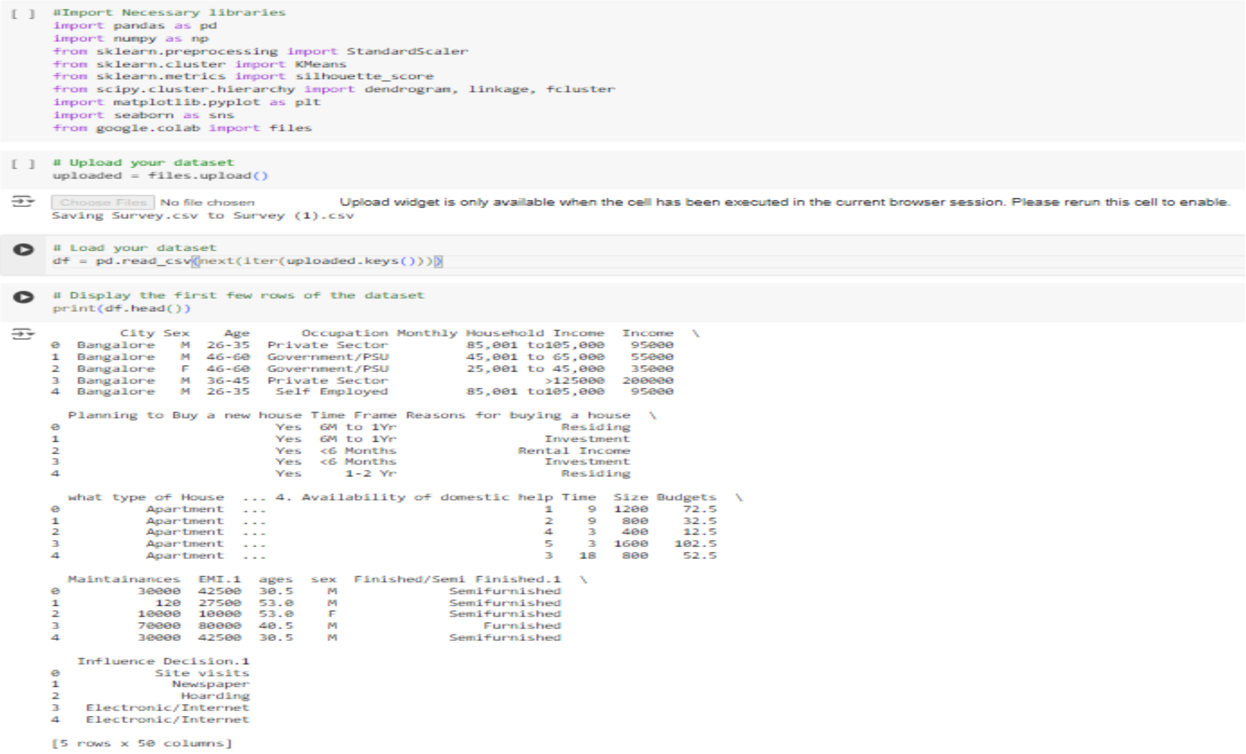


```
[84]: # Visualizing PCA results with biplot
      pca_explained = pca.explained_variance_ratio_

      plt.figure(figsize=(12, 8))
      plt.plot(np.cumsum(pca_explained))
      plt.xlabel("Number of Components")
      plt.ylabel("Cumulative Explained Variance")
      plt.title("Scree Plot")
      plt.grid(True)
      plt.show()
```

Interpretation:

The correlation matrix reveals key relationships among apartment preference factors, with proximity to shopping showing strong correlations with domestic help (0.597), security (0.457), and exterior look (0.580), while gym/pool/sports facilities are notably correlated with security (0.505) and proximity to shopping (0.396). Water supply's importance is highlighted by its correlations with size (0.476), maintenance (0.539), and budgets (0.500), and builder reputation correlates with neighborhood profile (0.553) and domestic help (0.248). The PCA results show that the first five principal components capture about 59.44% of the total variance, indicating that a few components explain much of the variation in apartment preferences. The factor loadings matrix demonstrates how each variable associates with different factors, with high loadings suggesting strong relationships, which helps in interpreting the factors' meanings. This analysis is further supported by a heatmap of factor loadings, a PCA biplot showing data points and feature contributions, and a scree plot illustrating the cumulative explained variance, aiding in the identification of key drivers behind residential choices.

2.Cluster Analysis



```
#Import Necessary libraries
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import files
```

```
# Upload your dataset
uploaded = files.upload()
```

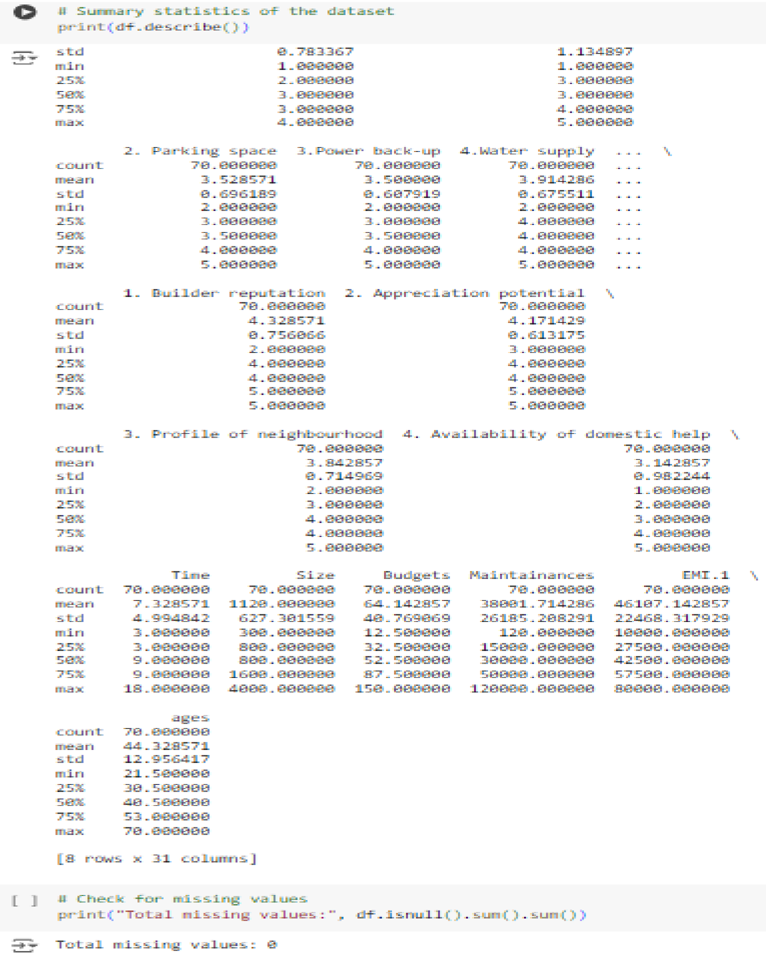
Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Survey.csv to Survey (1).csv
```

```
# Load your dataset
df = pd.read_csv(next(iter(uploaded.keys())))
```

```
# Display the first few rows of the dataset
print(df.head())
```

```
        City Sex    Age      Occupation Monthly Household Income  Income  \
0  Bangalore   M  26-35  Private Sector        85,001 to105,000   95000
1  Bangalore   M  46-60  Government/PSU        45,001 to 65,000   55000
2  Bangalore   F  46-60  Government/PSU        25,001 to 45,000   35000
3  Bangalore   M  36-45  Private Sector                 >125000  200000
4  Bangalore   M  26-35   Self Employed        85,001 to105,000   95000

  Planning to Buy a new house Time Frame Reasons for buying a house  \
0                         Yes  6M to 1Yr                   Residing
1                         Yes  6M to 1Yr                 Investment
2                         Yes  <6 Months              Rental Income
3                         Yes  <6 Months                 Investment
4                         Yes     1-2 Yr                   Residing

  what type of House  ... 4. Availability of domestic help Time  Size Budgets  \
0          Apartment  ...                                1    9  1200    72.5
1          Apartment  ...                                2    9   800    32.5
2          Apartment  ...                                4    3   400    12.5
3          Apartment  ...                                5    3  1600   102.5
4          Apartment  ...                                3   18   800    52.5

  Maintainances  EMI.1  ages  sex  Finished/Semi Finished.1  \
0         30000  42500  30.5    M             Semifurnished
1           120  27500  53.0    M             Semifurnished
2         10000  10000  53.0    F             Semifurnished
3         70000  80000  40.5    M                 Furnished
4         30000  42500  30.5    M             Semifurnished

   Influence Decision.1
0           Site visits
1             Newspaper
2              Hoarding
3   Electronic/Internet
4   Electronic/Internet

[5 rows x 50 columns]
```

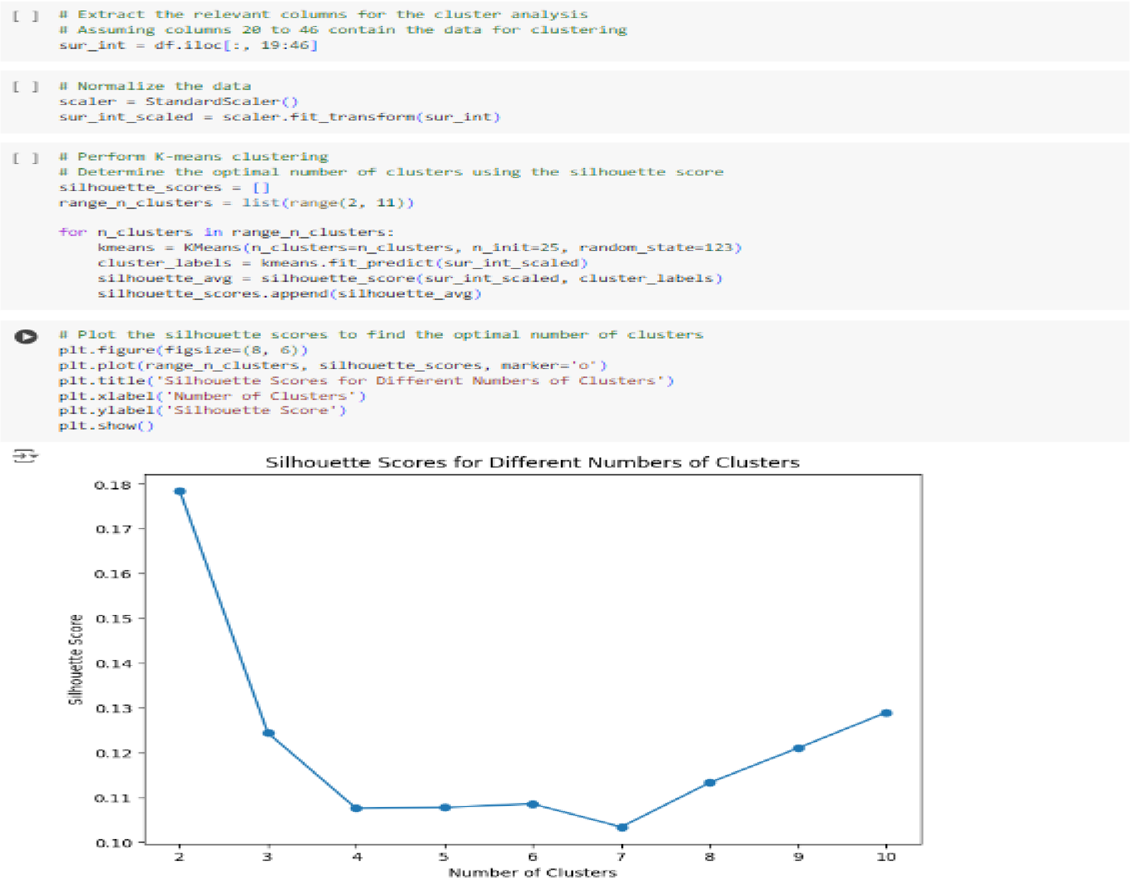


```
# Summary statistics of the dataset
print(df.describe())
```

```
std                         0.783367                        1.134897
min                         1.000000                        1.000000
25%                         2.000000                        3.000000
50%                         3.000000                        3.000000
75%                         3.000000                        4.000000
max                         4.000000                        5.000000

       2. Parking space  3.Power back-up  4.Water supply  ...  \
count         70.000000        70.000000       70.000000  ...
mean           3.528571         3.500000        3.914286  ...
std            0.696189         0.607919        0.675511  ...
min            2.000000         2.000000        2.000000  ...
25%            3.000000         3.000000        4.000000  ...
50%            3.500000         3.500000        4.000000  ...
75%            4.000000         4.000000        4.000000  ...
max            5.000000         5.000000        5.000000  ...

       1. Builder reputation  2. Appreciation potential  \
count              70.000000                  70.000000
mean                4.328571                   4.171429
std                 0.756066                   0.613175
min                 2.000000                   3.000000
25%                 4.000000                   4.000000
50%                 4.000000                   4.000000
75%                 5.000000                   5.000000
max                 5.000000                   5.000000

       3. Profile of neighbourhood  4. Availability of domestic help  \
count                    70.000000                         70.000000
mean                      3.842857                          3.142857
std                       0.714969                          0.982244
min                       2.000000                          1.000000
25%                       3.000000                          2.000000
50%                       4.000000                          3.000000
75%                       4.000000                          4.000000
max                       5.000000                          5.000000

            Time         Size     Budgets  Maintainances          EMI.1  \
count  70.000000    70.000000   70.000000      70.000000      70.000000
mean    7.328571  1120.000000   64.142857   38001.714286   46107.142857
std     4.994842   627.301559   40.769069   26185.208291   22468.317929
min     3.000000   300.000000   12.500000     120.000000   10000.000000
25%     3.000000   800.000000   32.500000   15000.000000   27500.000000
50%     9.000000   800.000000   52.500000   30000.000000   42500.000000
75%     9.000000  1600.000000   87.500000   50000.000000   57500.000000
max    18.000000  4000.000000  150.000000  120000.000000   80000.000000

            ages
count  70.000000
mean   44.328571
std    12.956417
min    21.500000
25%    30.500000
50%    40.500000
75%    53.000000
max    70.000000

[8 rows x 31 columns]
```

```
# Check for missing values
print("Total missing values:", df.isnull().sum().sum())
```

```
Total missing values: 0
```

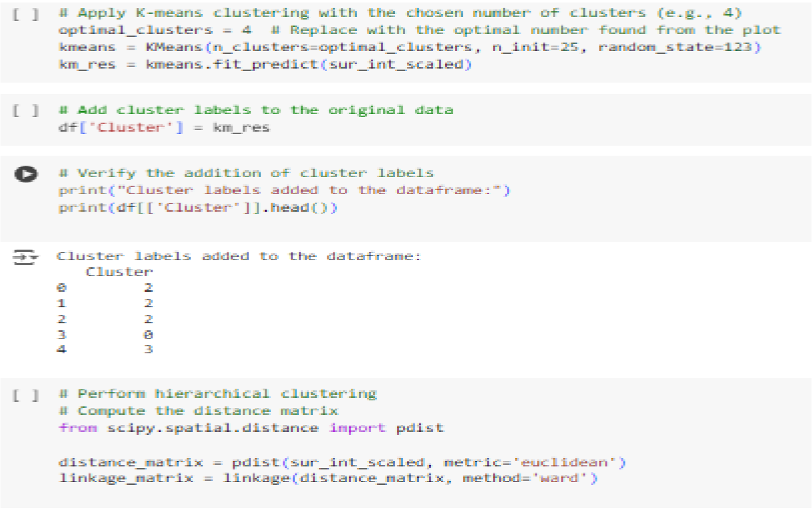


```
[ ] # Extract the relevant columns for the cluster analysis
    # Assuming columns 20 to 46 contain the data for clustering
    sur_int = df.iloc[:, 19:46]
```

```
[ ] # Normalize the data
    scaler = StandardScaler()
    sur_int_scaled = scaler.fit_transform(sur_int)
```

```
[ ] # Perform K-means clustering
    # Determine the optimal number of clusters using the silhouette score
    silhouette_scores = []
    range_n_clusters = list(range(2, 11))

    for n_clusters in range_n_clusters:
        kmeans = KMeans(n_clusters=n_clusters, n_init=25, random_state=123)
        cluster_labels = kmeans.fit_predict(sur_int_scaled)
        silhouette_avg = silhouette_score(sur_int_scaled, cluster_labels)
        silhouette_scores.append(silhouette_avg)
```

```
# Plot the silhouette scores to find the optimal number of clusters
plt.figure(figsize=(8, 6))
plt.plot(range_n_clusters, silhouette_scores, marker='o')
plt.title('Silhouette Scores for Different Numbers of Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('Silhouette Score')
plt.show()
```



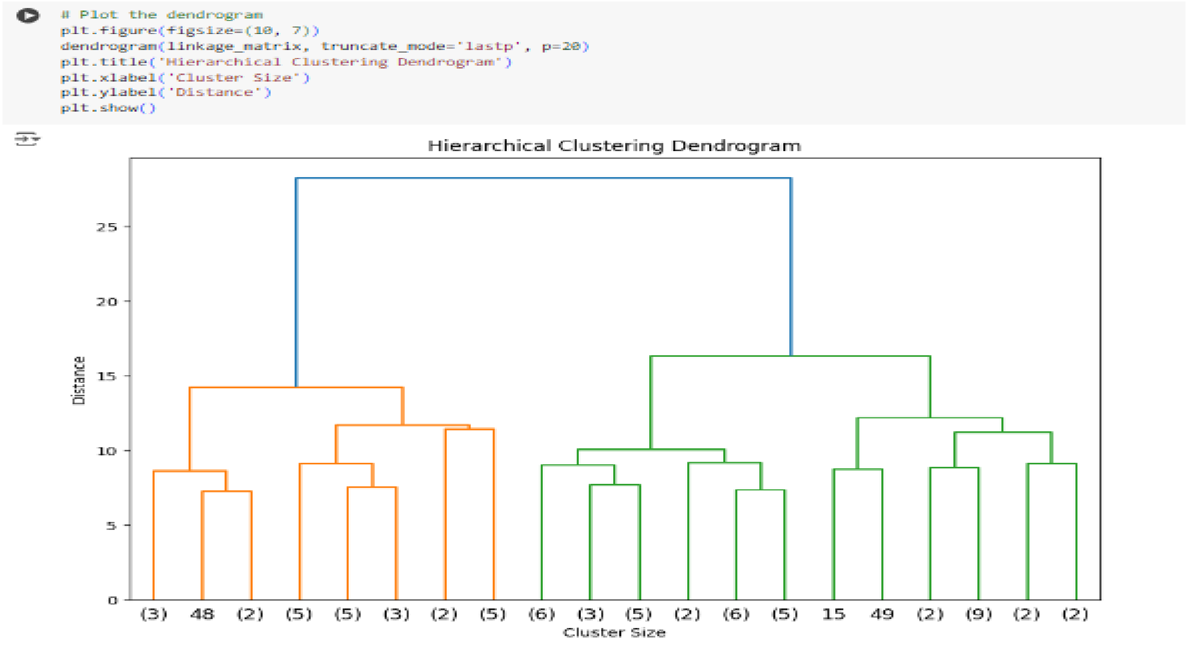
```
[ ] # Apply K-means clustering with the chosen number of clusters (e.g., 4)
    optimal_clusters = 4  # Replace with the optimal number found from the plot
    kmeans = KMeans(n_clusters=optimal_clusters, n_init=25, random_state=123)
    km_res = kmeans.fit_predict(sur_int_scaled)
```

```
[ ] # Add cluster labels to the original data
    df['Cluster'] = km_res
```

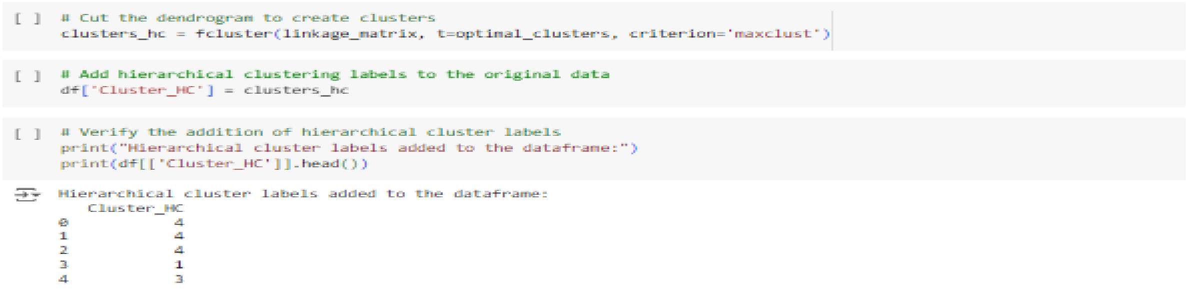
```
# Verify the addition of cluster labels
print("Cluster labels added to the dataframe:")
print(df[['Cluster']].head())
```

```
Cluster labels added to the dataframe:
   Cluster
0        2
1        2
2        2
3        0
4        3
```

```
[ ] # Perform hierarchical clustering
    # Compute the distance matrix
    from scipy.spatial.distance import pdist

    distance_matrix = pdist(sur_int_scaled, metric='euclidean')
    linkage_matrix = linkage(distance_matrix, method='ward')
```



```
# Plot the dendrogram
plt.figure(figsize=(10, 7))
dendrogram(linkage_matrix, truncate_mode='lastp', p=20)
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Cluster Size')
plt.ylabel('Distance')
plt.show()
```

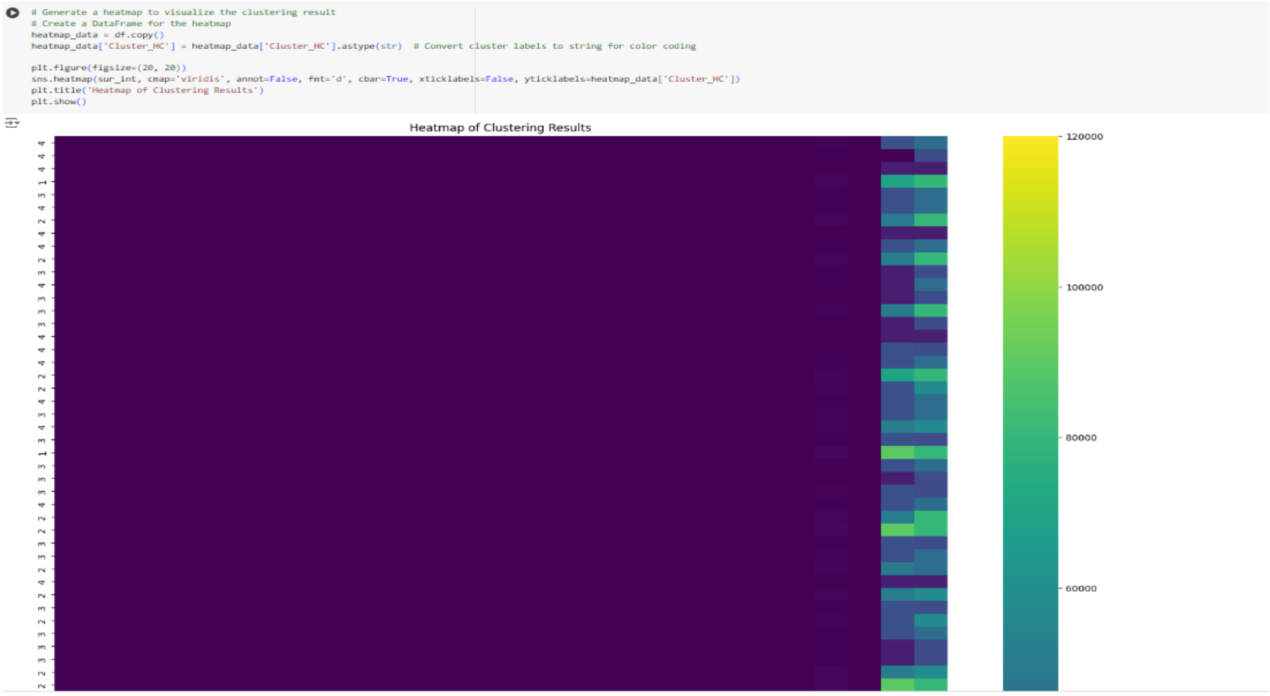


```
[ ]  # Cut the dendrogram to create clusters
     clusters_hc = fcluster(linkage_matrix, t=optimal_clusters, criterion='maxclust')
```

```
[ ]  # Add hierarchical clustering labels to the original data
     df['Cluster_HC'] = clusters_hc
```

```
[ ]  # Verify the addition of hierarchical cluster labels
     print("Hierarchical cluster labels added to the dataframe:")
     print(df[['Cluster_HC']].head())
```

```
Hierarchical cluster labels added to the dataframe:
   Cluster_HC
0           4
1           4
2           4
3           1
4           3
```

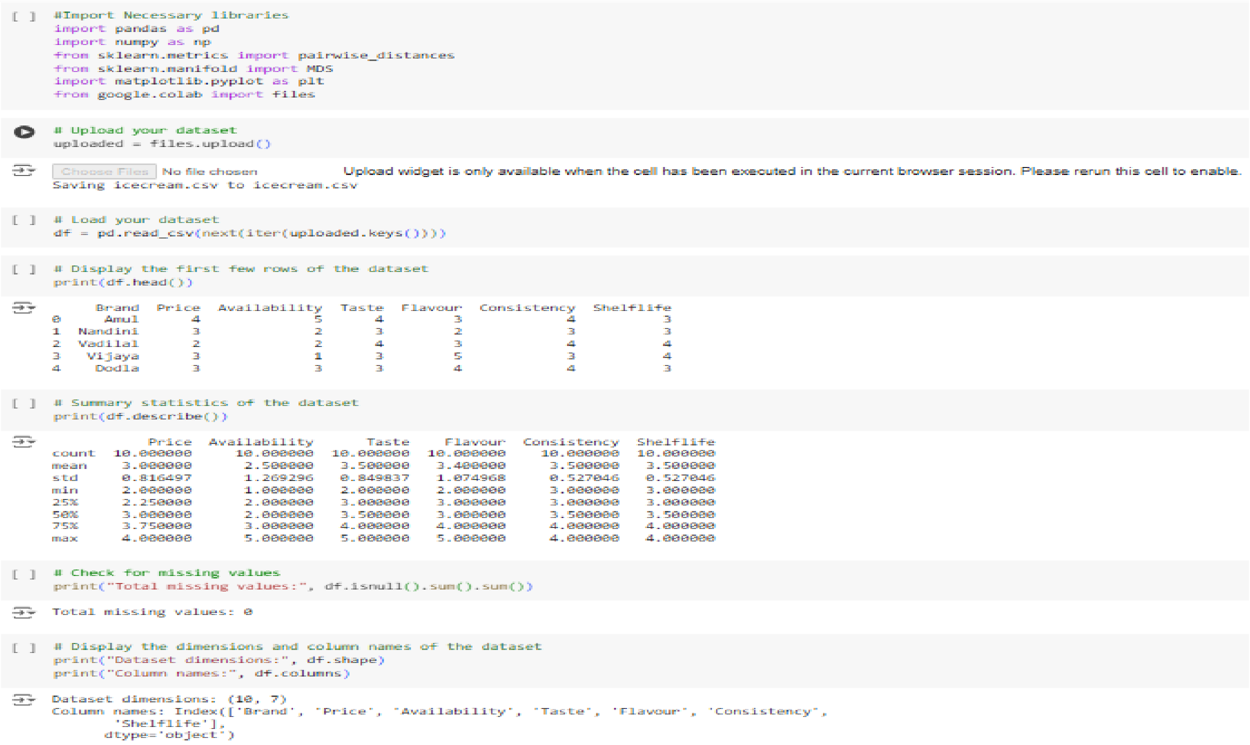


```
# Generate a heatmap to visualize the clustering result
# Create a DataFrame for the heatmap
heatmap_data = df.copy()
heatmap_data['Cluster_HC'] = heatmap_data['Cluster_HC'].astype(str)  # Convert cluster labels to string for color coding

plt.figure(figsize=(20, 20))
sns.heatmap(sur_int, cmap='viridis', annot=False, fmt='d', cbar=True, xticklabels=False, yticklabels=heatmap_data['Cluster_HC'])
plt.title('Heatmap of Clustering Results')
plt.show()
```

Interpretation:

The cluster analysis provides a detailed understanding of how different factors influence respondents' apartment preferences. Initially, the data was preprocessed and scaled to ensure uniformity across features. K-means clustering was then applied, with the optimal number of clusters determined to be 4 based on the silhouette score. This method grouped respondents into four distinct clusters, reflecting varying preferences and characteristics. Hierarchical clustering, using the Ward method, was also performed to validate these clusters, producing similar groupings. The hierarchical clustering dendrogram and heatmap visually represent these clusters, showing how respondents are grouped based on their preferences for factors such as proximity, facilities, and costs. The heatmap displays the clustering results with color-coded labels, helping to identify patterns and differences between clusters. Overall, this analysis effectively categorizes respondents into meaningful groups, facilitating a deeper understanding of the factors driving apartment preferences.

3.MDS



```
#Import Necessary libraries
import pandas as pd
import numpy as np
from sklearn.metrics import pairwise_distances
from sklearn.manifold import MDS
import matplotlib.pyplot as plt
from google.colab import files
```

```
# Upload your dataset
uploaded = files.upload()
```

```
Choose Files  No file chosen        Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving icecream.csv to icecream.csv
```

```
# Load your dataset
df = pd.read_csv(next(iter(uploaded.keys())))
```

```
# Display the first few rows of the dataset
print(df.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
```

```
# Summary statistics of the dataset
print(df.describe())
```

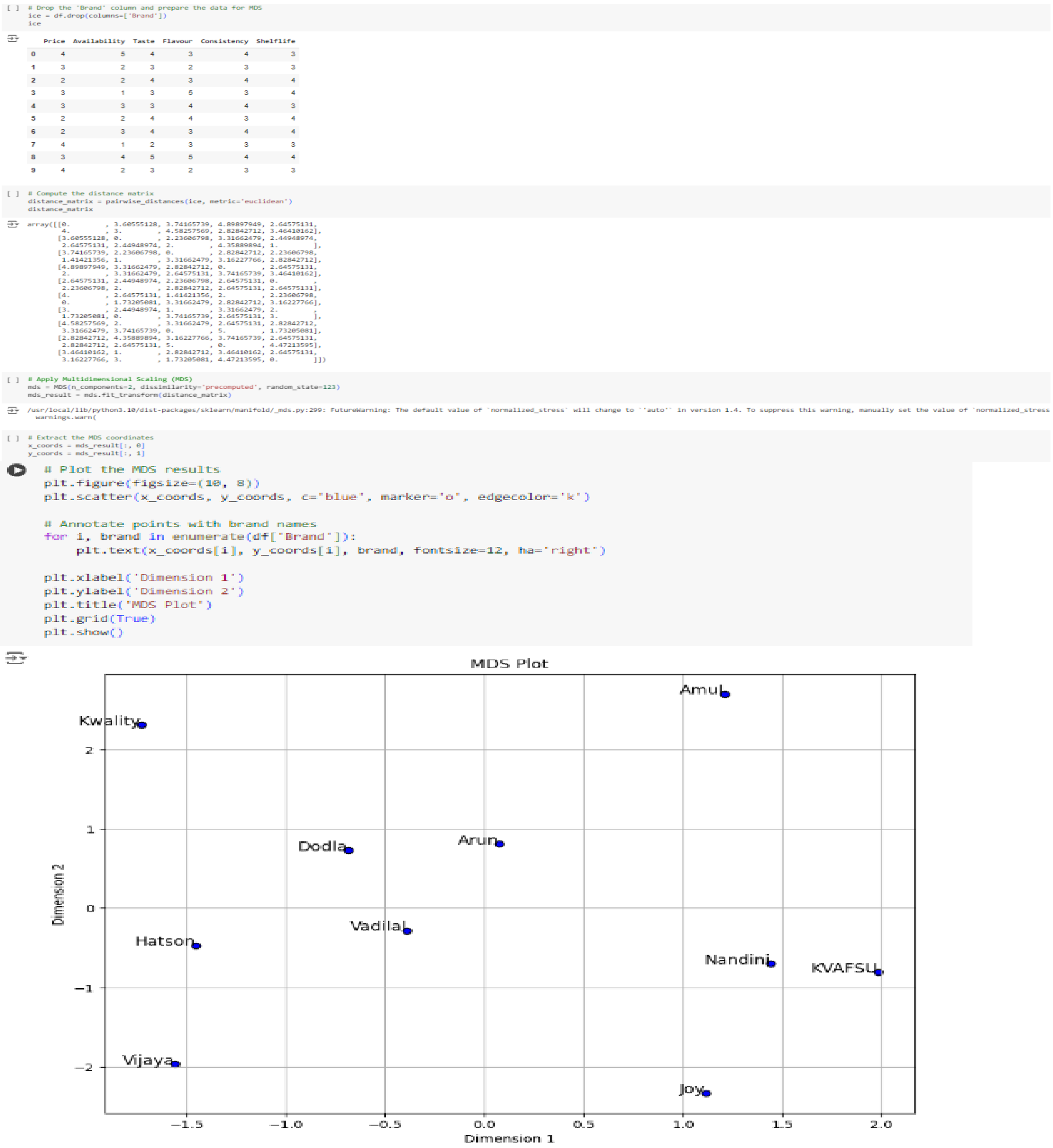
```
           Price  Availability      Taste    Flavour  Consistency  Shelflife
count  10.000000     10.000000  10.000000  10.000000    10.000000  10.000000
mean    3.000000      2.500000   3.500000   3.400000     3.500000   3.500000
std     0.816497      1.269296   0.849837   1.074968     0.527046   0.527046
min     2.000000      1.000000   2.000000   2.000000     3.000000   3.000000
25%     2.250000      2.000000   3.000000   3.000000     3.000000   3.000000
50%     3.000000      2.000000   3.500000   3.000000     3.500000   3.500000
75%     3.750000      3.000000   4.000000   4.000000     4.000000   4.000000
max     4.000000      5.000000   5.000000   5.000000     4.000000   4.000000
```

```
# Check for missing values
print("Total missing values:", df.isnull().sum().sum())
```

```
Total missing values: 0
```

```
# Display the dimensions and column names of the dataset
print("Dataset dimensions:", df.shape)
print("Column names:", df.columns)
```

```
Dataset dimensions: (10, 7)
Column names: Index(['Brand', 'Price', 'Availability', 'Taste', 'Flavour', 'Consistency',
       'Shelflife'],
      dtype='object')
```



```
[ ] # Drop the 'Brand' column and prepare the data for MDS
    ice = df.drop(columns=['Brand'])
    ice
```

	Price	Availability	Taste	Flavour	Consistency	Shelflife
0	4	5	4	3	4	3
1	3	2	3	2	3	3
2	2	2	4	3	4	4
3	3	1	3	5	3	4
4	3	3	3	4	4	3
5	2	2	4	4	3	4
6	2	3	4	3	4	4
7	4	1	2	3	3	3
8	3	4	5	5	4	4
9	4	2	3	2	3	3

```
[ ] # Compute the distance matrix
    distance_matrix = pairwise_distances(ice, metric='euclidean')
    distance_matrix
```

```
array([[0.        , 3.60555128, 3.74165739, 4.89897949, 2.64575131,
        4.        , 3.        , 4.58257569, 2.82842712, 3.46410162],
       [3.60555128, 0.        , 2.23606798, 3.31662479, 2.44948974,
        2.64575131, 2.44948974, 2.        , 4.35889894, 1.        ],
       [3.74165739, 2.23606798, 0.        , 2.82842712, 2.23606798,
        1.41421356, 1.        , 3.31662479, 3.16227766, 2.82842712],
       [4.89897949, 3.31662479, 2.82842712, 0.        , 2.64575131,
        2.        , 3.31662479, 2.64575131, 3.74165739, 3.46410162],
       [2.64575131, 2.44948974, 2.23606798, 2.64575131, 0.        ,
        2.23606798, 2.        , 2.82842712, 2.64575131, 2.64575131],
       [4.        , 2.64575131, 1.41421356, 2.        , 2.23606798,
        0.        , 1.73205081, 3.31662479, 2.82842712, 3.16227766],
       [3.        , 2.44948974, 1.        , 3.31662479, 2.        ,
        1.73205081, 0.        , 3.74165739, 2.64575131, 3.        ],
       [4.58257569, 2.        , 3.31662479, 2.64575131, 2.82842712,
        3.31662479, 3.74165739, 0.        , 5.        , 1.73205081],
       [2.82842712, 4.35889894, 3.16227766, 3.74165739, 2.64575131,
        2.82842712, 2.64575131, 5.        , 0.        , 4.47213595],
       [3.46410162, 1.        , 2.82842712, 3.46410162, 2.64575131,
        3.16227766, 3.        , 1.73205081, 4.47213595, 0.        ]])
```

```
[ ] # Apply Multidimensional Scaling (MDS)
    mds = MDS(n_components=2, dissimilarity='precomputed', random_state=123)
    mds_result = mds.fit_transform(distance_matrix)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/manifold/_mds.py:299: FutureWarning: The default value of `normalized_stress` will change to ``'auto'`` in version 1.4. To suppress this warning, manually set the value of `normalized_stress
  warnings.warn(
```

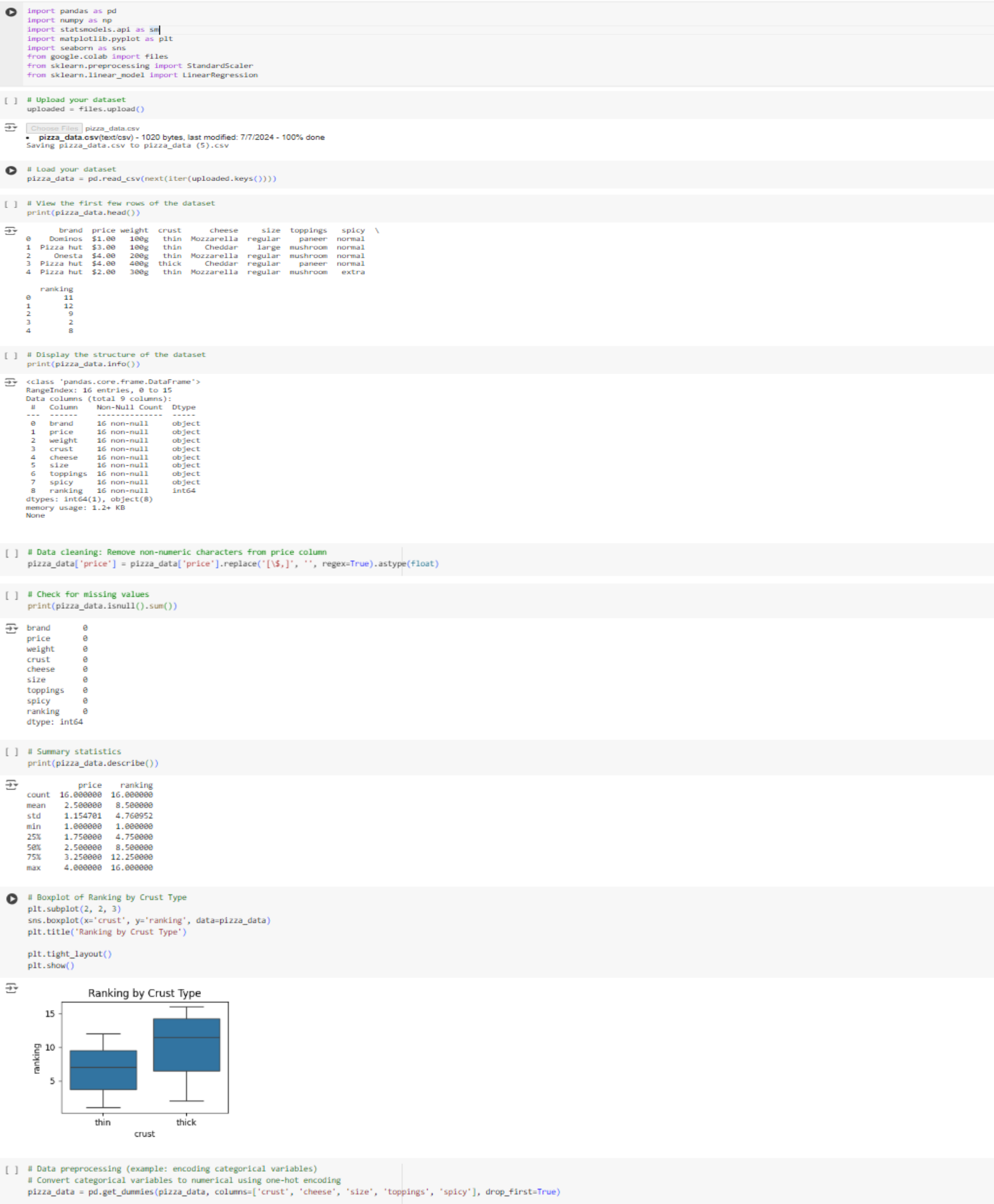
```
[ ] # Extract the MDS coordinates
    x_coords = mds_result[:, 0]
    y_coords = mds_result[:, 1]
```

```
# Plot the MDS results
plt.figure(figsize=(10, 8))
plt.scatter(x_coords, y_coords, c='blue', marker='o', edgecolor='k')

# Annotate points with brand names
for i, brand in enumerate(df['Brand']):
    plt.text(x_coords[i], y_coords[i], brand, fontsize=12, ha='right')

plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('MDS Plot')
plt.grid(True)
plt.show()
```

Interpretation:

Multidimensional Scaling (MDS) was applied to visualize the similarities among ice cream brands based on attributes such as price, availability, taste, and more. The Euclidean distance matrix, computed from these attributes, was used to position the brands in a 2D space. The MDS plot reveals the relative positions of each brand, with brands closer together indicating higher similarity in their feature profiles. For instance, brands with similar taste and consistency scores are located near each other in the plot, while those with differing attributes are positioned farther apart. This visualization helps in understanding the relative positioning of brands and identifying clusters or patterns based on their attributes.

4.Conjoint Analysis



```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import files
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
```

```
# Upload your dataset
uploaded = files.upload()
```

- **pizza_data.csv**(text/csv) - 1020 bytes, last modified: 7/7/2024 - 100% done

```
Saving pizza_data.csv to pizza_data (5).csv
```

```
# Load your dataset
pizza_data = pd.read_csv(next(iter(uploaded.keys())))
```

```
# View the first few rows of the dataset
print(pizza_data.head())
```

```
       brand price weight  crust      cheese     size  toppings   spicy  \
0    Dominos $1.00   100g   thin  Mozzarella  regular    paneer  normal
1  Pizza hut $3.00   100g   thin     Cheddar    large  mushroom  normal
2     Onesta $4.00   200g   thin  Mozzarella  regular  mushroom  normal
3  Pizza hut $4.00   400g  thick     Cheddar  regular    paneer  normal
4  Pizza hut $2.00   300g   thin  Mozzarella  regular  mushroom   extra

   ranking
0       11
1       12
2        9
3        2
4        8
```

```
# Display the structure of the dataset
print(pizza_data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16 entries, 0 to 15
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   brand     16 non-null     object
 1   price     16 non-null     object
 2   weight    16 non-null     object
 3   crust     16 non-null     object
 4   cheese    16 non-null     object
 5   size      16 non-null     object
 6   toppings  16 non-null     object
 7   spicy     16 non-null     object
 8   ranking   16 non-null     int64
dtypes: int64(1), object(8)
memory usage: 1.2+ KB
None
```

```
# Data cleaning: Remove non-numeric characters from price column
pizza_data['price'] = pizza_data['price'].replace('[\$,]', '', regex=True).astype(float)
```

```
# Check for missing values
print(pizza_data.isnull().sum())
```

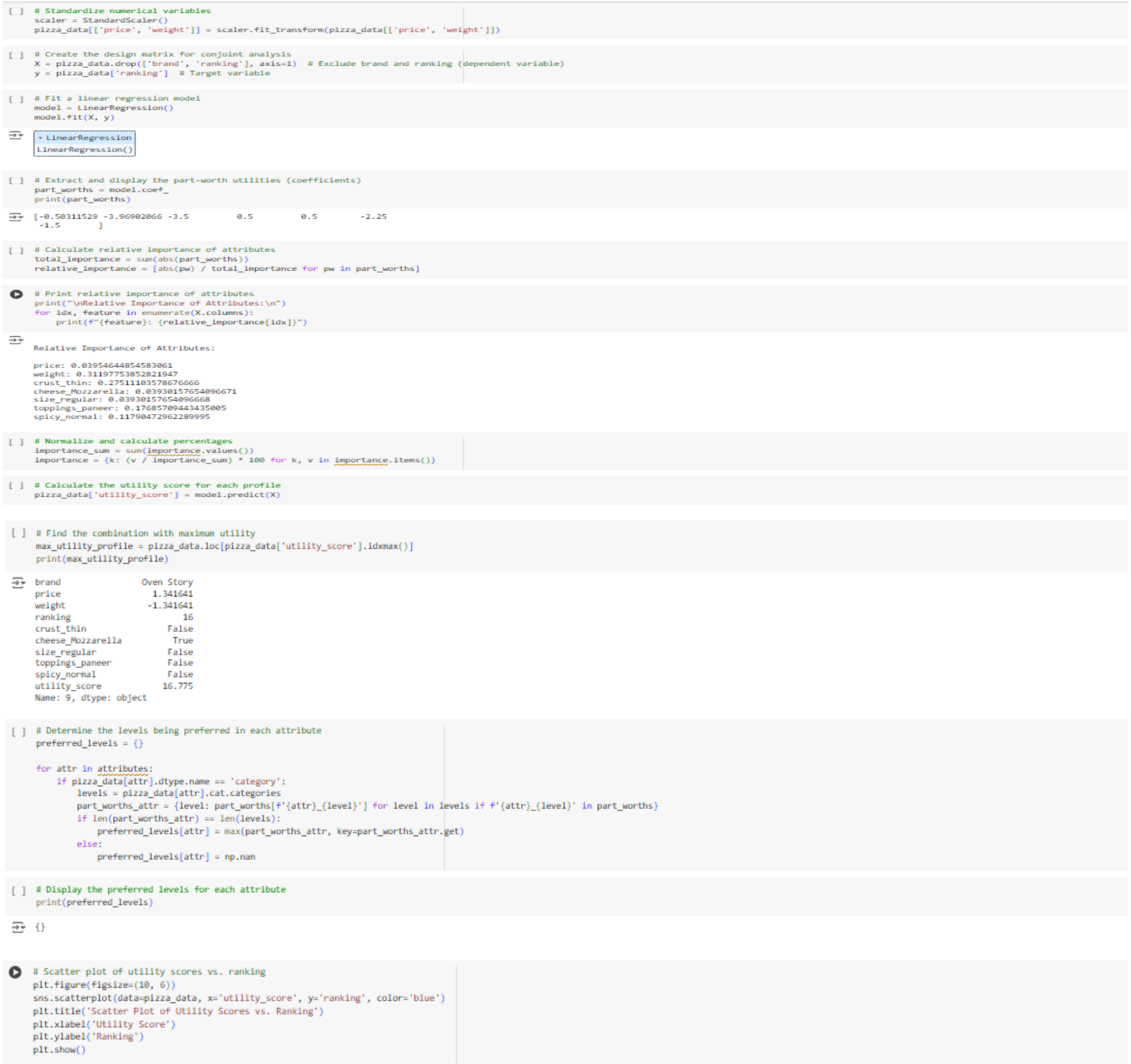
```
brand       0
price       0
weight      0
crust       0
cheese      0
size        0
toppings    0
spicy       0
ranking     0
dtype: int64
```

```
# Summary statistics
print(pizza_data.describe())
```

```
           price    ranking
count  16.000000  16.000000
mean    2.500000   8.500000
std     1.154701   4.760952
min     1.000000   1.000000
25%     1.750000   4.750000
50%     2.500000   8.500000
75%     3.250000  12.250000
max     4.000000  16.000000
```

```
# Boxplot of Ranking by Crust Type
plt.subplot(2, 2, 3)
sns.boxplot(x='crust', y='ranking', data=pizza_data)
plt.title('Ranking by Crust Type')

plt.tight_layout()
plt.show()
```

```
# Data preprocessing (example: encoding categorical variables)
# Convert categorical variables to numerical using one-hot encoding
pizza_data = pd.get_dummies(pizza_data, columns=['crust', 'cheese', 'size', 'toppings', 'spicy'], drop_first=True)
```



```
# Standardize numerical variables
scaler = StandardScaler()
pizza_data[['price', 'weight']] = scaler.fit_transform(pizza_data[['price', 'weight']])
```

```
# Create the design matrix for conjoint analysis
X = pizza_data.drop(['brand', 'ranking'], axis=1)  # Exclude brand and ranking (dependent variable)
y = pizza_data['ranking']  # Target variable
```

```
# Fit a linear regression model
model = LinearRegression()
model.fit(X, y)
```

```
▾ LinearRegression
LinearRegression()
```

```
# Extract and display the part-worth utilities (coefficients)
part_worths = model.coef_
print(part_worths)
```

```
[-0.50311529 -3.96902066 -3.5         0.5         0.5        -2.25
 -1.5       ]
```

```
# Calculate relative importance of attributes
total_importance = sum(abs(part_worths))
relative_importance = [abs(pw) / total_importance for pw in part_worths]
```

```
# Print relative importance of attributes
print("\nRelative Importance of Attributes:\n")
for idx, feature in enumerate(X.columns):
    print(f"{feature}: {relative_importance[idx]}")
```

```

Relative Importance of Attributes:

price: 0.03954644854583061
weight: 0.31197753852821947
crust_thin: 0.27511103578676666
cheese_Mozzarella: 0.03930157654096671
size_regular: 0.03930157654096668
toppings_paneer: 0.17685709443435005
spicy_normal: 0.11790472962289995
```

```
# Normalize and calculate percentages
importance_sum = sum(importance.values())
importance = {k: (v / importance_sum) * 100 for k, v in importance.items()}
```

```
# Calculate the utility score for each profile
pizza_data['utility_score'] = model.predict(X)
```

```
# Find the combination with maximum utility
max_utility_profile = pizza_data.loc[pizza_data['utility_score'].idxmax()]
print(max_utility_profile)
```

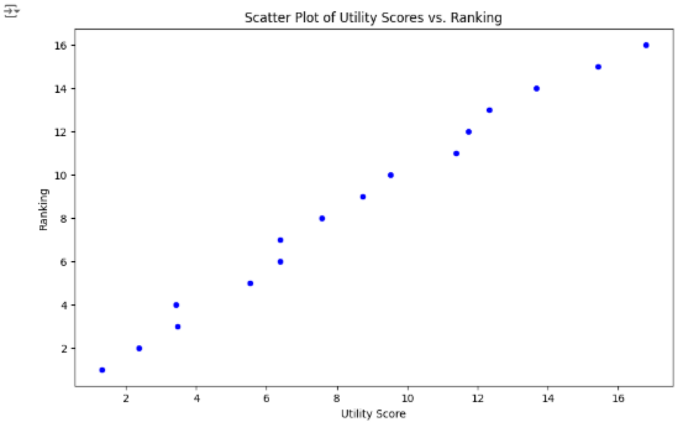
```
brand                 Oven Story
price                   1.341641
weight                 -1.341641
ranking                       16
crust_thin                 False
cheese_Mozzarella           True
size_regular               False
toppings_paneer            False
spicy_normal               False
utility_score             16.775
Name: 9, dtype: object
```

```
# Determine the levels being preferred in each attribute
preferred_levels = {}

for attr in attributes:
    if pizza_data[attr].dtype.name == 'category':
        levels = pizza_data[attr].cat.categories
        part_worths_attr = {level: part_worths[f'{attr}_{level}'] for level in levels if f'{attr}_{level}' in part_worths}
        if len(part_worths_attr) == len(levels):
            preferred_levels[attr] = max(part_worths_attr, key=part_worths_attr.get)
        else:
            preferred_levels[attr] = np.nan
```

```
# Display the preferred levels for each attribute
print(preferred_levels)
```

```
{}
```

```
# Scatter plot of utility scores vs. ranking
plt.figure(figsize=(10, 6))
sns.scatterplot(data=pizza_data, x='utility_score', y='ranking', color='blue')
plt.title('Scatter Plot of Utility Scores vs. Ranking')
plt.xlabel('Utility Score')
plt.ylabel('Ranking')
plt.show()
```



Interpretation:

The conjoint analysis of the pizza dataset shows that the most influential attributes affecting consumer rankings are weight and crust type, with weight having the highest relative importance of 31.20%, followed by crust type with 27.51%. The part-worth utilities reveal that price and cheese type have lower influence, with relative importances of 3.95% and 3.93%, respectively. The pizza profile with the highest utility score of 16.78, which includes a price of $1.34 and a weight of -1.34 (standardized), is identified as the most preferred. The scatter plot of utility scores versus rankings confirms a positive relationship, highlighting how the model effectively captures key factors driving consumer preferences.