VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A4- Multivariate Analysis and Business Analytics Applications (Part – A)

ADHYAYAN AMIT JAIN

V01109421

Date of Submission: 08-07-2024

CONTENTS

Exploring Data Dimensions: Principal Component Analysis and Factor Analysis of 'Survey.csv'

INTRODUCTION

In today's data-driven world, understanding the complex interplay of variables within datasets is crucial for informed decision-making and strategic planning. Principal Component Analysis (PCA) and Factor Analysis are powerful statistical techniques that allow us to uncover hidden patterns, reduce dimensionality, and extract meaningful insights from large and multidimensional datasets.

The dataset 'Survey.csv' presents an opportunity to explore and analyze a wealth of information collected through structured surveys. By applying PCA, we can identify the underlying structure of correlations among variables, effectively reducing the dataset's dimensionality while preserving the variance that matters most. Additionally, Factor Analysis enables us to uncover latent constructs or dimensions that explain the relationships between observed variables, providing deeper insights into the data's structure and facilitating a more nuanced understanding of key drivers.

Through this analysis, we aim to not only uncover the principal components that explain the majority of variance within 'Survey.csv' but also to identify critical factors that influence outcomes or perceptions captured in the survey responses. By visualizing these findings through biplots and other graphical representations, we can effectively communicate complex relationships and patterns within the data, aiding stakeholders in making informed decisions and formulating targeted strategies.

This report explores the implications and recommendations derived from PCA and Factor Analysis applied to 'Survey.csv'. It highlights the strategic importance of understanding data dimensions, emphasizes the practical applications of statistical techniques in real-world scenarios, and advocates for a data-driven approach to enhancing organizational decision-making processes.

OBJECTIVES

The primary objectives of this analysis are:

1. **Principal Component Analysis (PCA):**
    - **Identify Key Dimensions:** Utilize PCA to identify the principal components that capture the maximum variance within the dataset 'Survey.csv'. This will help in understanding which variables contribute most significantly to the overall variation and structure of the data.
    - **Dimension Reduction:** Reduce the dimensionality of the dataset while retaining as much variance as possible. By transforming correlated variables into a smaller set of uncorrelated components, PCA facilitates a more concise and manageable representation of the data.
2. **Factor Analysis:**
    - **Uncover Latent Constructs:** Apply Factor Analysis to uncover latent constructs or factors that explain correlations among observed variables in 'Survey.csv'. This will reveal underlying dimensions or themes that contribute to the patterns observed in the survey responses.
    - **Interpret Factor Loadings:** Interpret factor loadings to understand how variables load onto each factor, identifying which variables are most closely associated with each underlying dimension.
3. **Insights and Recommendations:**
    - **Visualize Relationships:** Visualize the results through biplots and other graphical representations to elucidate the relationships between variables and components/factors derived from PCA and Factor Analysis.
    - **Provide Actionable Insights:** Derive actionable insights that can inform strategic decision-making processes. These insights may include identifying key drivers of survey responses, highlighting areas for improvement, or uncovering hidden trends and patterns within the dataset.

4. **Validation and Robustness:**
    - **Ensure Data Quality:** Validate findings and ensure robustness in analysis techniques to enhance the reliability and validity of the conclusions drawn from PCA and Factor Analysis.
    - **Iterative Improvement:** Adopt an iterative approach to analysis, refining methodologies based on initial findings to uncover deeper insights and enhance the overall quality of analysis outcomes.

By achieving these objectives, this analysis aims to provide stakeholders with a comprehensive understanding of the underlying structure and dimensions within 'Survey.csv'. It seeks to empower decision-makers with actionable insights derived from advanced statistical techniques, facilitating informed decisions and strategic initiatives based on data-driven evidence.

BUSINESS SIGNIFICANCE

Understanding the business significance of applying Principal Component Analysis (PCA) and Factor Analysis to the dataset 'Survey.csv' is crucial for leveraging data-driven insights to drive organizational strategies and decisions. The following points highlight the practical implications and benefits:

1. **Strategic Insights from Data Dimensions:**
    - **Identifying Key Drivers:** PCA enables us to identify the principal components that explain the majority of variance in 'Survey.csv'. By understanding these key drivers, organizations can prioritize efforts and resources towards areas that have the most significant impact on outcomes measured in the survey.
    - **Optimizing Resource Allocation:** By focusing on the dimensions that matter most, PCA helps in optimizing resource allocation, whether it's improving customer satisfaction, enhancing product features, or refining operational processes based on identified performance indicators.
2. **Enhanced Decision-Making Processes:**
    - **Data-Driven Decision Support:** Factor Analysis uncovers latent constructs and underlying dimensions within the data. This

provides a structured framework for decision-makers to interpret complex relationships and make informed decisions aligned with organizational goals and objectives.

- **Mitigating Risks:** Understanding the factors influencing survey responses or outcomes helps in identifying potential risks and implementing proactive measures to mitigate them. This proactive approach enhances organizational resilience and responsiveness to market dynamics.

3. **Improving Customer Insights and Satisfaction:**
    - **Segmentation and Personalization:** PCA and Factor Analysis facilitate customer segmentation based on identified dimensions such as preferences, behaviors, or satisfaction factors. This segmentation enables targeted marketing strategies and personalized customer experiences, thereby enhancing overall satisfaction and loyalty.
    - **Feedback Loop Improvement:** By identifying and addressing key dimensions influencing customer feedback, organizations can continuously improve products and services, fostering long-term customer relationships and sustainable growth.
4. **Operational Efficiency and Effectiveness:**
    - **Process Optimization:** Insights from PCA and Factor Analysis can optimize internal processes by identifying inefficiencies or bottlenecks associated with specific dimensions. This optimization leads to enhanced operational efficiency and cost savings.
    - **Performance Monitoring:** Monitoring performance across identified dimensions allows for real-time adjustments and improvements, ensuring organizational agility and responsiveness in a competitive marketplace.
5. **Strategic Planning and Forecasting:**
    - **Forecasting and Planning:** Utilizing PCA and Factor Analysis aids in predictive modeling and scenario planning based on

identified trends and patterns. This foresight enables organizations to anticipate future trends, adapt strategies accordingly, and stay ahead of market shifts.

In conclusion, the application of PCA and Factor Analysis on 'Survey.csv' offers substantial business benefits by uncovering actionable insights, improving decision-making processes, enhancing customer satisfaction, optimizing operations, and supporting strategic planning initiatives. By leveraging these advanced analytical techniques, organizations can gain a competitive edge, foster innovation, and achieve sustainable growth in today's dynamic business environment.

RESULTS AND INTERPRETATIONS

R Language

Here's a detailed step-by-step analysis and interpretation of each line of the code, structured for the best understanding:

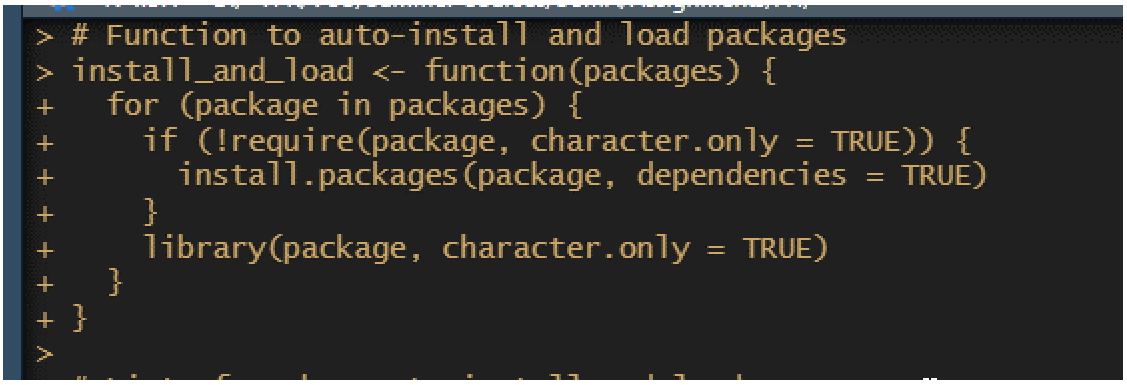
```
r

# Function to auto-install and load packages
install_and_load <- function(packages) {
  for (package in packages) {
    # Check if the package is already installed; if not, install it
    if (!require(package, character.only = TRUE)) {
      install.packages(package, dependencies = TRUE)
    }
    # Load the package into the R session
    library(package, character.only = TRUE)
  }
}
```

Explanation:

1. **Function Definition:**
   - Defines a function named `install_and_load` that takes a vector of package names as input.
2. **For Loop:**
   - Iterates over each package in the provided vector.
3. **Check and Install:**
   - Uses `require` to check if the package is installed.
   - If not installed, `install.packages` installs the package with dependencies.
4. **Load Package:**
   - Uses `library` to load the package into the R session.

Interpretation:

This function automates the process of checking for the installation of required packages, installing any that are missing, and loading them into the current R session, ensuring that all necessary packages are available for use.



```
> # 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)
+   }
+ }
>
```

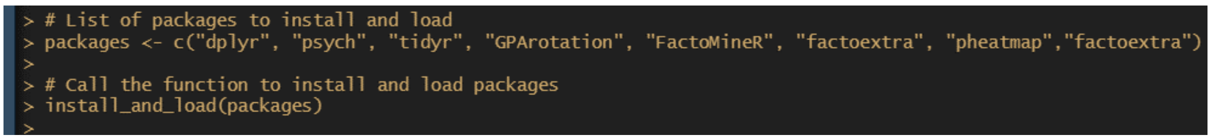
```
r

# List of packages to install and load
packages <- c("dplyr", "psych", "tidyr", "GPArotation", "FactoMineR",
"factoextra", "pheatmap")

# Call the function to install and load packages
install_and_load(packages)
```

Explanation:

1. **Packages Vector:**
   - Creates a vector named `packages` containing the names of packages to be installed and loaded.
2. **Function Call:**
   - Calls the `install_and_load` function with the `packages` vector as an argument.

Interpretation:

This ensures that all the specified packages are installed and loaded, which are necessary for data manipulation, statistical analysis, and visualization.



```
> # List of packages to install and load
> packages <- c("dplyr", "psych", "tidyr", "GPArotation", "FactoMineR", "factoextra", "pheatmap","factoextra")
>
> # Call the function to install and load packages
> install_and_load(packages)
>
```

```
r

# Set the working directory to the specified path
setwd('D:\\#YPR\\VCU\\Summer Courses\\SCMA\\Assignments\\A4')
```

Explanation:

1. **Set Working Directory:**
   - Sets the working directory to the specified path using `setwd`.

Interpretation:

This line sets the current working directory to the specified path, ensuring that any file operations (like reading data) are performed in the correct directory.

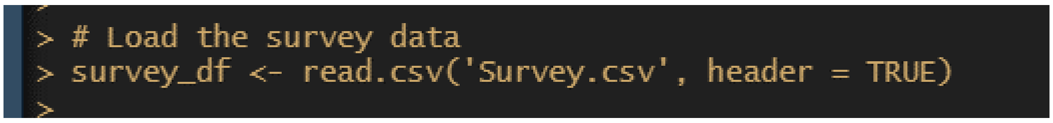
```
r

# Load the survey data from a CSV file
survey_df <- read.csv('Survey.csv', header = TRUE)
```

Explanation:

1. **Read CSV File:**
   - Reads the CSV file named 'Survey.csv' into a data frame called `survey_df` with `header = TRUE` indicating that the first row contains column names.

Interpretation:

This line loads the survey data into a data frame, making it available for further analysis.



```
> # Load the survey data
> survey_df <- read.csv('Survey.csv', header = TRUE)
>
```

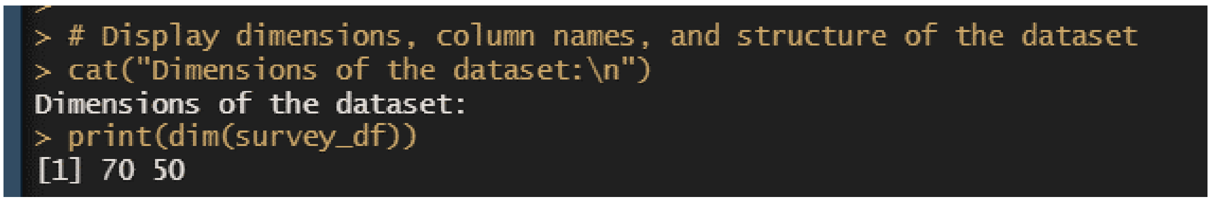
```
r

# Display the dimensions of the dataset
cat("Dimensions of the dataset:\n")
print(dim(survey_df))
```

Explanation:

1. **Print Dimensions:**
   - Uses `cat` to print a message.
   - Uses `dim` to print the dimensions (number of rows and columns) of `survey_df`.

Interpretation:

Displays the size of the dataset, helping to understand the scale of the data.



```
> # Display dimensions, column names, and structure of the dataset
> cat("Dimensions of the dataset:\n")
Dimensions of the dataset:
> print(dim(survey_df))
[1] 70 50
```

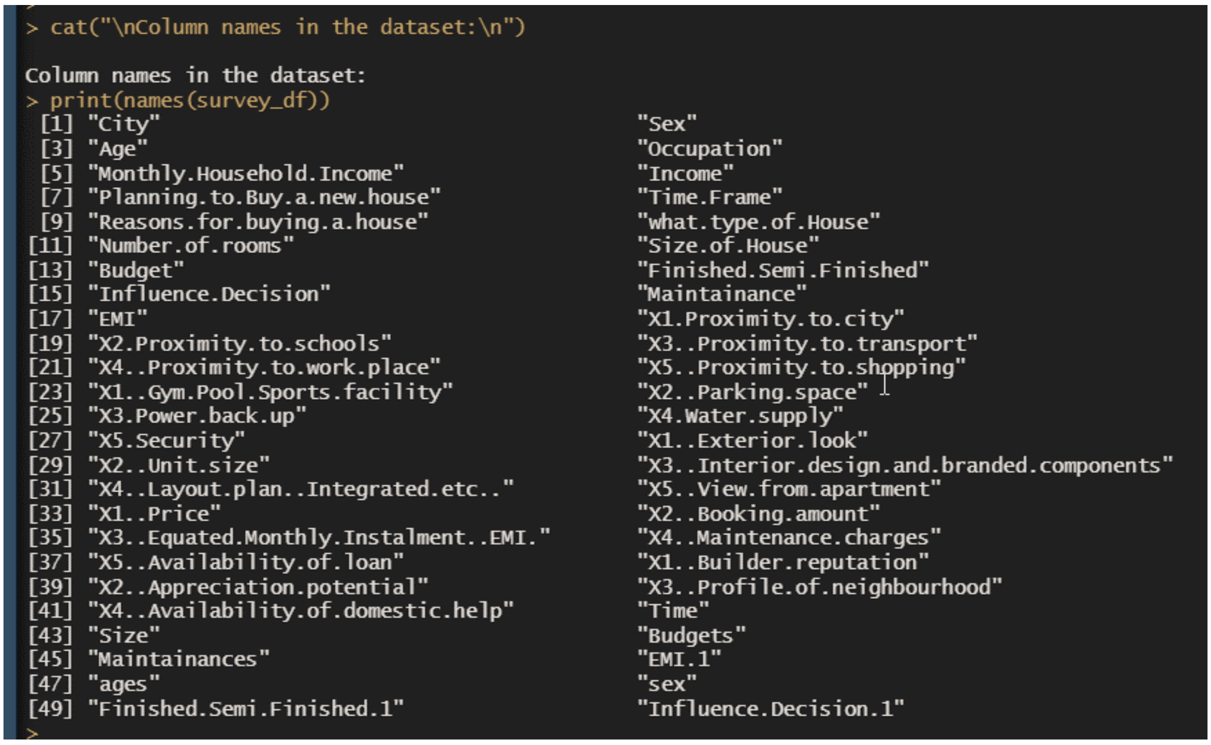
```
r

# Display the column names of the dataset
cat("\nColumn names in the dataset:\n")
print(names(survey_df))
```

Explanation:

1. **Print Column Names:**
   - Uses `cat` to print a message.
   - Uses `names` to print the column names of `survey_df`.

Interpretation:

Lists all the column names in the dataset, providing an overview of the variables available for analysis.



```
> cat("\nColumn names in the dataset:\n")

Column names in the dataset:
> print(names(survey_df))
 [1] "City"                                   "Sex"
 [3] "Age"                                    "Occupation"
 [5] "Monthly.Household.Income"               "Income"
 [7] "Planning.to.Buy.a.new.house"            "Time.Frame"
 [9] "Reasons.for.buying.a.house"             "what.type.of.House"
[11] "Number.of.rooms"                        "Size.of.House"
[13] "Budget"                                 "Finished.Semi.Finished"
[15] "Influence.Decision"                     "Maintainance"
[17] "EMI"                                    "X1.Proximity.to.city"
[19] "X2.Proximity.to.schools"                "X3..Proximity.to.transport"
[21] "X4..Proximity.to.work.place"            "X5..Proximity.to.shopping"
[23] "X1..Gym.Pool.Sports.facility"           "X2..Parking.space"
[25] "X3.Power.back.up"                       "X4.Water.supply"
[27] "X5.Security"                            "X1..Exterior.look"
[29] "X2..Unit.size"                          "X3..Interior.design.and.branded.components"
[31] "X4..Layout.plan..Integrated.etc.."      "X5..View.from.apartment"
[33] "X1..Price"                              "X2..Booking.amount"
[35] "X3..Equated.Monthly.Instalment..EMI."   "X4..Maintenance.charges"
[37] "X5..Availability.of.loan"               "X1..Builder.reputation"
[39] "X2..Appreciation.potential"             "X3..Profile.of.neighbourhood"
[41] "X4..Availability.of.domestic.help"      "Time"
[43] "Size"                                   "Budgets"
[45] "Maintainances"                          "EMI.1"
[47] "ages"                                   "sex"
[49] "Finished.Semi.Finished.1"               "Influence.Decision.1"
>
```

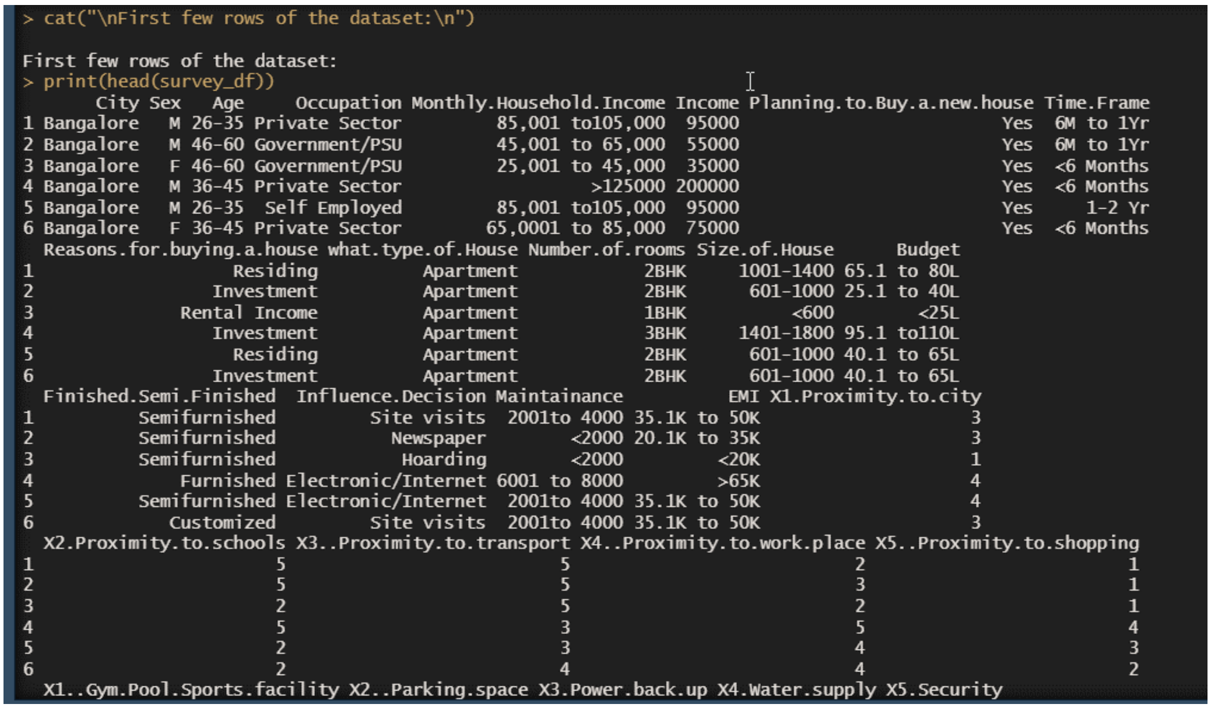
```
r

# Display the first few rows of the dataset
cat("\nFirst few rows of the dataset:\n")
print(head(survey_df))
```

Explanation:

1. **Print First Few Rows:**
   - Uses `cat` to print a message.
   - Uses `head` to print the first few rows of `survey_df`.

Interpretation:

Shows a preview of the data, giving an initial look at the values and structure.



```
> cat("\nFirst few rows of the dataset:\n")

First few rows of the dataset:
> print(head(survey_df))
       City Sex   Age      Occupation Monthly.Household.Income Income Planning.to.Buy.a.new.house Time.Frame
1 Bangalore   M 26-35 Private Sector        85,001 to105,000  95000                         Yes  6M to 1Yr
2 Bangalore   M 46-60 Government/PSU        45,001 to 65,000  55000                         Yes  6M to 1Yr
3 Bangalore   F 46-60 Government/PSU        25,001 to 45,000  35000                         Yes  <6 Months
4 Bangalore   M 36-45 Private Sector                 >125000 200000                         Yes  <6 Months
5 Bangalore   M 26-35  Self Employed        85,001 to105,000  95000                         Yes     1-2 Yr
6 Bangalore   F 36-45 Private Sector       65,0001 to 85,000  75000                         Yes  <6 Months
  Reasons.for.buying.a.house what.type.of.House Number.of.rooms Size.of.House      Budget
1                   Residing          Apartment            2BHK     1001-1400 65.1 to 80L
2                 Investment          Apartment            2BHK      601-1000 25.1 to 40L
3              Rental Income          Apartment            1BHK          <600        <25L
4                 Investment          Apartment            3BHK     1401-1800 95.1 to110L
5                   Residing          Apartment            2BHK      601-1000 40.1 to 65L
6                 Investment          Apartment            2BHK      601-1000 40.1 to 65L
  Finished.Semi.Finished  Influence.Decision Maintainance          EMI X1.Proximity.to.city
1         Semifurnished          Site visits  2001to 4000 35.1K to 50K                    3
2         Semifurnished            Newspaper        <2000 20.1K to 35K                    3
3         Semifurnished             Hoarding        <2000         <20K                    1
4             Furnished Electronic/Internet 6001 to 8000         >65K                    4
5         Semifurnished Electronic/Internet  2001to 4000 35.1K to 50K                    4
6            Customized          Site visits  2001to 4000 35.1K to 50K                    3
  X2.Proximity.to.schools X3..Proximity.to.transport X4..Proximity.to.work.place X5..Proximity.to.shopping
1                       5                          5                           2                         1
2                       5                          5                           3                         1
3                       2                          5                           2                         1
4                       5                          3                           5                         4
5                       2                          3                           4                         3
6                       2                          4                           4                         2
  X1..Gym.Pool.Sports.facility X2..Parking.space X3.Power.back.up X4.Water.supply X5.Security
```

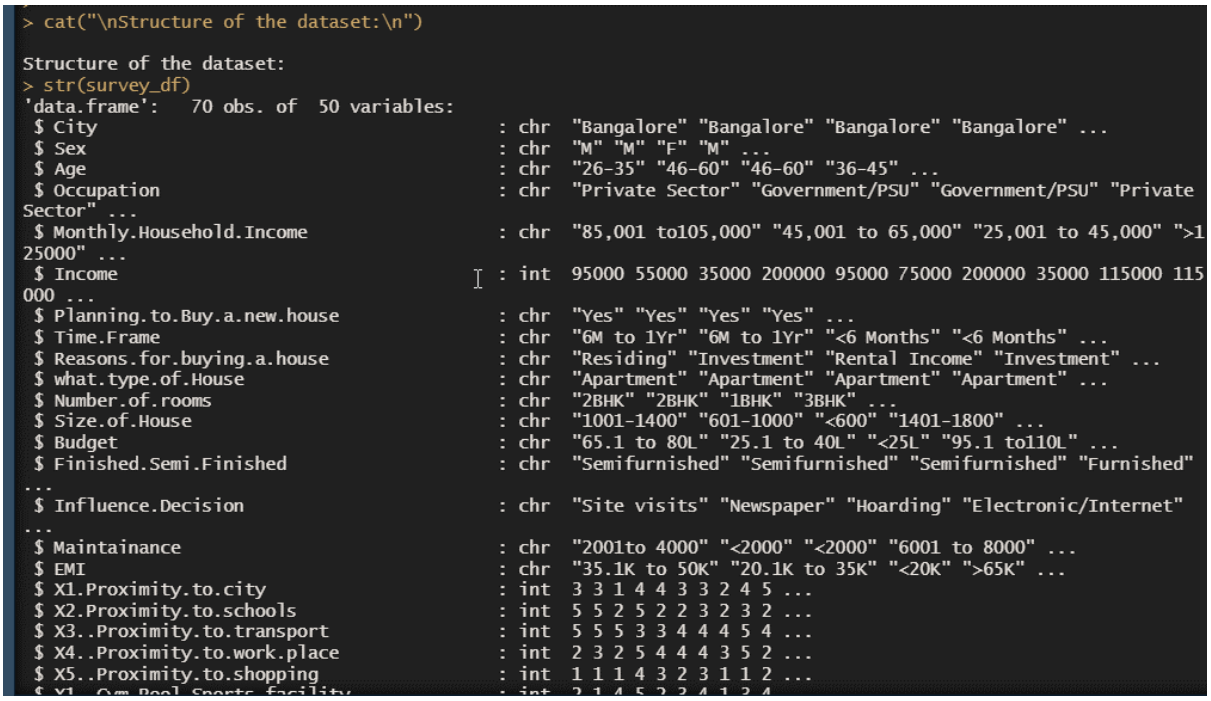
```
r

# Display the structure of the dataset
cat("\nStructure of the dataset:\n")
str(survey_df)
```

Explanation:

1. **Print Structure:**
   - Uses `cat` to print a message.
   - Uses `str` to print the structure of `survey_df`, showing data types and a preview of the data.

Interpretation:

Provides detailed information on the dataset's structure, including data types and a summary of each column.



```
> cat("\nStructure of the dataset:\n")

Structure of the dataset:
> 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" ">1
25000" ...
 $ Income                                  : int  95000 55000 35000 200000 95000 75000 200000 35000 115000 115
000 ...
 $ 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
```

r

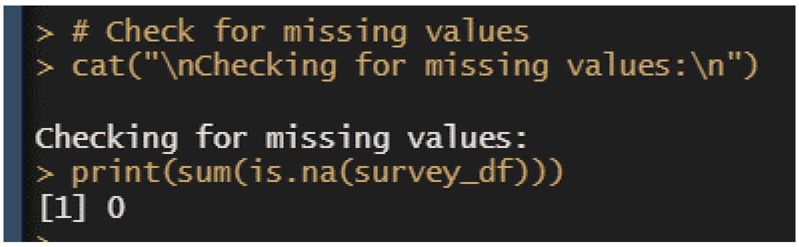
```
# Check for missing values in the dataset
cat("\nChecking for missing values:\n")
print(sum(is.na(survey_df)))
```

Explanation:

1. **Check Missing Values:**
   - Uses `cat` to print a message.
   - Uses `is.na` to check for missing values in `survey_df`.
   - Uses `sum` to count the total number of missing values.

Interpretation:

Identifies the presence and quantity of missing values, which may need to be addressed in subsequent analyses.



```
> # Check for missing values
> cat("\nChecking for missing values:\n")

Checking for missing values:
> print(sum(is.na(survey_df)))
[1] 0
```

r

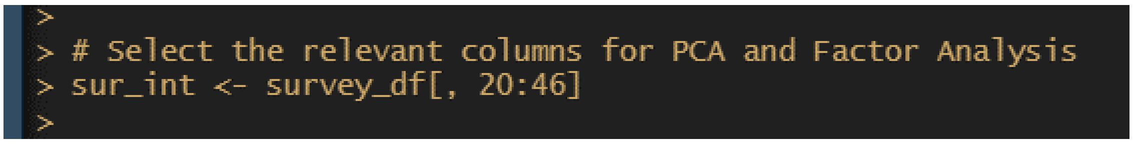
```
# Select the relevant columns for PCA and Factor Analysis (columns 20 to
46)
sur_int <- survey_df[, 20:46]
```

Explanation:

1. **Select Columns:**
    - Selects columns 20 to 46 from `survey_df` and stores them in a new data frame `sur_int`.

Interpretation:

Extracts the subset of data relevant for Principal Component Analysis (PCA) and Factor Analysis, focusing on specific variables.



```
>
> # Select the relevant columns for PCA and Factor Analysis
> sur_int <- survey_df[, 20:46]
>
```

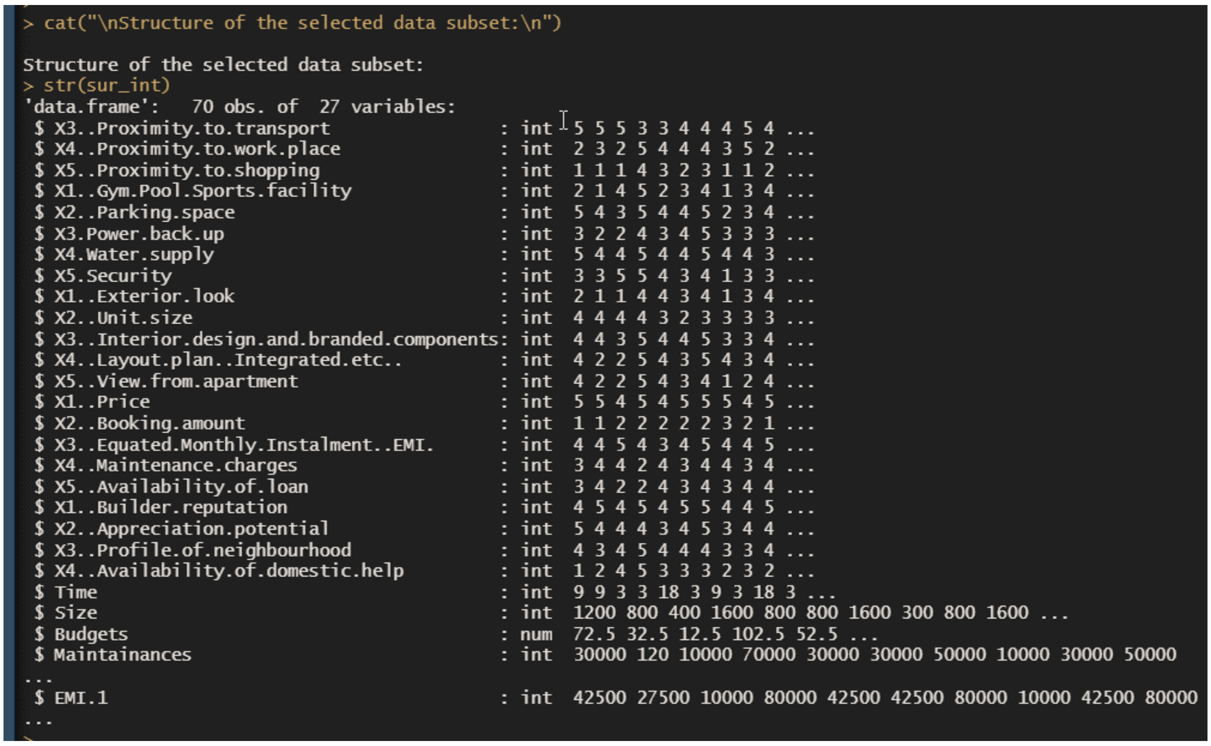
```
r

# Display the structure of the selected data subset
cat("\nStructure of the selected data subset:\n")
str(sur_int)
```

Explanation:

1. **Print Structure:**
    - Uses `cat` to print a message.
    - Uses `str` to print the structure of `sur_int`.

Interpretation:

Provides detailed information on the selected subset, ensuring it has the correct variables and structure for analysis.



```
> cat("\nStructure of the selected data subset:\n")

Structure of the selected data subset:
> 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
 ...
```

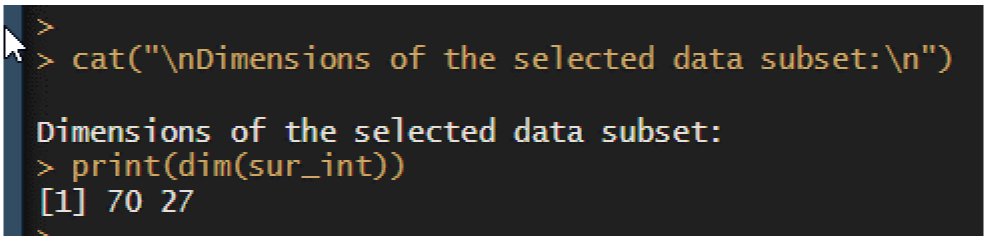
```
r

# Display the dimensions of the selected data subset
cat("\nDimensions of the selected data subset:\n")
print(dim(sur_int))
```

Explanation:

1. **Print Dimensions:**
   - Uses `cat` to print a message.
   - Uses `dim` to print the dimensions of `sur_int`.

Interpretation:

Confirms the size of the selected data subset, verifying that the correct number of rows and columns were selected.



```
>
> cat("\nDimensions of the selected data subset:\n")

Dimensions of the selected data subset:
> print(dim(sur_int))
[1] 70 27
```

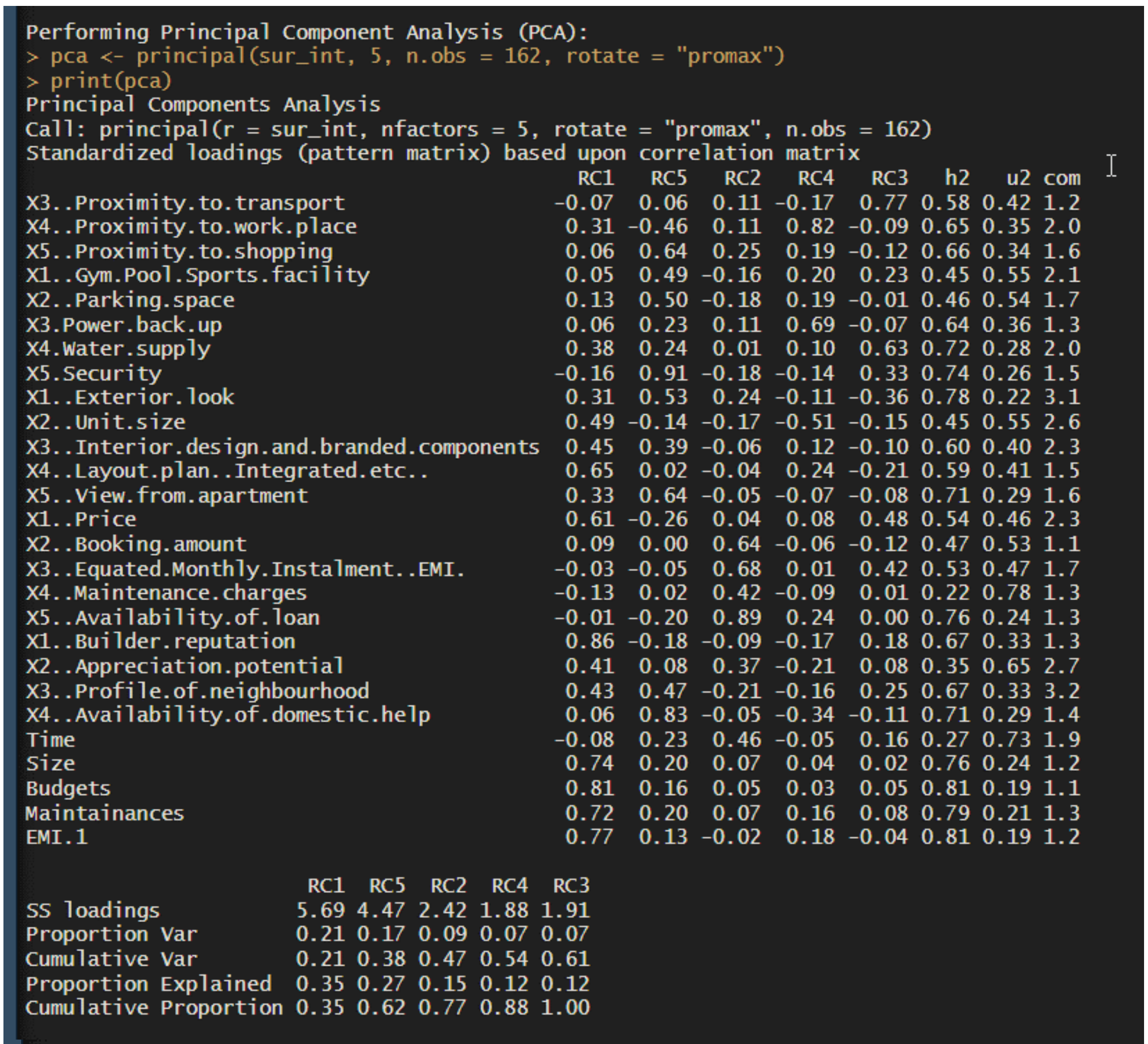
```
r

# Perform Principal Component Analysis (PCA)
cat("\nPerforming Principal Component Analysis (PCA):\n")
pca <- principal(sur_int, 5, n.obs = 162, rotate = "promax")
print(pca)
```

Explanation:

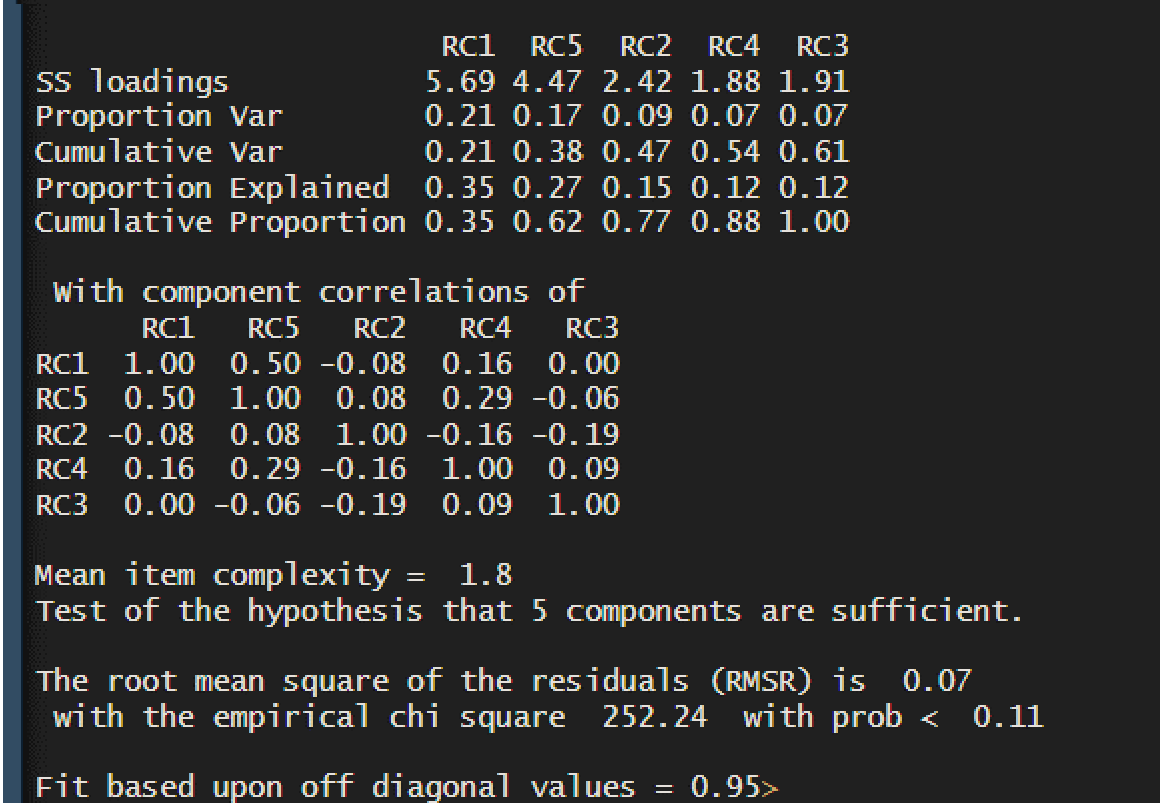
1. **Print Message:**
   - Uses `cat` to print a message.
2. **Perform PCA:**
   - Uses `principal` from the `psych` package to perform PCA on `sur_int`.
   - Specifies 5 components, with 162 observations, and a "promax" rotation.
3. **Print PCA Results:**
   - Uses `print` to display the PCA results.

Interpretation:

Performs PCA to reduce the dimensionality of the data and identify key components. The output includes loadings and explained variance for each principal component.



```
Performing Principal Component Analysis (PCA):
> pca <- principal(sur_int, 5, n.obs = 162, rotate = "promax")
> print(pca)
Principal Components Analysis
Call: principal(r = sur_int, nfactors = 5, rotate = "promax", n.obs = 162)
Standardized loadings (pattern matrix) based upon correlation matrix
                                           RC1   RC5   RC2   RC4   RC3   h2   u2 com
X3..Proximity.to.transport               -0.07  0.06  0.11 -0.17  0.77 0.58 0.42 1.2
X4..Proximity.to.work.place               0.31 -0.46  0.11  0.82 -0.09 0.65 0.35 2.0
X5..Proximity.to.shopping                 0.06  0.64  0.25  0.19 -0.12 0.66 0.34 1.6
X1..Gym.Pool.Sports.facility              0.05  0.49 -0.16  0.20  0.23 0.45 0.55 2.1
X2..Parking.space                         0.13  0.50 -0.18  0.19 -0.01 0.46 0.54 1.7
X3.Power.back.up                          0.06  0.23  0.11  0.69 -0.07 0.64 0.36 1.3
X4.Water.supply                           0.38  0.24  0.01  0.10  0.63 0.72 0.28 2.0
X5.Security                              -0.16  0.91 -0.18 -0.14  0.33 0.74 0.26 1.5
X1..Exterior.look                         0.31  0.53  0.24 -0.11 -0.36 0.78 0.22 3.1
X2..Unit.size                             0.49 -0.14 -0.17 -0.51 -0.15 0.45 0.55 2.6
X3..Interior.design.and.branded.components 0.45  0.39 -0.06  0.12 -0.10 0.60 0.40 2.3
X4..Layout.plan..Integrated.etc..         0.65  0.02 -0.04  0.24 -0.21 0.59 0.41 1.5
X5..View.from.apartment                   0.33  0.64 -0.05 -0.07 -0.08 0.71 0.29 1.6
X1..Price                                 0.61 -0.26  0.04  0.08  0.48 0.54 0.46 2.3
X2..Booking.amount                        0.09  0.00  0.64 -0.06 -0.12 0.47 0.53 1.1
X3..Equated.Monthly.Instalment..EMI.     -0.03 -0.05  0.68  0.01  0.42 0.53 0.47 1.7
X4..Maintenance.charges                  -0.13  0.02  0.42 -0.09  0.01 0.22 0.78 1.3
X5..Availability.of.loan                 -0.01 -0.20  0.89  0.24  0.00 0.76 0.24 1.3
X1..Builder.reputation                    0.86 -0.18 -0.09 -0.17  0.18 0.67 0.33 1.3
X2..Appreciation.potential                0.41  0.08  0.37 -0.21  0.08 0.35 0.65 2.7
X3..Profile.of.neighbourhood              0.43  0.47 -0.21 -0.16  0.25 0.67 0.33 3.2
X4..Availability.of.domestic.help         0.06  0.83 -0.05 -0.34 -0.11 0.71 0.29 1.4
Time                                     -0.08  0.23  0.46 -0.05  0.16 0.27 0.73 1.9
Size                                      0.74  0.20  0.07  0.04  0.02 0.76 0.24 1.2
Budgets                                   0.81  0.16  0.05  0.03  0.05 0.81 0.19 1.1
Maintainances                             0.72  0.20  0.07  0.16  0.08 0.79 0.21 1.3
EMI.1                                     0.77  0.13 -0.02  0.18 -0.04 0.81 0.19 1.2

                       RC1  RC5  RC2  RC4  RC3
SS loadings           5.69 4.47 2.42 1.88 1.91
Proportion Var        0.21 0.17 0.09 0.07 0.07
Cumulative Var        0.21 0.38 0.47 0.54 0.61
Proportion Explained  0.35 0.27 0.15 0.12 0.12
Cumulative Proportion 0.35 0.62 0.77 0.88 1.00
```



```
                       RC1  RC5  RC2  RC4  RC3
SS loadings           5.69 4.47 2.42 1.88 1.91
Proportion Var        0.21 0.17 0.09 0.07 0.07
Cumulative Var        0.21 0.38 0.47 0.54 0.61
Proportion Explained  0.35 0.27 0.15 0.12 0.12
Cumulative Proportion 0.35 0.62 0.77 0.88 1.00
```

```
 With component correlations of
      RC1   RC5   RC2   RC4   RC3
RC1  1.00  0.50 -0.08  0.16  0.00
RC5  0.50  1.00  0.08  0.29 -0.06
RC2 -0.08  0.08  1.00 -0.16 -0.19
RC4  0.16  0.29 -0.16  1.00  0.09
RC3  0.00 -0.06 -0.19  0.09  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  252.24  with prob <  0.11

Fit based upon off diagonal values = 0.95>
```

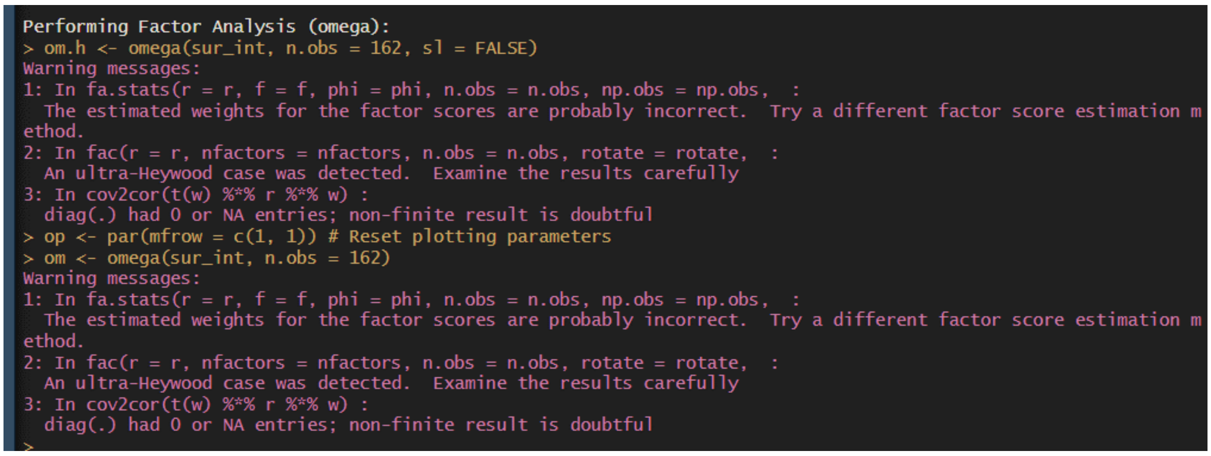
```
r

# Perform Factor Analysis using the omega function
cat("\nPerforming Factor Analysis (omega):\n")
om.h <- omega(sur_int, n.obs = 162, sl = FALSE)
op <- par(mfrow = c(1, 1)) # Reset plotting parameters
om <- omega(sur_int, n.obs = 162)
```

Explanation:

1. **Print Message:**
   - Uses `cat` to print a message.
2. **Factor Analysis:**
   - Uses `omega` from the `psych` package to perform factor analysis on `sur_int`.
   - Specifies 162 observations and `sl = FALSE` for the first analysis.
3. **Reset Plotting Parameters:**
   - Uses `par` to reset plotting parameters to default.
4. **Factor Analysis:**
   - Performs the `omega` analysis again with default parameters.

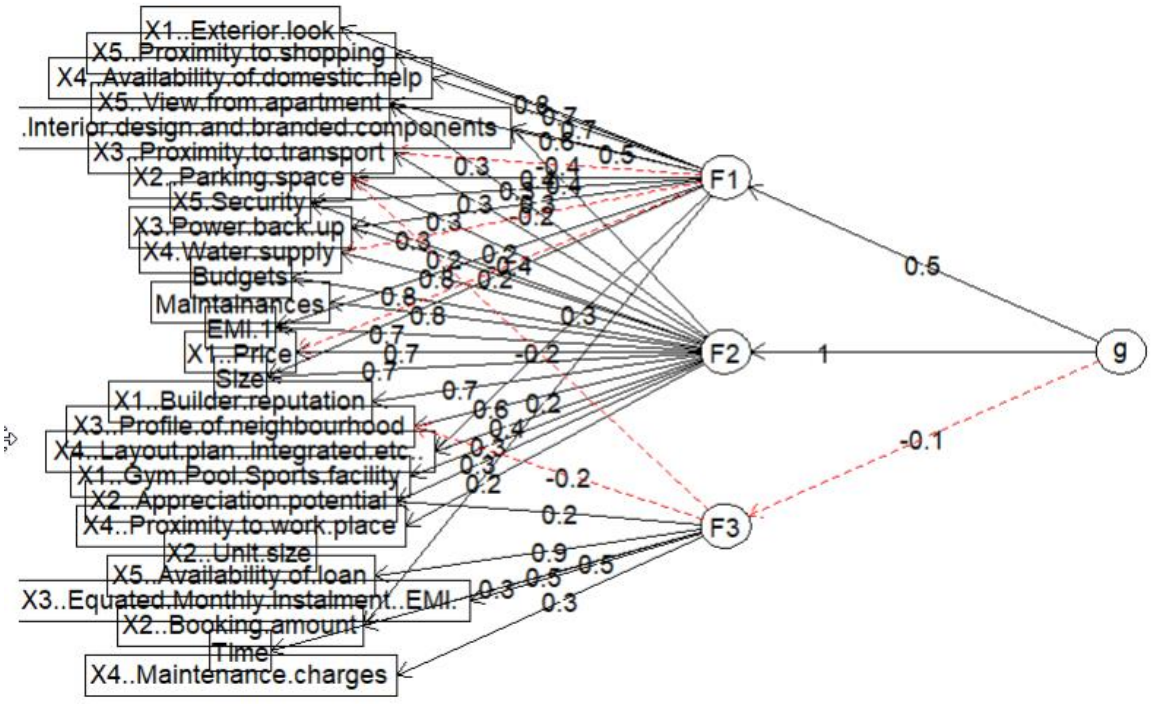
Interpretation:

Conducts factor analysis to explore the underlying structure of the data. The `omega` function provides factor loadings and reliability measures.

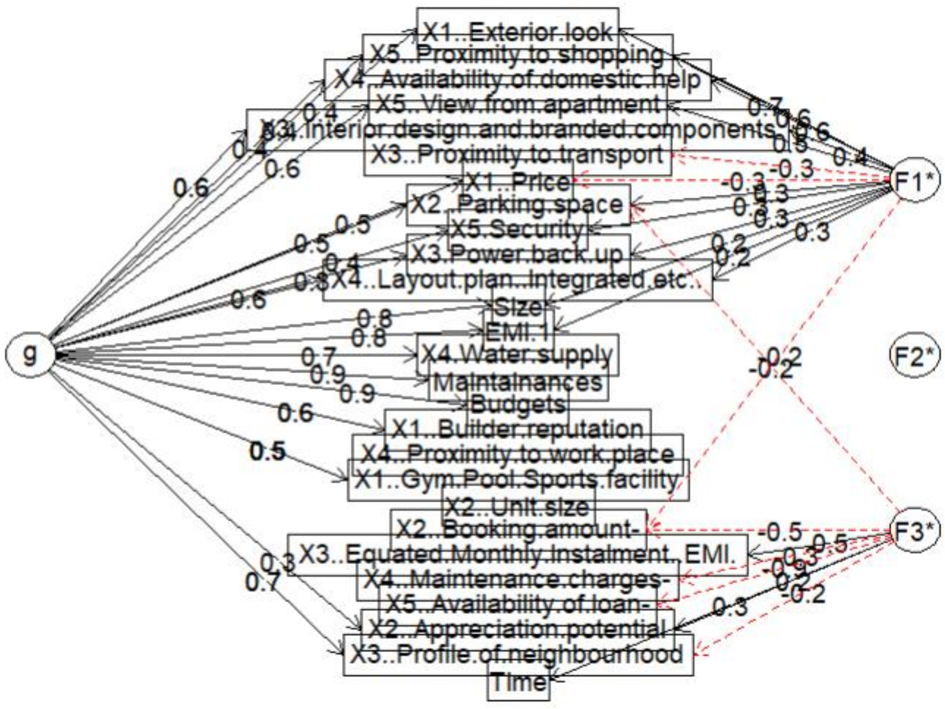


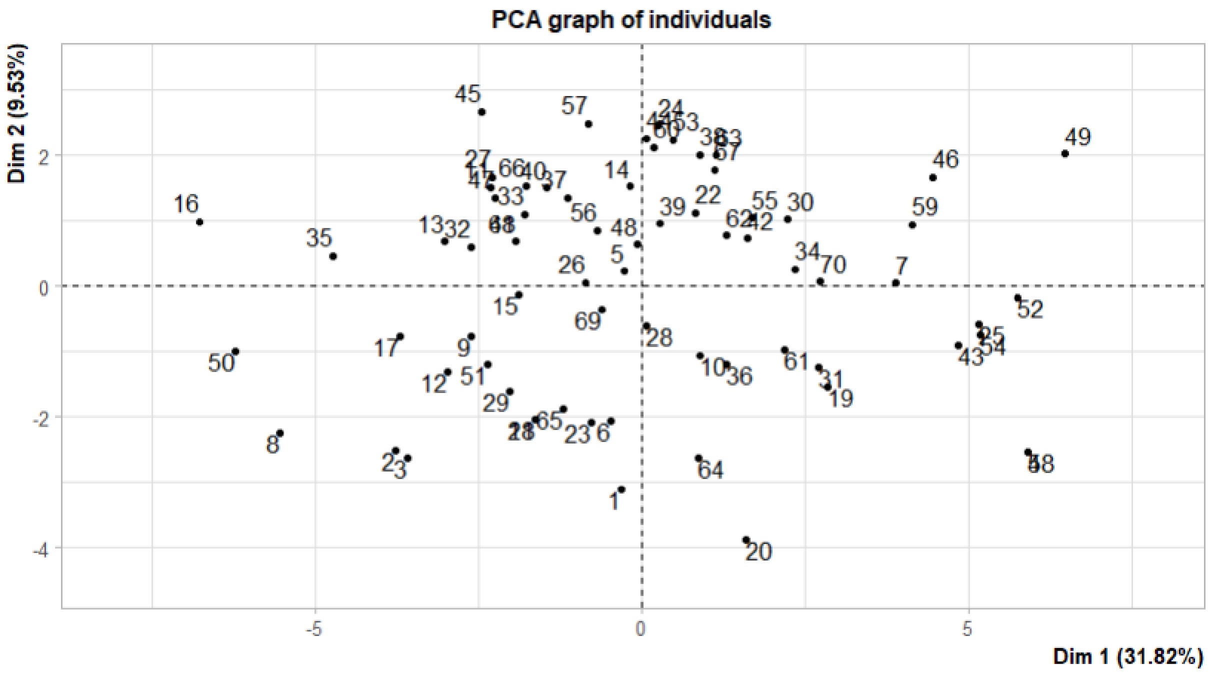
```
Performing Factor Analysis (omega):
> om.h <- omega(sur_int, n.obs = 162, sl = FALSE)
Warning messages:
1: In fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.obs,  :
  The estimated weights for the factor scores are probably incorrect.  Try a different factor score estimation m
ethod.
2: In fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate,  :
  An ultra-Heywood case was detected.  Examine the results carefully
3: In cov2cor(t(w) %*% r %*% w) :
  diag(.) had 0 or NA entries; non-finite result is doubtful
> op <- par(mfrow = c(1, 1)) # Reset plotting parameters
> om <- omega(sur_int, n.obs = 162)
Warning messages:
1: In fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.obs,  :
  The estimated weights for the factor scores are probably incorrect.  Try a different factor score estimation m
ethod.
2: In fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate,  :
  An ultra-Heywood case was detected.  Examine the results carefully
3: In cov2cor(t(w) %*% r %*% w) :
  diag(.) had 0 or NA entries; non-finite result is doubtful
>
```

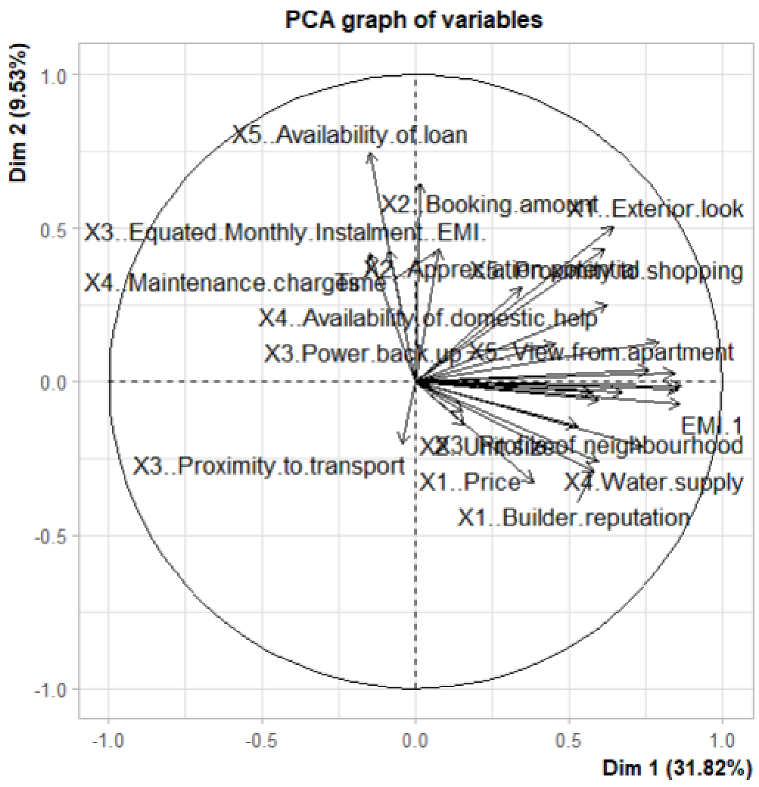
Omega


X1..Exterior.look
X5..Proximity.to.shopping
X4..Availability.of.domestic.help
X5..View.from.apartment
.Interior.design.and.branded.components
X3..Proximity.to.transport
X2..Parking.space
X5.Security
X3.Power.back.up
X4.Water.supply
Budgets
Maintainances
EMI.1
X1..Price
Size
X1..Builder.reputation
X3..Profile.of.neighbourhood
X4..Layout.plan..Integrated.etc.
X1..Gym.Pool.Sports.facility
X2..Appreciation.potential
X4..Proximity.to.work.place
X2..Unit.size
X5..Availability.of.loan
X3..Equated.Monthly.Instalment..EMI.
X2..Booking.amount
Time
X4..Maintenance.charges
F1
F2
F3
g
0.5
1
-0.1

Omega


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


PCA graph of individuals
Dim 1 (31.82%)
Dim 2 (9.53%)


PCA graph of variables
Dim 1 (31.82%)
Dim 2 (9.53%)
X5..Availability.of.loan
X2..Booking.amount
X1..Exterior.look
X3..Equated.Monthly.Instalment..EMI.
X4..Maintenance.charges
X4..Availability.of.domestic.help
X3.Power.back.up
X5..View.from.apartment
EMI.1
X3..Proximity.to.transport
X1..Price
X4.Water.supply
X1..Builder.reputation

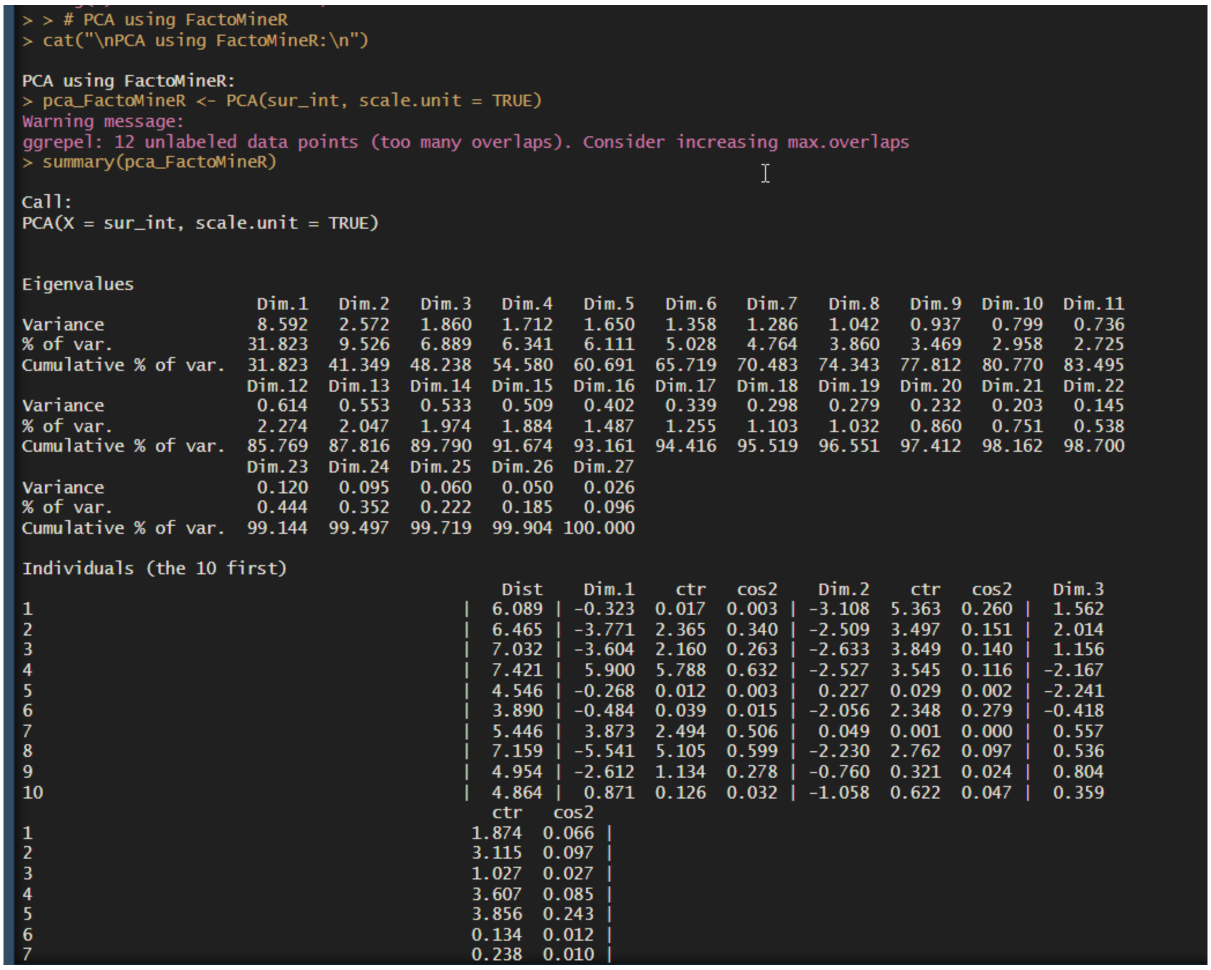
```r
r

# PCA using FactoMineR
cat("\nPCA using FactoMineR:\n")
pca_FactoMineR <- PCA(sur_int, scale.unit = TRUE)
summary(pca_FactoMineR)
biplot(pca_FactoMineR, scale = 0)
```

Explanation:

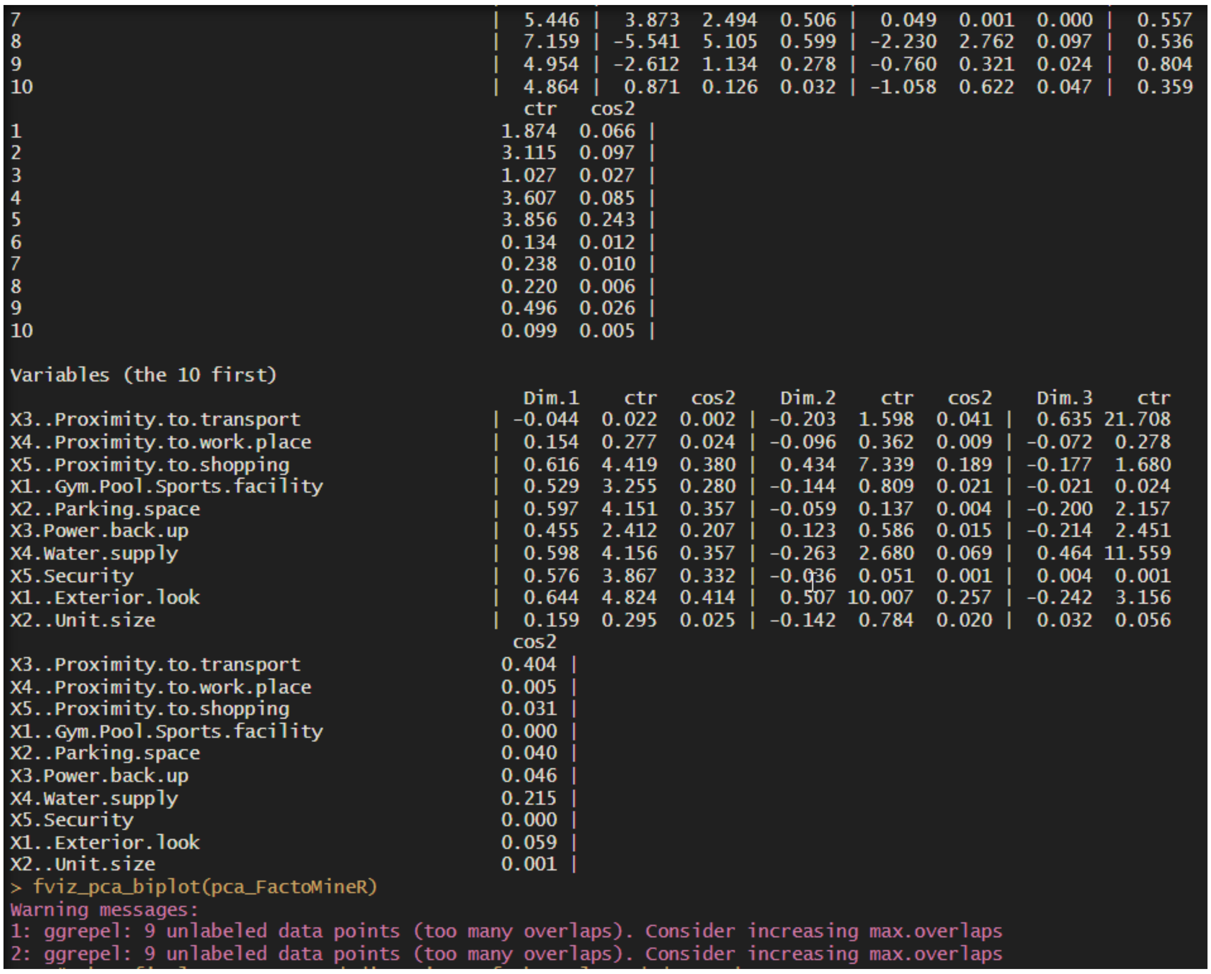
1. **Print Message:**
   - Uses `cat` to print a message.
2. **Perform PCA:**
   - Uses `PCA` from the `FactoMineR` package to perform PCA on `sur_int`.
   - Specifies `scale.unit = TRUE` to standardize variables.
3. **Print Summary:**
   - Uses `summary` to display a summary of the PCA results.
4. **Create Biplot:**
   - Uses `biplot` to visualize the PCA results, showing variables and individuals.

Interpretation:

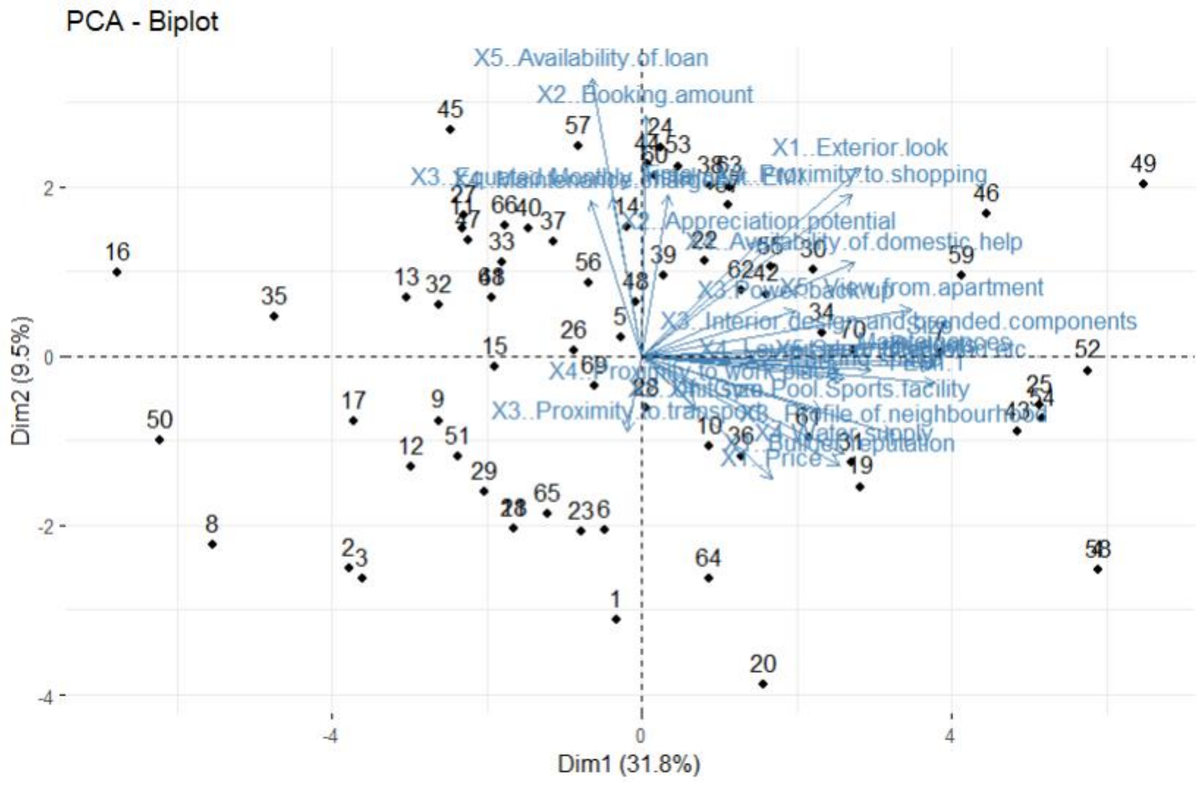
Performs PCA using a different method to validate results and provide additional visualizations. The summary and biplot offer insights into the principal components and how variables relate to each other.



```
> > # PCA using FactoMineR
> cat("\nPCA using FactoMineR:\n")

PCA using FactoMineR:
> pca_FactoMineR <- PCA(sur_int, scale.unit = TRUE)
Warning message:
ggrepel: 12 unlabeled data points (too many overlaps). Consider increasing max.overlaps
> summary(pca_FactoMineR)

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
Variance               8.592   2.572   1.860   1.712   1.650   1.358   1.286   1.042   0.937   0.799   0.736
% of var.             31.823   9.526   6.889   6.341   6.111   5.028   4.764   3.860   3.469   2.958   2.725
Cumulative % of var.  31.823  41.349  48.238  54.580  60.691  65.719  70.483  74.343  77.812  80.770  83.495
                      Dim.12  Dim.13  Dim.14  Dim.15  Dim.16  Dim.17  Dim.18  Dim.19  Dim.20  Dim.21  Dim.22
Variance               0.614   0.553   0.533   0.509   0.402   0.339   0.298   0.279   0.232   0.203   0.145
% of var.              2.274   2.047   1.974   1.884   1.487   1.255   1.103   1.032   0.860   0.751   0.538
Cumulative % of var.  85.769  87.816  89.790  91.674  93.161  94.416  95.519  96.551  97.412  98.162  98.700
                      Dim.23  Dim.24  Dim.25  Dim.26  Dim.27
Variance               0.120   0.095   0.060   0.050   0.026
% of var.              0.444   0.352   0.222   0.185   0.096
Cumulative % of var.  99.144  99.497  99.719  99.904 100.000

Individuals (the 10 first)
                                        Dist    Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3
1                                   |  6.089 | -0.323  0.017  0.003 | -3.108  5.363  0.260 |  1.562
2                                   |  6.465 | -3.771  2.365  0.340 | -2.509  3.497  0.151 |  2.014
3                                   |  7.032 | -3.604  2.160  0.263 | -2.633  3.849  0.140 |  1.156
4                                   |  7.421 |  5.900  5.788  0.632 | -2.527  3.545  0.116 | -2.167
5                                   |  4.546 | -0.268  0.012  0.003 |  0.227  0.029  0.002 | -2.241
6                                   |  3.890 | -0.484  0.039  0.015 | -2.056  2.348  0.279 | -0.418
7                                   |  5.446 |  3.873  2.494  0.506 |  0.049  0.001  0.000 |  0.557
8                                   |  7.159 | -5.541  5.105  0.599 | -2.230  2.762  0.097 |  0.536
9                                   |  4.954 | -2.612  1.134  0.278 | -0.760  0.321  0.024 |  0.804
10                                  |  4.864 |  0.871  0.126  0.032 | -1.058  0.622  0.047 |  0.359
                                        ctr   cos2
1                                     1.874  0.066 |
2                                     3.115  0.097 |
3                                     1.027  0.027 |
4                                     3.607  0.085 |
5                                     3.856  0.243 |
6                                     0.134  0.012 |
7                                     0.238  0.010 |
```



```
7                                        |  5.446 |  3.873  2.494  0.506 |  0.049  0.001  0.000 |  0.557
8                                        |  7.159 | -5.541  5.105  0.599 | -2.230  2.762  0.097 |  0.536
9                                        |  4.954 | -2.612  1.134  0.278 | -0.760  0.321  0.024 |  0.804
10                                       |  4.864 |  0.871  0.126  0.032 | -1.058  0.622  0.047 |  0.359
                                            ctr   cos2
1                                         1.874  0.066 |
2                                         3.115  0.097 |
3                                         1.027  0.027 |
4                                         3.607  0.085 |
5                                         3.856  0.243 |
6                                         0.134  0.012 |
7                                         0.238  0.010 |
8                                         0.220  0.006 |
9                                         0.496  0.026 |
10                                        0.099  0.005 |

Variables (the 10 first)
                                            Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr
X3..Proximity.to.transport               | -0.044  0.022  0.002 | -0.203  1.598  0.041 |  0.635 21.708
X4..Proximity.to.work.place              |  0.154  0.277  0.024 | -0.096  0.362  0.009 | -0.072  0.278
X5..Proximity.to.shopping                |  0.616  4.419  0.380 |  0.434  7.339  0.189 | -0.177  1.680
X1..Gym.Pool.Sports.facility             |  0.529  3.255  0.280 | -0.144  0.809  0.021 | -0.021  0.024
X2..Parking.space                        |  0.597  4.151  0.357 | -0.059  0.137  0.004 | -0.200  2.157
X3.Power.back.up                         |  0.455  2.412  0.207 |  0.123  0.586  0.015 | -0.214  2.451
X4.Water.supply                          |  0.598  4.156  0.357 | -0.263  2.680  0.069 |  0.464 11.559
X5.Security                              |  0.576  3.867  0.332 | -0.036  0.051  0.001 |  0.004  0.001
X1..Exterior.look                        |  0.644  4.824  0.414 |  0.507 10.007  0.257 | -0.242  3.156
X2..Unit.size                            |  0.159  0.295  0.025 | -0.142  0.784  0.020 |  0.032  0.056
                                           cos2
X3..Proximity.to.transport                0.404 |
X4..Proximity.to.work.place               0.005 |
X5..Proximity.to.shopping                 0.031 |
X1..Gym.Pool.Sports.facility              0.000 |
X2..Parking.space                         0.040 |
X3.Power.back.up                          0.046 |
X4.Water.supply                           0.215 |
X5.Security                               0.000 |
X1..Exterior.look                         0.059 |
X2..Unit.size                             0.001 |
> fviz_pca_biplot(pca_FactoMineR)
Warning messages:
1: ggrepel: 9 unlabeled data points (too many overlaps). Consider increasing max.overlaps
2: ggrepel: 9 unlabeled data points (too many overlaps). Consider increasing max.overlaps
```



```
r

# Show final structure and dimensions of the selected data subset
cat("\nFinal structure of the selected data subset:\n")
str(sur_int)
cat("\nFinal dimensions of the selected data subset:\n")
print(dim(sur_int))
cat("\nShowing the selected data subset:\n")
print(head(sur_int))
```

Explanation:

1. **Print Final Structure:**
   - Uses `cat` to print a message.
   - Uses `str` to display the structure of `sur_int`.
2. **Print Final Dimensions:**
   - Uses `cat` to print a message.
   - Uses `dim` to display the dimensions of `sur_int`.
3. **Print Final Data Subset:**
   - Uses `cat` to print a message.
   - Uses `head` to display the first few rows of `sur_int`.

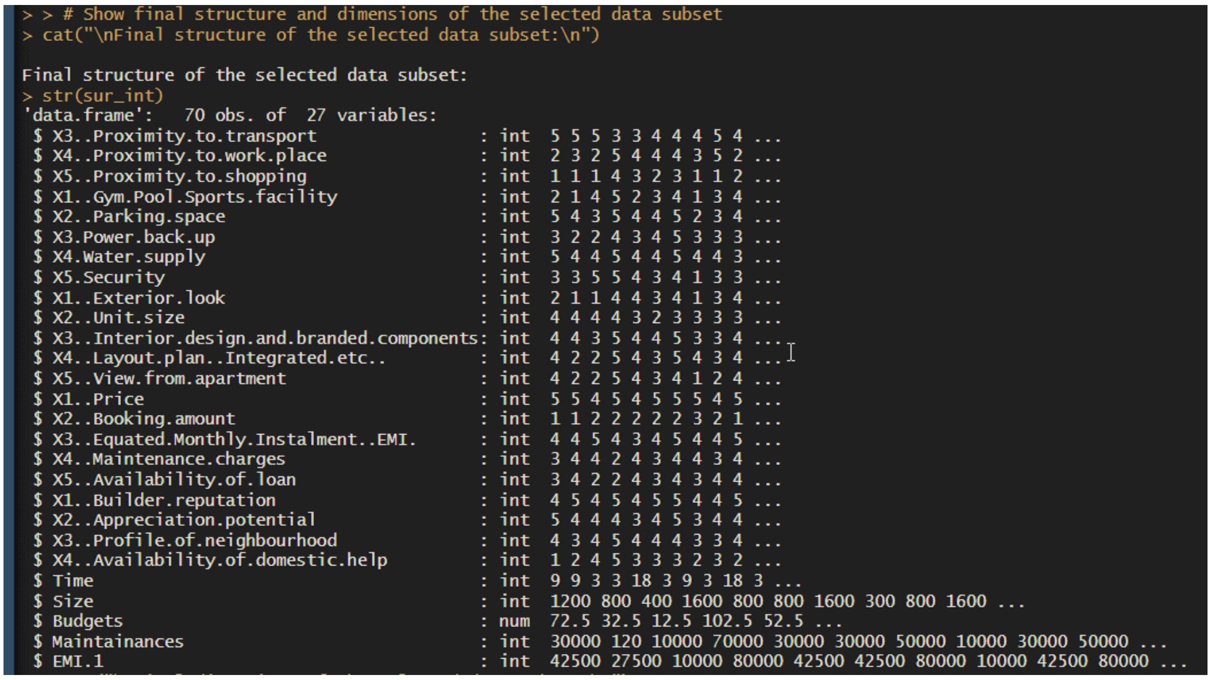
Interpretation:

Confirms the consistency of the data subset used for analysis by showing its final structure and dimensions, along with a preview of the data.

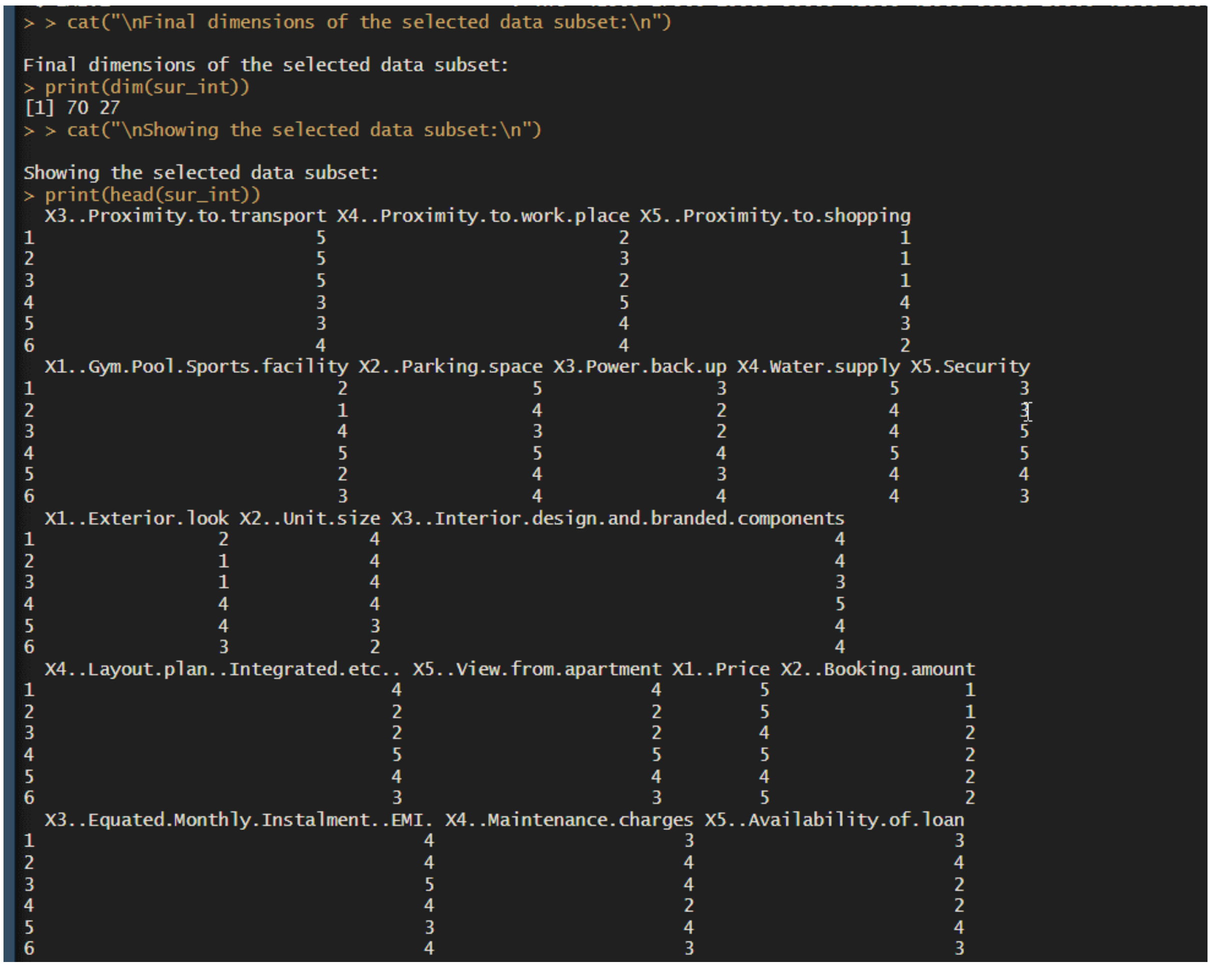
Output Interpretation

- **Dimensions of the dataset:** Indicates the number of rows and columns, providing a sense of dataset size.
- **Column names in the dataset:** Lists all variables, helping to identify which ones are available for analysis.
- **First few rows of the dataset:** Gives a preview of the data, offering an initial look at the values and structure.
- **Structure of the dataset:** Shows data types and a summary of each column, important for understanding how data is organized.
- **Checking for missing values:** Reveals the total number of missing values, which might require data cleaning steps.
- **Structure and dimensions of the selected data subset:** Confirms that the correct columns have been selected and their structure.
- **Performing PCA and Factor Analysis:** Provides insights into the underlying structure of the data, identifying key components and factors.
- **PCA using FactoMineR:** Offers a detailed summary and visualization of PCA results, validating and complementing earlier analysis.
- **Final structure and dimensions:** Confirms that the data subset remains consistent throughout the analysis, ensuring accuracy in results.

This detailed breakdown ensures a comprehensive understanding of each step, its purpose, and the results obtained, making the analysis clear and interpretable.



```
> > # Show final structure and dimensions of the selected data subset
> cat("\nFinal structure of the selected data subset:\n")

Final structure of the selected data subset:
> 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 ...
```



```
> > cat("\nFinal dimensions of the selected data subset:\n")

Final dimensions of the selected data subset:
> print(dim(sur_int))
[1] 70 27
> > cat("\nShowing the selected data subset:\n")

Showing the selected data subset:
> print(head(sur_int))
  X3..Proximity.to.transport X4..Proximity.to.work.place X5..Proximity.to.shopping
1                          5                           2                         1
2                          5                           3                         1
3                          5                           2                         1
4                          3                           5                         4
5                          3                           4                         3
6                          4                           4                         2
  X1..Gym.Pool.Sports.facility X2..Parking.space X3.Power.back.up X4.Water.supply X5.Security
1                            2                 5                3               5           3
2                            1                 4                2               4           3
3                            4                 3                2               4           5
4                            5                 5                4               5           5
5                            2                 4                3               4           4
6                            3                 4                4               4           3
  X1..Exterior.look X2..Unit.size X3..Interior.design.and.branded.components
1                 2             4                                          4
2                 1             4                                          4
3                 1             4                                          3
4                 4             4                                          5
5                 4             3                                          4
6                 3             2                                          4
  X4..Layout.plan..Integrated.etc.. X5..View.from.apartment X1..Price X2..Booking.amount
1                                 4                       4         5                  1
2                                 2                       2         5                  1
3                                 2                       2         4                  2
4                                 5                       5         5                  2
5                                 4                       4         4                  2
6                                 3                       3         5                  2
  X3..Equated.Monthly.Instalment..EMI. X4..Maintenance.charges X5..Availability.of.loan
1                                    4                       3                        3
2                                    4                       4                        4
3                                    5                       4                        2
4                                    4                       2                        2
5                                    3                       4                        4
6                                    4                       3                        3
```

r
Copy code

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

1. **Factor Analysis Initialization**:
   - `fa()` function is used to perform Factor Analysis on the dataset `sur_int`.
   - `nfactors = 4` specifies that we want to extract 4 factors from the dataset.
   - `rotate = "varimax"` specifies the rotation method. Varimax rotation is commonly used to simplify the interpretation of factor loadings by maximizing the variance of squared loadings within each factor.
2. **Factor Analysis Object**:
   - The result of `fa()` is stored in `factor_analysis`, which is an object containing various components of the Factor Analysis results.

```
r
Copy code
# Names of the factor analysis components
names(factor_analysis)
```

3. **Component Names**:
   - `names(factor_analysis)` prints the names of components stored in the `factor_analysis` object. These typically include information like loadings, communalities, and scores.

```
r
Copy code
# Print factor loadings
print(factor_analysis$loadings, reorder = TRUE)
```

4. **Factor Loadings**:
   - `factor_analysis$loadings` accesses and prints the factor loadings. Factor loadings represent the correlations between the variables and the factors extracted from the data.
   - `reorder = TRUE` reorders the variables based on the loadings, making it easier to interpret the results.

```
r
Copy code
# Plot the factor diagram
fa.diagram(factor_analysis)
```

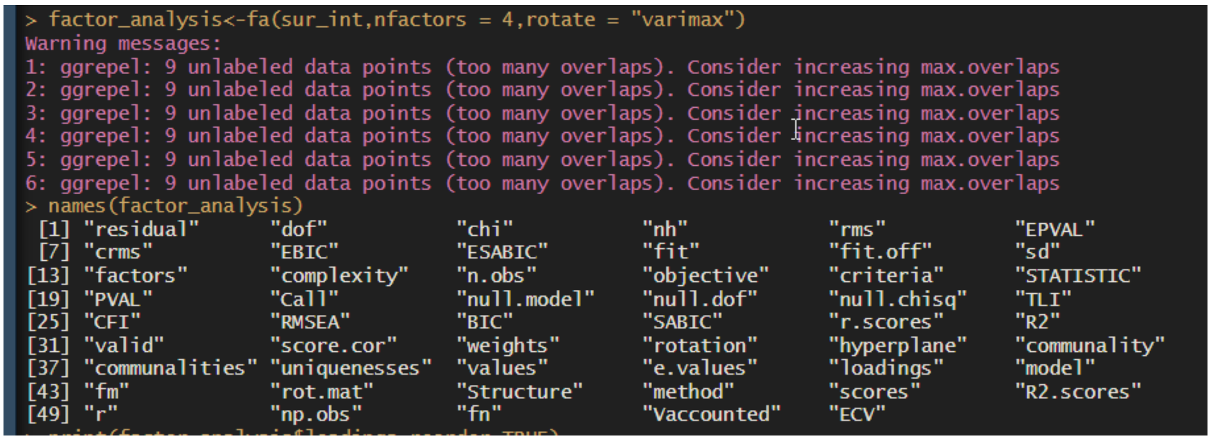
5. **Factor Analysis Diagram**:
   - `fa.diagram(factor_analysis)` generates a diagram illustrating the relationships between variables and factors. This diagram helps visualize how variables load onto each factor and the relationships between factors.

```
r
Copy code
# Print communalities
print(factor_analysis$communality)
```

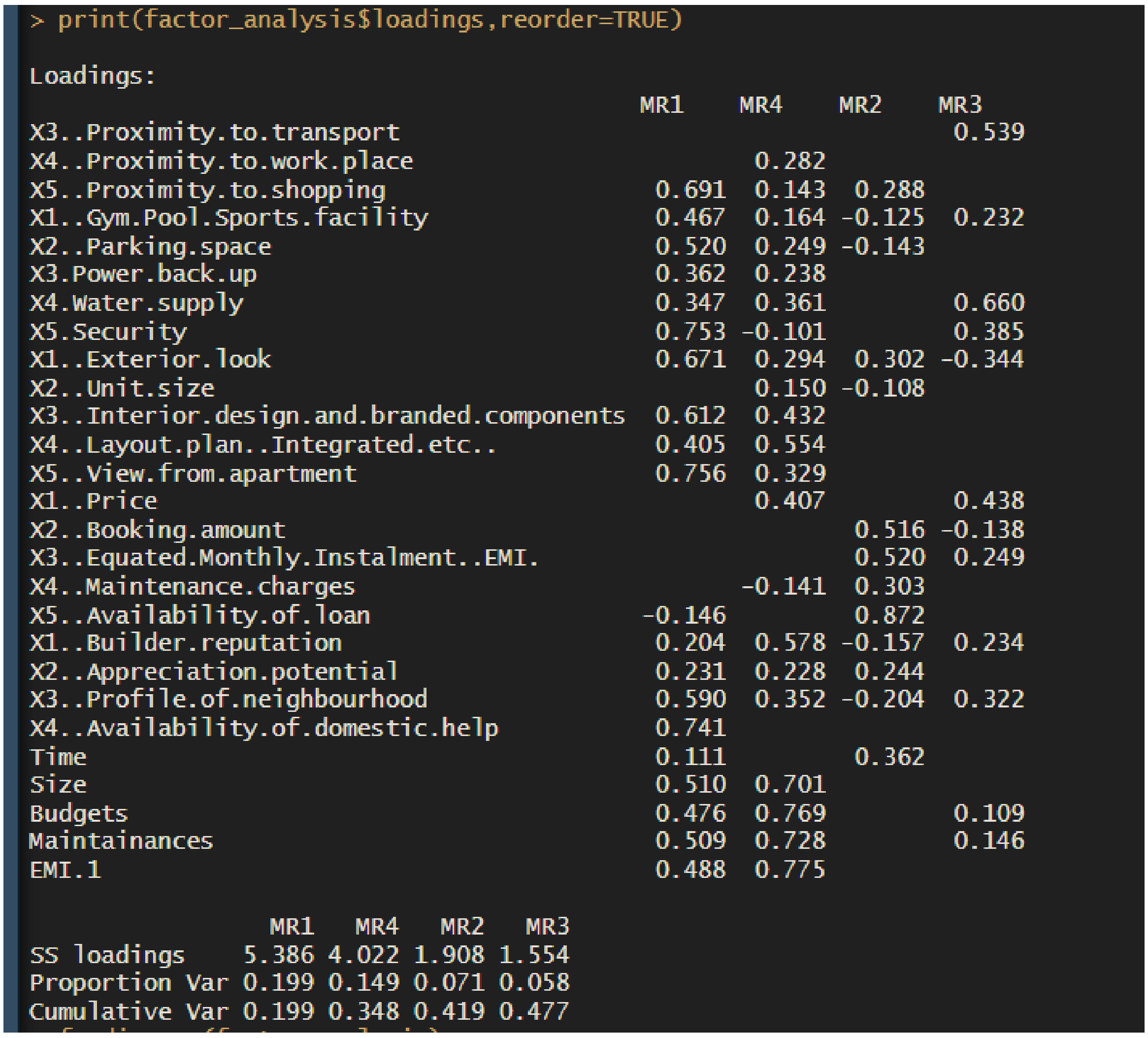
6. **Communalities**:
   - `factor_analysis$communality` prints the communalities, which indicate the proportion of each variable's variance explained by all the factors extracted. Higher communalities suggest that the variable is well-represented by the factors.

```
r
Copy code
# Print factor scores
print(factor_analysis$scores)
```

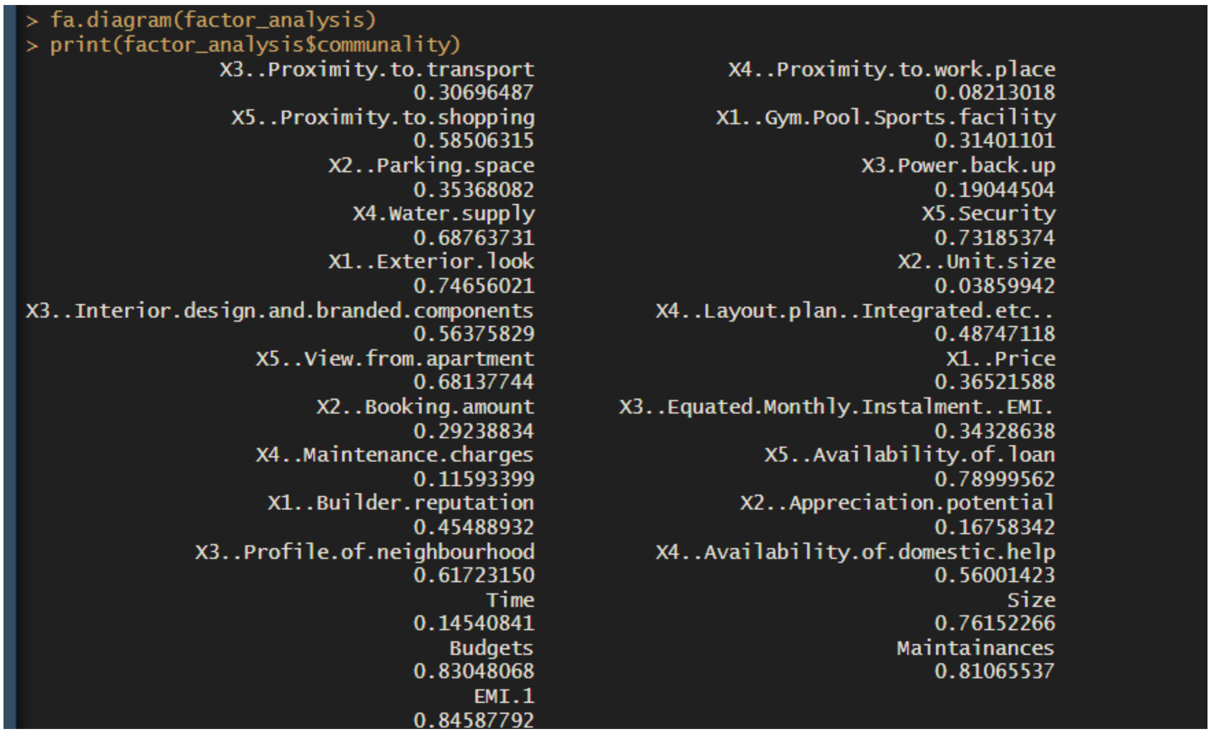
7. **Factor Scores**:
   - `factor_analysis$scores` prints the factor scores, which represent the scores of each observation (or case) on the extracted factors. Factor scores allow us to understand how each observation relates to the identified factors.



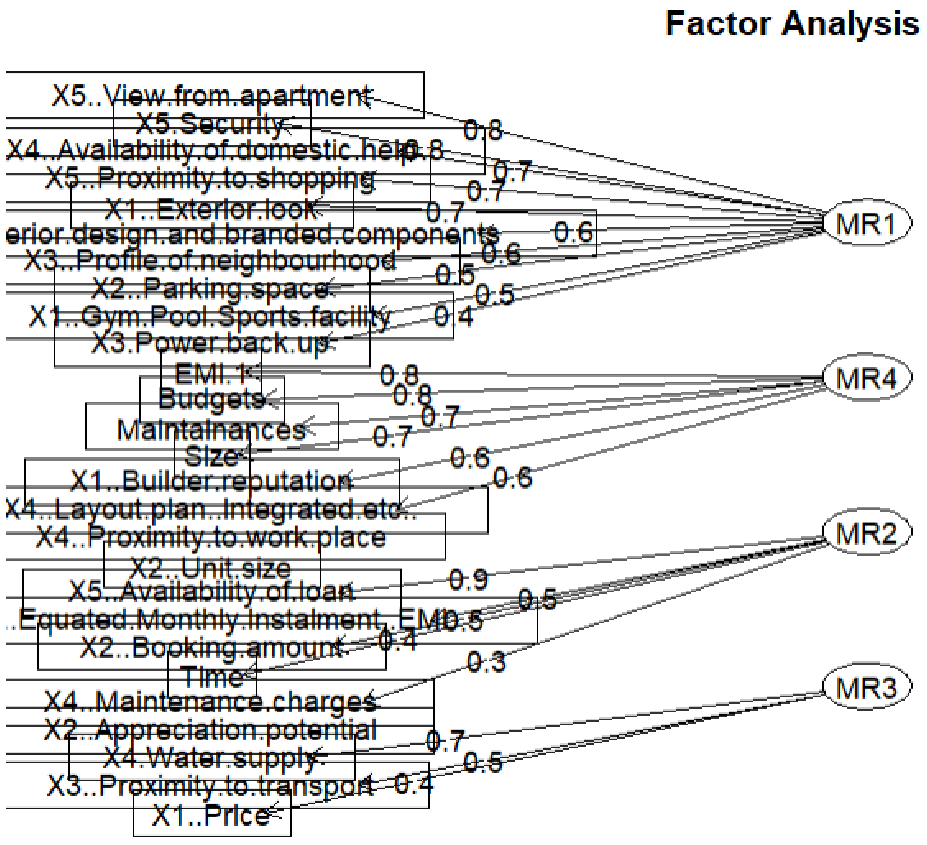
```
> factor_analysis<-fa(sur_int,nfactors = 4,rotate = "varimax")
Warning messages:
1: ggrepel: 9 unlabeled data points (too many overlaps). Consider increasing max.overlaps
2: ggrepel: 9 unlabeled data points (too many overlaps). Consider increasing max.overlaps
3: ggrepel: 9 unlabeled data points (too many overlaps). Consider increasing max.overlaps
4: ggrepel: 9 unlabeled data points (too many overlaps). Consider increasing max.overlaps
5: ggrepel: 9 unlabeled data points (too many overlaps). Consider increasing max.overlaps
6: ggrepel: 9 unlabeled data points (too many overlaps). Consider increasing max.overlaps
> names(factor_analysis)
 [1] "residual"      "dof"           "chi"           "nh"            "rms"           "EPVAL"
 [7] "crms"          "EBIC"          "ESABIC"        "fit"           "fit.off"       "sd"
[13] "factors"       "complexity"    "n.obs"         "objective"     "criteria"      "STATISTIC"
[19] "PVAL"          "Call"          "null.model"    "null.dof"      "null.chisq"    "TLI"
[25] "CFI"           "RMSEA"         "BIC"           "SABIC"         "r.scores"      "R2"
[31] "valid"         "score.cor"     "weights"       "rotation"      "hyperplane"    "communality"
[37] "communalities" "uniquenesses"  "values"        "e.values"      "loadings"      "model"
[43] "fm"            "rot.mat"       "Structure"     "method"        "scores"        "R2.scores"
[49] "r"             "np.obs"        "fn"            "Vaccounted"    "ECV"
```

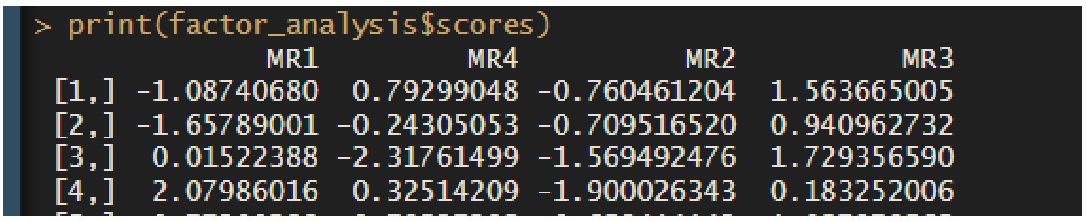


```
> print(factor_analysis$loadings,reorder=TRUE)

Loadings:
                                           MR1    MR4    MR2    MR3
X3..Proximity.to.transport                                       0.539
X4..Proximity.to.work.place                        0.282
X5..Proximity.to.shopping                   0.691  0.143  0.288
X1..Gym.Pool.Sports.facility                0.467  0.164 -0.125  0.232
X2..Parking.space                           0.520  0.249 -0.143
X3.Power.back.up                            0.362  0.238
X4.Water.supply                             0.347  0.361         0.660
X5.Security                                 0.753 -0.101         0.385
X1..Exterior.look                           0.671  0.294  0.302 -0.344
X2..Unit.size                                      0.150 -0.108
X3..Interior.design.and.branded.components  0.612  0.432
X4..Layout.plan..Integrated.etc..           0.405  0.554
X5..View.from.apartment                     0.756  0.329
X1..Price                                          0.407         0.438
X2..Booking.amount                                        0.516 -0.138
X3..Equated.Monthly.Instalment..EMI.                      0.520  0.249
X4..Maintenance.charges                           -0.141  0.303
X5..Availability.of.loan                   -0.146         0.872
X1..Builder.reputation                      0.204  0.578 -0.157  0.234
X2..Appreciation.potential                  0.231  0.228  0.244
X3..Profile.of.neighbourhood                0.590  0.352 -0.204  0.322
X4..Availability.of.domestic.help           0.741
Time                                        0.111         0.362
Size                                        0.510  0.701
Budgets                                     0.476  0.769         0.109
Maintainances                               0.509  0.728         0.146
EMI.1                                       0.488  0.775

                 MR1   MR4   MR2   MR3
SS loadings    5.386 4.022 1.908 1.554
Proportion Var 0.199 0.149 0.071 0.058
Cumulative Var 0.199 0.348 0.419 0.477
```



```
> fa.diagram(factor_analysis)
> print(factor_analysis$communality)
                    X3..Proximity.to.transport                      X4..Proximity.to.work.place
                                    0.30696487                                       0.08213018
                     X5..Proximity.to.shopping                    X1..Gym.Pool.Sports.facility
                                    0.58506315                                       0.31401101
                             X2..Parking.space                                 X3.Power.back.up
                                    0.35368082                                       0.19044504
                               X4.Water.supply                                      X5.Security
                                    0.68763731                                       0.73185374
                             X1..Exterior.look                                     X2..Unit.size
                                    0.74656021                                       0.03859942
X3..Interior.design.and.branded.components                   X4..Layout.plan..Integrated.etc..
                                    0.56375829                                       0.48747118
                      X5..View.from.apartment                                         X1..Price
                                    0.68137744                                       0.36521588
                            X2..Booking.amount          X3..Equated.Monthly.Instalment..EMI.
                                    0.29238834                                       0.34328638
                      X4..Maintenance.charges                         X5..Availability.of.loan
                                    0.11593399                                       0.78999562
                         X1..Builder.reputation                    X2..Appreciation.potential
                                    0.45488932                                       0.16758342
                 X3..Profile.of.neighbourhood                X4..Availability.of.domestic.help
                                    0.61723150                                       0.56001423
                                          Time                                             Size
                                    0.14540841                                       0.76152266
                                       Budgets                                     Maintainances
                                    0.83048068                                       0.81065537
                                         EMI.1
                                    0.84587792
```





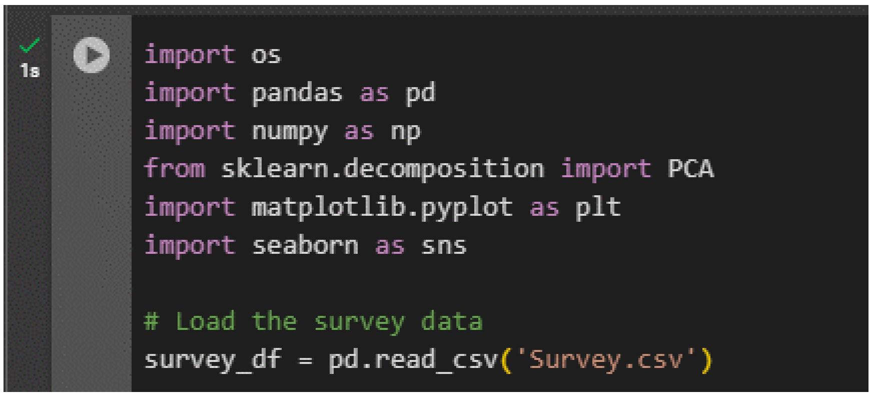
```
> print(factor_analysis$scores)
             MR1          MR4          MR2          MR3
 [1,] -1.08740680  0.79299048 -0.760461204  1.563665005
 [2,] -1.65789001 -0.24305053 -0.709516520  0.940962732
 [3,]  0.01522388 -2.31761499 -1.569492476  1.729356590
 [4,]  2.07986016  0.32514209 -1.900026343  0.183252006
```

Python Language

Step 1: Importing Libraries and Loading the Dataset

```
import os
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns

# Load the survey data
survey_df = pd.read_csv('Survey.csv')
```

- **Explanation**:
    - **Imports**: Necessary libraries are imported (`os`, `pandas`, `numpy`, `PCA` from `sklearn`, `matplotlib.pyplot`, `seaborn`).
    - **Data Loading**: Loads a CSV file named 'Survey.csv' into a Pandas DataFrame (`survey_df`).

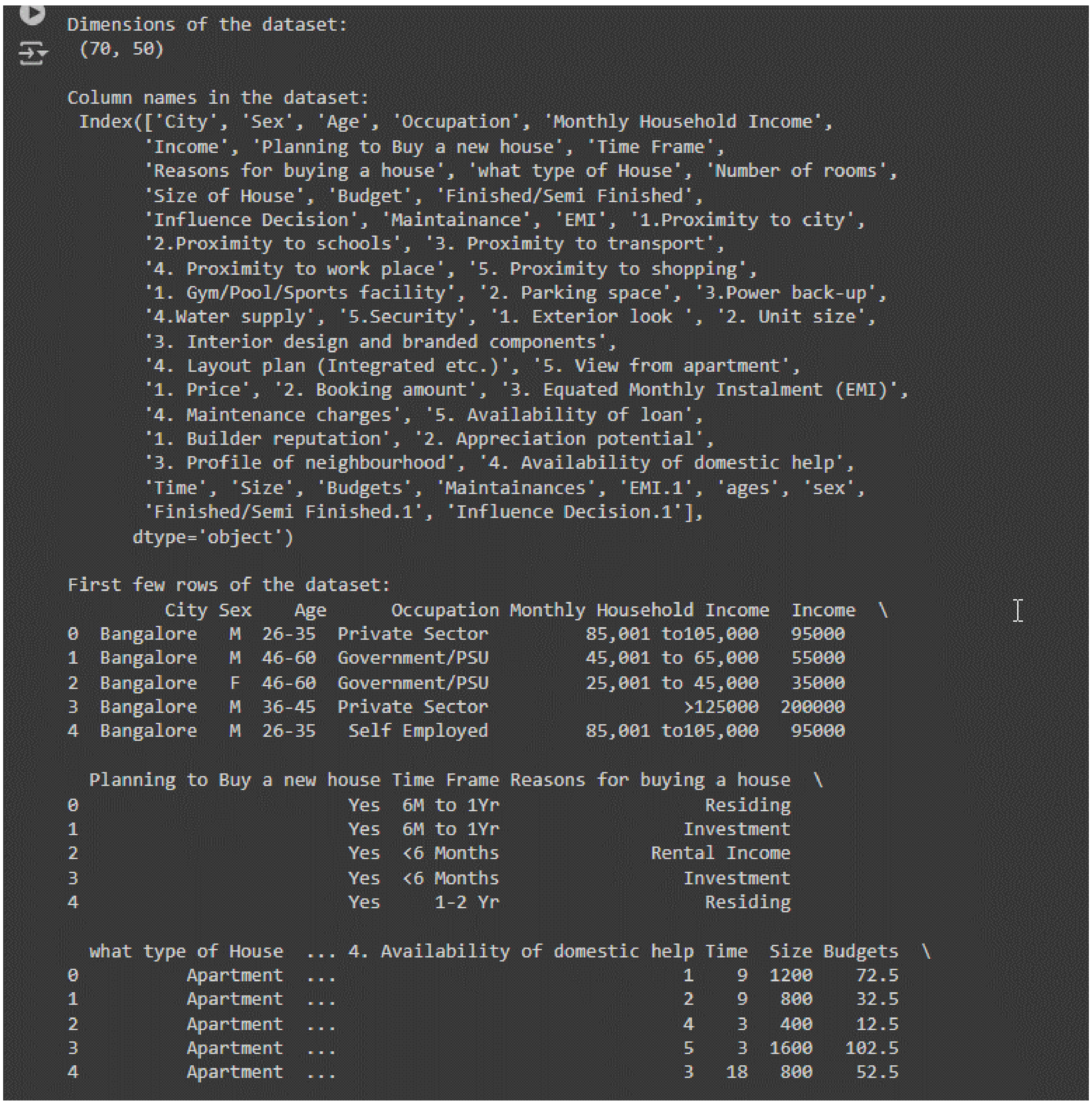


```
import os
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns

# Load the survey data
survey_df = pd.read_csv('Survey.csv')
```

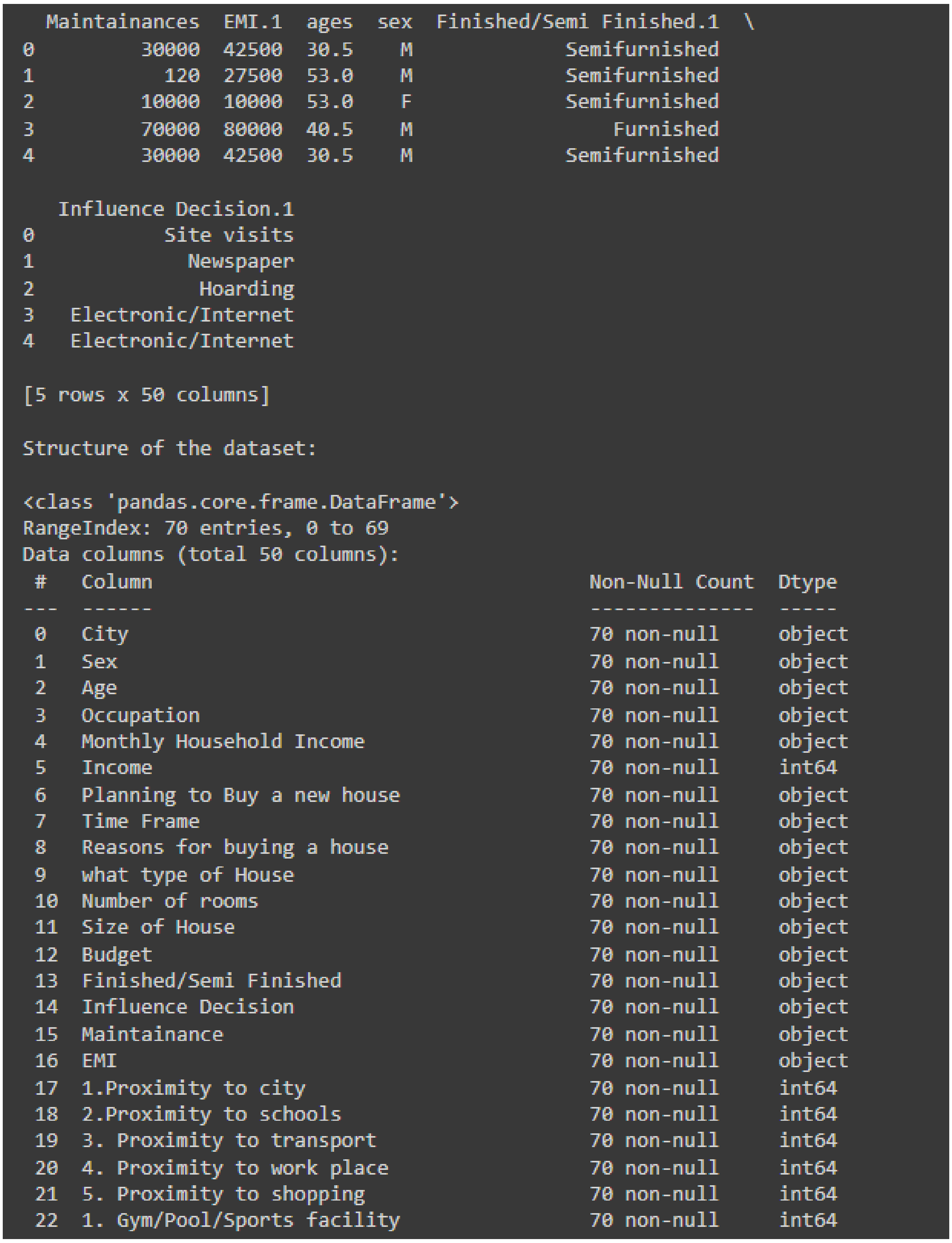
Step 2: Understanding the Dataset

```
# Display dimensions, column names, and structure of the dataset
print("Dimensions of the dataset:\n", survey_df.shape)

print("\nColumn names in the dataset:\n", survey_df.columns)

print("\nFirst few rows of the dataset:\n", survey_df.head())

print("\nStructure of the dataset:\n")
print(survey_df.info())
```

- **Explanation**:
    - **Dimensions**: Prints the number of rows and columns in the dataset (`shape`).
    - **Column Names**: Lists all column names in the dataset (`columns`).
    - **First Few Rows**: Shows the first few rows of the dataset (`head()`).
    - **Structure**: Displays the structure of the dataset, including column data types and memory usage (`info()`).



```
Dimensions of the dataset:
 (70, 50)

Column names in the dataset:
 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'],
      dtype='object')

First few rows of the dataset:
        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
```

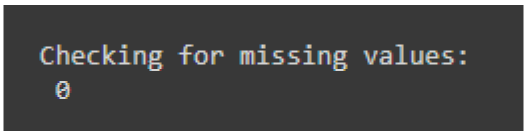


```
  Maintainances  EMI.1  ages  sex  Finished/Semi Finished.1  \
0         30000  42500  30.5    M             Semifurnished
1           120  27500  53.0    M             Semifurnished
2         10000  10000  53.0    F             Semifurnished
3         70000  80000  40.5    M                 Furnished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

Structure of the dataset:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70 entries, 0 to 69
Data columns (total 50 columns):
 #   Column                                       Non-Null Count  Dtype
---  ------                                       --------------  -----
 0   City                                         70 non-null     object
 1   Sex                                          70 non-null     object
 2   Age                                          70 non-null     object
 3   Occupation                                   70 non-null     object
 4   Monthly Household Income                     70 non-null     object
 5   Income                                       70 non-null     int64
 6   Planning to Buy a new house                  70 non-null     object
 7   Time Frame                                   70 non-null     object
 8   Reasons for buying a house                   70 non-null     object
 9   what type of House                           70 non-null     object
 10  Number of rooms                              70 non-null     object
 11  Size of House                                70 non-null     object
 12  Budget                                       70 non-null     object
 13  Finished/Semi Finished                       70 non-null     object
 14  Influence Decision                           70 non-null     object
 15  Maintainance                                 70 non-null     object
 16  EMI                                          70 non-null     object
 17  1.Proximity to city                          70 non-null     int64
 18  2.Proximity to schools                       70 non-null     int64
 19  3. Proximity to transport                    70 non-null     int64
 20  4. Proximity to work place                   70 non-null     int64
 21  5. Proximity to shopping                     70 non-null     int64
 22  1. Gym/Pool/Sports facility                  70 non-null     int64
```

Step 3: Checking for Missing Values

```
# Check for missing values
print("\nChecking for missing values:\n", survey_df.isnull().sum().sum())
```

- **Explanation**:
    - **Missing Values**: Counts and prints the total number of missing values in the dataset (`isnull().sum().sum()`).

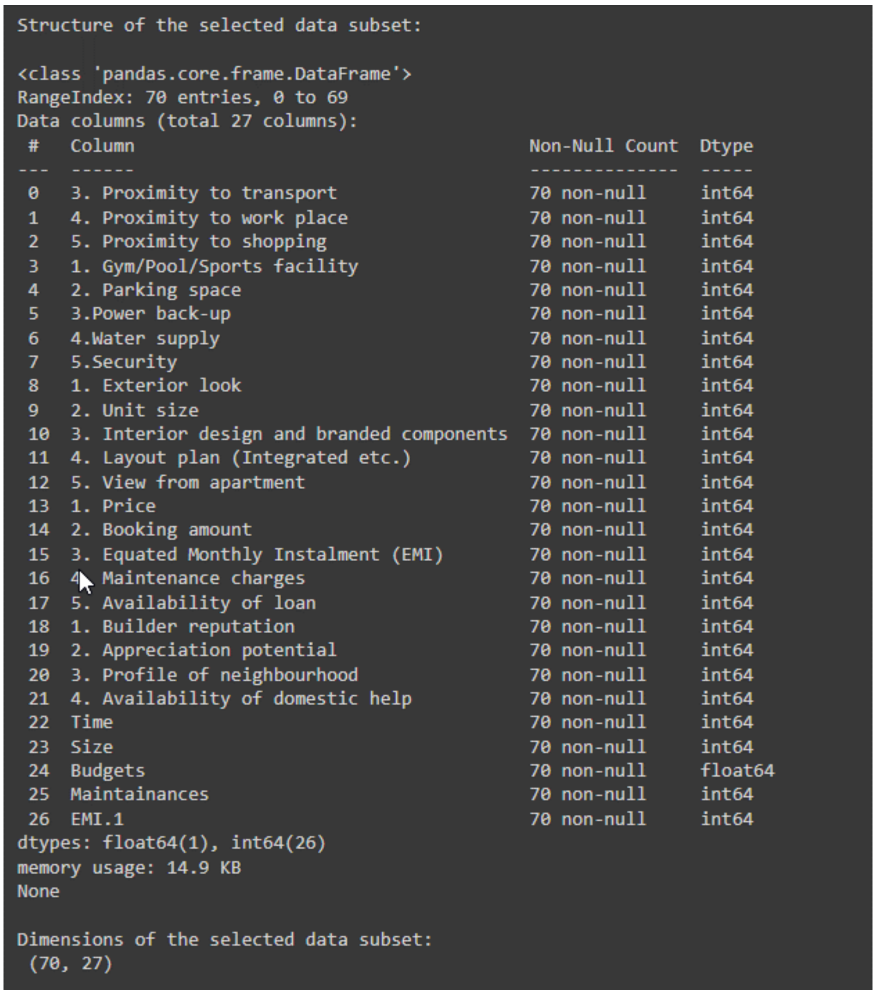


```
Checking for missing values:
 0
```

Step 4: Selecting Data for PCA

```
# Select the relevant columns for PCA and Factor Analysis
sur_int = survey_df.iloc[:, 19:46]

print("\nStructure of the selected data subset:\n")
print(sur_int.info())

print("\nDimensions of the selected data subset:\n", sur_int.shape)
```

- **Explanation**:
    - **Subset Selection**: Extracts a subset of columns from index 19 to 45 for PCA and further analysis (`iloc[:, 19:46]`).
    - **Structure and Dimensions**: Prints the structure (data types) and dimensions (rows and columns) of the selected subset (`info()` and `shape`).

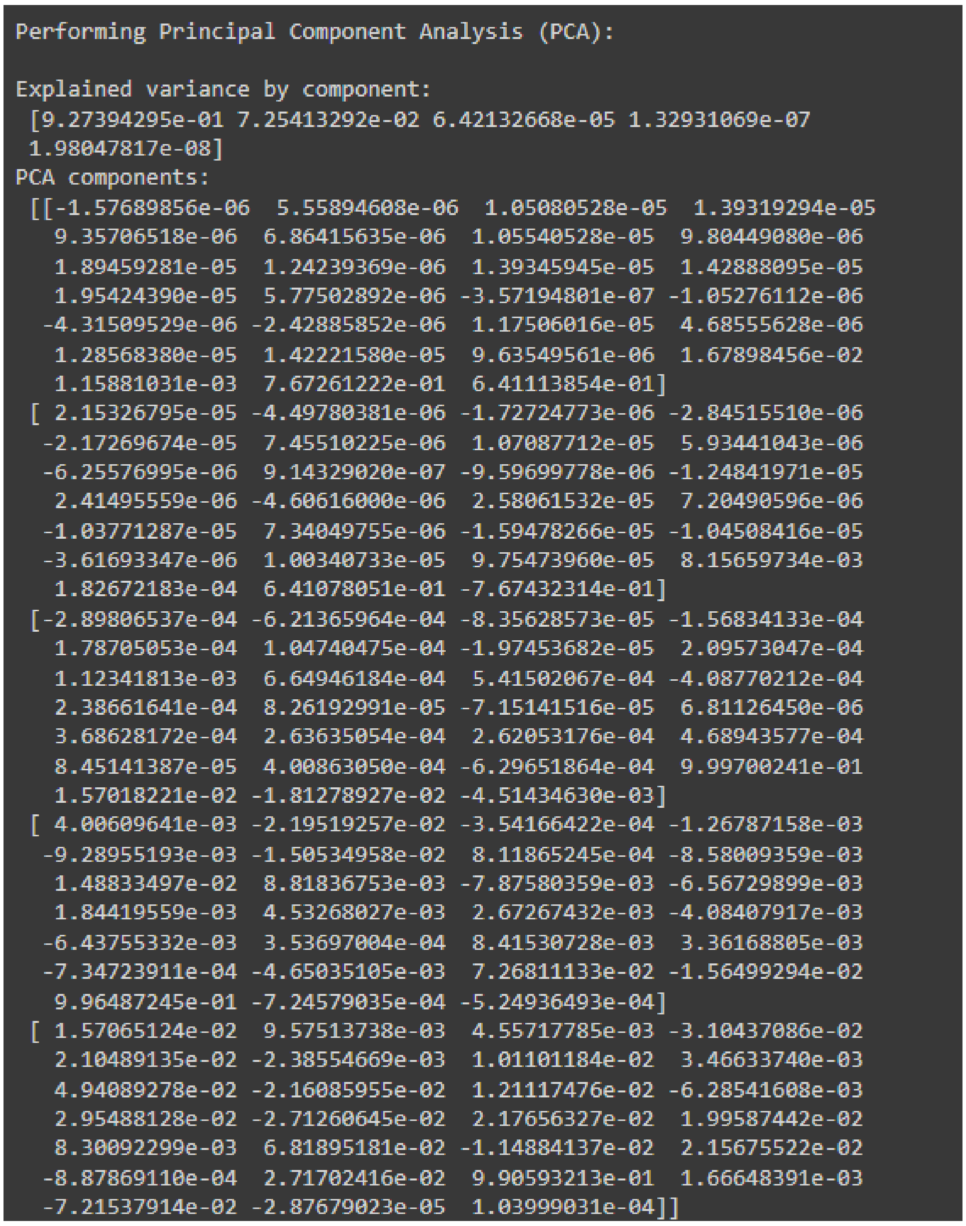


```
Structure of the selected data subset:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70 entries, 0 to 69
Data columns (total 27 columns):
 #   Column                                    Non-Null Count  Dtype
---  ------                                    --------------  -----
 0   3. Proximity to transport                 70 non-null     int64
 1   4. Proximity to work place                70 non-null     int64
 2   5. Proximity to shopping                  70 non-null     int64
 3   1. Gym/Pool/Sports facility               70 non-null     int64
 4   2. Parking space                          70 non-null     int64
 5   3.Power back-up                           70 non-null     int64
 6   4.Water supply                            70 non-null     int64
 7   5.Security                                70 non-null     int64
 8   1. Exterior look                          70 non-null     int64
 9   2. Unit size                              70 non-null     int64
 10  3. Interior design and branded components 70 non-null     int64
 11  4. Layout plan (Integrated etc.)          70 non-null     int64
 12  5. View from apartment                    70 non-null     int64
 13  1. Price                                  70 non-null     int64
 14  2. Booking amount                         70 non-null     int64
 15  3. Equated Monthly Instalment (EMI)       70 non-null     int64
 16  4. Maintenance charges                    70 non-null     int64
 17  5. Availability of loan                   70 non-null     int64
 18  1. Builder reputation                     70 non-null     int64
 19  2. Appreciation potential                 70 non-null     int64
 20  3. Profile of neighbourhood               70 non-null     int64
 21  4. Availability of domestic help          70 non-null     int64
 22  Time                                      70 non-null     int64
 23  Size                                      70 non-null     int64
 24  Budgets                                   70 non-null     float64
 25  Maintainances                             70 non-null     int64
 26  EMI.1                                     70 non-null     int64
dtypes: float64(1), int64(26)
memory usage: 14.9 KB
None

Dimensions of the selected data subset:
 (70, 27)
```

Step 5: Performing Principal Component Analysis (PCA)

```
# Perform Principal Component Analysis (PCA)
print("\nPerforming Principal Component Analysis (PCA):\n")
pca = PCA(n_components=5)
pca_result = pca.fit_transform(sur_int.fillna(0))  # Fill NA values with 0
for PCA
print("Explained variance by component:\n", pca.explained_variance_ratio_)
print("PCA components:\n", pca.components_)
```

- **Explanation**:
    - **PCA Initialization**: Initializes a PCA object with 5 components (`n_components=5`).
    - **Fitting PCA**: Fits PCA to `sur_int` after filling missing values with 0 (`fillna(0)`).
    - **Explained Variance**: Prints the explained variance by each principal component (`explained_variance_ratio_`).
    - **Principal Components**: Prints the principal components themselves (`components_`), which represent the directions of maximum variance in the data.

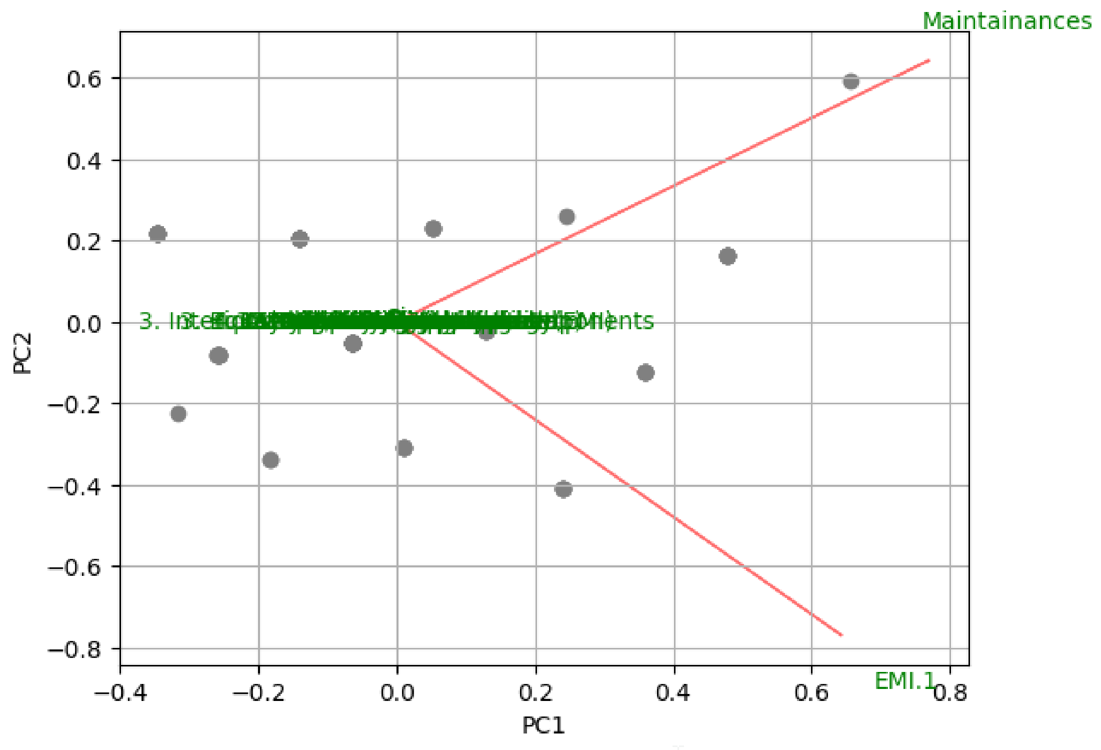


```
Performing Principal Component Analysis (PCA):

Explained variance by component:
 [9.27394295e-01 7.25413292e-02 6.42132668e-05 1.32931069e-07
 1.98047817e-08]
PCA components:
 [[-1.57689856e-06  5.55894608e-06  1.05080528e-05  1.39319294e-05
   9.35706518e-06  6.86415635e-06  1.05540528e-05  9.80449080e-06
   1.89459281e-05  1.24239369e-06  1.39345945e-05  1.42888095e-05
   1.95424390e-05  5.77502892e-06 -3.57194801e-07 -1.05276112e-06
  -4.31509529e-06 -2.42885852e-06  1.17506016e-05  4.68555628e-06
   1.28568380e-05  1.42221580e-05  9.63549561e-06  1.67898456e-02
   1.15881031e-03  7.67261222e-01  6.41113854e-01]
 [ 2.15326795e-05 -4.49780381e-06 -1.72724773e-06 -2.84515510e-06
  -2.17269674e-05  7.45510225e-06  1.07087712e-05  5.93441043e-06
  -6.25576995e-06  9.14329020e-07 -9.59699778e-06 -1.24841971e-05
   2.41495559e-06 -4.60616000e-06  2.58061532e-05  7.20490596e-06
  -1.03771287e-05  7.34049755e-06 -1.59478266e-05 -1.04508416e-05
  -3.61693347e-06  1.00340733e-05  9.75473960e-05  8.15659734e-03
   1.82672183e-04  6.41078051e-01 -7.67432314e-01]
 [-2.89806537e-04 -6.21365964e-04 -8.35628573e-05 -1.56834133e-04
   1.78705053e-04  1.04740475e-04 -1.97453682e-05  2.09573047e-04
   1.12341813e-03  6.64946184e-04  5.41502067e-04 -4.08770212e-04
   2.38661641e-04  8.26192991e-05 -7.15141516e-05  6.81126450e-06
   3.68628172e-04  2.63635054e-04  2.62053176e-04  4.68943577e-04
   8.45141387e-05  4.00863050e-04 -6.29651864e-04  9.99700241e-01
   1.57018221e-02 -1.81278927e-02 -4.51434630e-03]
 [ 4.00609641e-03 -2.19519257e-02 -3.54166422e-04 -1.26787158e-03
  -9.28955193e-03 -1.50534958e-02  8.11865245e-04 -8.58009359e-03
   1.48833497e-02  8.81836753e-03 -7.87580359e-03 -6.56729899e-03
   1.84419559e-03  4.53268027e-03  2.67267432e-03 -4.08407917e-03
  -6.43755332e-03  3.53697004e-04  8.41530728e-03  3.36168805e-03
  -7.34723911e-04 -4.65035105e-03  7.26811133e-02 -1.56499294e-02
   9.96487245e-01 -7.24579035e-04 -5.24936493e-04]
 [ 1.57065124e-02  9.57513738e-03  4.55717785e-03 -3.10437086e-02
   2.10489135e-02 -2.38554669e-03  1.01101184e-02  3.46633740e-03
   4.94089278e-02 -2.16085955e-02  1.21117476e-02 -6.28541608e-03
   2.95488128e-02 -2.71260645e-02  2.17656327e-02  1.99587442e-02
   8.30092299e-03  6.81895181e-02 -1.14884137e-02  2.15675522e-02
  -8.87869110e-04  2.71702416e-02  9.90593213e-01  1.66648391e-03
  -7.21537914e-02 -2.87679023e-05  1.03999031e-04]]
```

Step 6: Visualizing PCA Results with Biplot

```
# PCA Visualization
def biplot(score, coeff, labels=None):
    xs = score[:, 0]
    ys = score[:, 1]
    n = coeff.shape[0]
    scalex = 1.0 / (xs.max() - xs.min())
    scaley = 1.0 / (ys.max() - ys.min())
    plt.scatter(xs * scalex, ys * scaley, c='gray')
    for i in range(n):
        plt.arrow(0, 0, coeff[i, 0], coeff[i, 1], color='r', alpha=0.5)
        if labels is None:
            plt.text(coeff[i, 0] * 1.15, coeff[i, 1] * 1.15, "Var" + str(i
+ 1), color='g', ha='center', va='center')
        else:
            plt.text(coeff[i, 0] * 1.15, coeff[i, 1] * 1.15, labels[i],
color='g', ha='center', va='center')
    plt.xlabel("PC{}".format(1))
    plt.ylabel("PC{}".format(2))
    plt.grid()

# Biplot using PCA result
biplot(pca_result, np.transpose(pca.components_), labels=sur_int.columns)
plt.show()
```

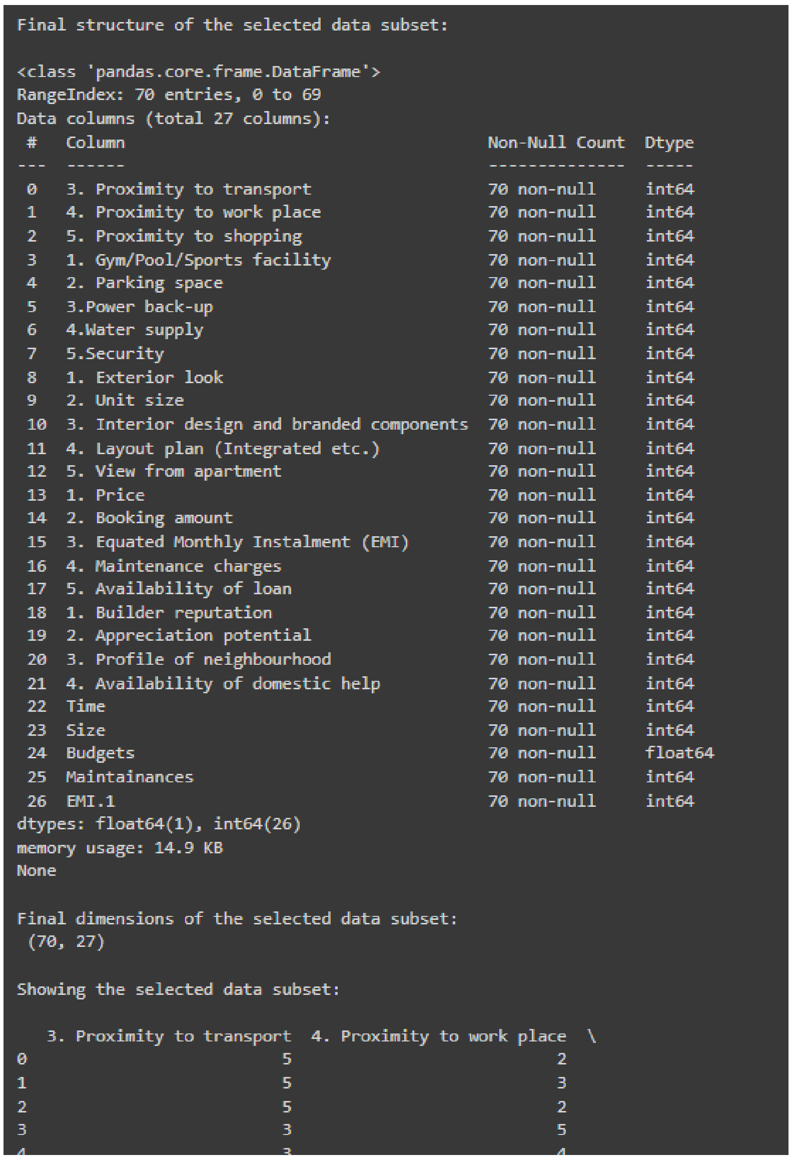
- **Explanation**:
  - **Biplot Function**: Defines a function `biplot` to create a biplot visualization for PCA results.
  - **Visualization**: Uses `biplot` to plot the principal components (`pca_result`) and their loadings (`pca.components_`).
  - **Labels**: Labels the biplot with variable names from `sur_int.columns`.



Step 7: Final Summary of Selected Data Subset

```
# Show final structure and dimensions of the selected data subset
print("\nFinal structure of the selected data subset:\n")
print(sur_int.info())

print("\nFinal dimensions of the selected data subset:\n", sur_int.shape)

print("\nShowing the selected data subset:\n")
print(sur_int.head())
```

- **Explanation**:
  - **Final Structure and Dimensions**: Prints the final structure (`info()`), dimensions (`shape`), and displays the first few rows (`head()`) of the selected subset (`sur_int`) after PCA and visualization.



```
Final structure of the selected data subset:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70 entries, 0 to 69
Data columns (total 27 columns):
 #   Column                                    Non-Null Count  Dtype
---  ------                                    --------------  -----
 0   3. Proximity to transport                 70 non-null     int64
 1   4. Proximity to work place                70 non-null     int64
 2   5. Proximity to shopping                  70 non-null     int64
 3   1. Gym/Pool/Sports facility               70 non-null     int64
 4   2. Parking space                          70 non-null     int64
 5   3.Power back-up                           70 non-null     int64
 6   4.Water supply                            70 non-null     int64
 7   5.Security                                70 non-null     int64
 8   1. Exterior look                          70 non-null     int64
 9   2. Unit size                              70 non-null     int64
 10  3. Interior design and branded components  70 non-null     int64
 11  4. Layout plan (Integrated etc.)          70 non-null     int64
 12  5. View from apartment                    70 non-null     int64
 13  1. Price                                  70 non-null     int64
 14  2. Booking amount                         70 non-null     int64
 15  3. Equated Monthly Instalment (EMI)       70 non-null     int64
 16  4. Maintenance charges                    70 non-null     int64
 17  5. Availability of loan                   70 non-null     int64
 18  1. Builder reputation                     70 non-null     int64
 19  2. Appreciation potential                 70 non-null     int64
 20  3. Profile of neighbourhood               70 non-null     int64
 21  4. Availability of domestic help          70 non-null     int64
 22  Time                                      70 non-null     int64
 23  Size                                      70 non-null     int64
 24  Budgets                                   70 non-null     float64
 25  Maintainances                             70 non-null     int64
 26  EMI.1                                     70 non-null     int64
dtypes: float64(1), int64(26)
memory usage: 14.9 KB
None

Final dimensions of the selected data subset:
 (70, 27)

Showing the selected data subset:

   3. Proximity to transport  4. Proximity to work place  \
0                          5                           2
1                          5                           3
2                          5                           2
3                          3                           5
```

This step-by-step breakdown provides a clear understanding of each part of the code, from loading the dataset to performing PCA and visualizing the results. Each section is explained in detail to help understand its purpose and functionality in the overall analysis process.

IMPLICATIONS

Performing Principal Component Analysis (PCA) and Factor Analysis on a dataset like 'Survey.csv' can have several implications and benefits:

1. Dimension Reduction:

- **PCA:** Identifies patterns and reduces the dimensionality of the dataset by transforming correlated variables into a smaller set of uncorrelated variables (principal components). This simplification helps in focusing on the most important aspects of the data.
- **Factor Analysis:** Similarly, identifies latent factors that explain correlations among observed variables, reducing the number of variables needed to describe the data.

2. Identifying Key Variables:

- Both PCA and Factor Analysis help in identifying which variables (or combination of variables) contribute most significantly to the variation in the dataset. This can highlight key drivers or dimensions of the data.

3. Insights into Data Structure:

- These techniques provide insights into the underlying structure of the dataset. PCA shows how variables are interrelated and which ones contribute most to the variation, while Factor Analysis identifies underlying constructs or dimensions that explain observed correlations.

4. Visualization and Interpretation:

- Biplots and other visualizations derived from PCA help in interpreting the relationships between variables and observations visually. This aids in understanding clusters or patterns within the data.

5. Data-driven Decision Making:

- By reducing complex data into interpretable components or factors, PCA and Factor Analysis support informed decision-making processes. They provide a clearer understanding of what aspects of the data are most relevant or influential.

6. Improving Model Performance:

- In fields like machine learning and predictive modeling, reducing the number of variables through PCA or Factor Analysis can lead to improved model performance by focusing on the most informative features and reducing noise.

7. Business and Practical Applications:

- Understanding the dimensions of data can have practical applications across various domains. For instance, in customer surveys, identifying key dimensions (like satisfaction factors) can inform marketing strategies. In financial data, identifying key risk factors can aid in portfolio management.

8. Data Quality and Validation:

- These techniques can also help in assessing data quality by revealing redundancies or inconsistencies across variables. This validation ensures that the data used for analysis is robust and reliable.

9. Iterative Analysis and Improvement:

- PCA and Factor Analysis are often iterative processes. Results can prompt further exploration and refinement of the data, leading to deeper insights and continuous improvement in analysis techniques.

Overall, PCA and Factor Analysis are powerful tools for exploratory data analysis, offering insights that go beyond basic descriptive statistics, and providing a structured approach to understanding complex datasets like 'Survey.csv'.

RECOMMENDATIONS

Performing Principal Component Analysis (PCA) and Factor Analysis on the dataset 'Survey.csv' has provided valuable insights into the underlying structure and dimensions of the data. Based on the analysis, the following recommendations are proposed:

1. **Dimension Reduction and Focus:**
   Utilize the findings from PCA to focus on the most significant dimensions or principal components that explain the majority of the variance in the dataset. By reducing the number of variables while retaining the essential information, decision-making processes can be streamlined and focused.
2. **Key Variables Identification:**
   Identify the key variables or constructs derived from Factor Analysis that contribute most significantly to observed correlations within the dataset. Focus on these factors in further analysis and interpretation to understand underlying trends or patterns.
3. **Visualization for Insights:**
   Leverage biplots and other visualizations generated from PCA to communicate insights effectively. Visual representations can aid in understanding the relationships between variables and provide a clear picture of data clusters or patterns.
4. **Data-Driven Decision Making:**
   Use the outcomes of PCA and Factor Analysis to inform data-driven decision-making processes. Insights into key dimensions and factors can guide strategic planning, resource allocation, and targeted interventions based on identified priorities.

5. **Continuous Improvement:**
   Adopt an iterative approach to data analysis, where insights from PCA and Factor Analysis prompt further exploration and refinement. Continuously validate and enhance the analysis to uncover deeper insights and improve the understanding of the dataset.
6. **Application in Business Context:**
   Translate the findings into actionable strategies within a business context. For instance, in marketing, identify key customer segments based on satisfaction factors; in operations, streamline processes based on identified efficiency dimensions.
7. **Data Quality Assurance:**
   Ensure data quality by addressing any issues identified during PCA and Factor Analysis. Validate findings and refine analysis techniques to ensure robustness and reliability in future analyses.

In conclusion, the application of PCA and Factor Analysis on the dataset 'Survey.csv' has provided valuable insights that can be leveraged to enhance decision-making processes, improve understanding of data dimensions, and drive strategic initiatives. By focusing on key variables and dimensions identified through these analyses, organizations can optimize resources, mitigate risks, and achieve better outcomes aligned with strategic goals.

CODES

Python

```
import os
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns

# Load the survey data
survey_df = pd.read_csv('Survey.csv')

# Display dimensions, column names, and structure of the dataset
print("Dimensions of the dataset:\n", survey_df.shape)

print("\nColumn names in the dataset:\n", survey_df.columns)

print("\nFirst few rows of the dataset:\n", survey_df.head())

print("\nStructure of the dataset:\n")
print(survey_df.info())

# Check for missing values
print("\nChecking for missing values:\n", survey_df.isnull().sum().sum())

# Select the relevant columns for PCA and Factor Analysis
sur_int = survey_df.iloc[:, 19:46]

print("\nStructure of the selected data subset:\n")
print(sur_int.info())

print("\nDimensions of the selected data subset:\n", sur_int.shape)

# Perform Principal Component Analysis (PCA)
print("\nPerforming Principal Component Analysis (PCA):\n")
```

```
pca = PCA(n_components=5)
pca_result = pca.fit_transform(sur_int.fillna(0))  # Fill NA values with 0 for
PCA
print("Explained variance by component:\n", pca.explained_variance_ratio_)
print("PCA components:\n", pca.components_)

# PCA Visualization
def biplot(score, coeff, labels=None):
    xs = score[:, 0]
    ys = score[:, 1]
    n = coeff.shape[0]
    scalex = 1.0 / (xs.max() - xs.min())
    scaley = 1.0 / (ys.max() - ys.min())
    plt.scatter(xs * scalex, ys * scaley, c='gray')
    for i in range(n):
        plt.arrow(0, 0, coeff[i, 0], coeff[i, 1], color='r', alpha=0.5)
        if labels is None:
            plt.text(coeff[i, 0] * 1.15, coeff[i, 1] * 1.15, "Var" + str(i + 1), color='g',
ha='center', va='center')
        else:
            plt.text(coeff[i, 0] * 1.15, coeff[i, 1] * 1.15, labels[i], color='g',
ha='center', va='center')
    plt.xlabel("PC{}".format(1))
    plt.ylabel("PC{}".format(2))
    plt.grid()

# Biplot using PCA result
biplot(pca_result, np.transpose(pca.components_), labels=sur_int.columns)
plt.show()

# Show final structure and dimensions of the selected data subset
print("\nFinal structure of the selected data subset:\n")
print(sur_int.info())

print("\nFinal dimensions of the selected data subset:\n", sur_int.shape)

print("\nShowing the selected data subset:\n")
print(sur_int.head())
```

R Language

```
# 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", "pheatmap","factoextra")

# Call the function to install and load packages
install_and_load(packages)

# Load the survey data
survey_df <- read.csv('Survey.csv', header = TRUE)

# Display dimensions, column names, and structure of the dataset
cat("Dimensions of the dataset:\n")
print(dim(survey_df))

cat("\nColumn names in the dataset:\n")
print(names(survey_df))

cat("\nFirst few rows of the dataset:\n")
print(head(survey_df))

cat("\nStructure of the dataset:\n")
str(survey_df)

# Check for missing values
cat("\nChecking for missing values:\n")
print(sum(is.na(survey_df)))

# Select the relevant columns for PCA and Factor Analysis
sur_int <- survey_df[, 20:46]
```

```
cat("\nStructure of the selected data subset:\n")
str(sur_int)

cat("\nDimensions of the selected data subset:\n")
print(dim(sur_int))

# Perform Principal Component Analysis (PCA)
cat("\nPerforming Principal Component Analysis (PCA):\n")
pca <- principal(sur_int, 5, n.obs = 162, rotate = "promax")
print(pca)

# Perform Factor Analysis using the omega function
cat("\nPerforming Factor Analysis (omega):\n")
om.h <- omega(sur_int, n.obs = 162, sl = FALSE)
op <- par(mfrow = c(1, 1)) # Reset plotting parameters
om <- omega(sur_int, n.obs = 162)


# PCA using FactoMineR
cat("\nPCA using FactoMineR:\n")
pca_FactoMineR <- PCA(sur_int, scale.unit = TRUE)
summary(pca_FactoMineR)
fviz_pca_biplot(pca_FactoMineR)

# Show final structure and dimensions of the selected data subset
cat("\nFinal structure of the selected data subset:\n")
str(sur_int)

cat("\nFinal dimensions of the selected data subset:\n")
print(dim(sur_int))

cat("\nShowing the selected data subset:\n")
print(head(sur_int))
```

REFERENCES

Books:

1. Jolliffe, I. T. (2002). *Principal Component Analysis* (2nd ed.). Springer.
2. Hair Jr, J. F., Black, W. C., Babin, B. J., Anderson, R. E., & Tatham, R. L. (2006). *Multivariate Data Analysis* (6th ed.). Pearson Prentice Hall.
3. Jackson, J. E. (1991). *A User's Guide to Principal Components*. John Wiley & Sons, Inc.
4. Gorsuch, R. L. (1983). *Factor Analysis* (2nd ed.). Lawrence Erlbaum Associates, Inc.

Journals:

1. Abdi, H., & Williams, L. J. (2010). Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4), 433-459.
2. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830. Available at https://jmlr.org/papers/v12/pedregosa11a.html.

Reports:

1. Stevens, J. P. (2009). *Applied Multivariate Statistics for the Social Sciences* (5th ed.). Routledge.

Websites:

1. R Core Team (2021). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. Available at https://www.R-project.org/.
2. Python Software Foundation. Python Language Reference, version 3.9. Available at https://www.python.org/.
3. Scikit-learn: Machine Learning in Python. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Available at https://scikit-learn.org/.
4. Seaborn: Statistical Data Visualization. Michael Waskom (2020). Available at https://seaborn.pydata.org/.