

ADVANCED DATA ANALYSIS - MIDTERM

Mrunali Vikas Patil

2024-02-14

Question 1 –

Critique Journal Article

1)How are they applying Factor Analysis?

The Article applies Exploratory Factor Analysis (EFA) using Principal Component Analysis (PCA) with Varimax rotation and Kaiser Normalization. It aims to extract underlying components of technostress from a 28-item questionnaire for primary school teachers. The suitability of factor analysis for their data is assessed using the Kaiser-Meyer-Olkin measure and Bartlett's Test of Sphericity. This process results in five components explaining 71.1% of the total variance, with each factor having an eigenvalue greater than 1.0, reflecting a satisfactory level of explained variance for each factor.

2)What kind of rotation do they use?

The Article uses Varimax rotation with Kaiser Normalization in their factor analysis. Varimax is an orthogonal rotation method designed to simplify the interpretation by making the factor structure more parsimonious. It achieves this by maximizing the variance of loadings within factors, which in turn makes each factor more distinct by having a few large loadings and the rest close to zero. This choice facilitates a clearer understanding of which variables strongly relate to which factors, aiding in the meaningful labeling and interpretation of the factor solution.

3)How many components do they concentrate on in their analysis? How did they arrive at these number of components?

The researchers concentrate on five components. They arrived at this number by conducting an Exploratory Factor Analysis (EFA) with Principal Component Analysis (PCA) as the extraction method, and items were retained based on factor loadings of 0.55 or greater. The total variance explained by these components was found to be 71.1%, which is considered satisfactory for the construct being measured. The five components were determined through the analysis and were named based on the thematic content of the items loading onto each component.

4)Explain the breakdown of the components and the significance of their names.

The breakdown of the components in the study and the significance of their names are as follows:

- **Technical:** This component likely represents items related to the technical aspects of technology use, such as hardware and software issues, which can contribute to stress when they malfunction or do not meet teachers' needs.
- **Professional:** This component probably pertains to the professional growth and development aspects of technostress, which may include the pressure to stay updated with new technology or the impact of technology on professional responsibilities.
- **Personal:** This component seems to reflect the personal impact of technology, such as the influence on work-life balance or personal efficiency and organization.
- **Social:** Items falling under this component are likely associated with social interactions and networks, reflecting how technology impacts social dynamics within the school setting or the community.
- **Learning-Teaching Process:** This component appears to cover items related to the core educational processes, such as how technology affects teaching methodologies, learning outcomes, and classroom management.

5) How do they evaluate the stability of the components (i.e. factorability)?

The stability of the components, or factorability, in the study is evaluated using the Kaiser-Meyer-Olkin (KMO) measure and Bartlett's Test of Sphericity. The KMO test measures the adequacy of sampling, with a value of 0.884 indicating that a significant amount of variance in the variables is common and suitable for structure detection. Bartlett's Test of Sphericity confirms the suitability of the data for factor analysis with a significance level below the 0.05 threshold, suggesting that the correlation matrix is not an identity matrix and that the factors are related. These tests together support the factorability of the data.

6) Do they use these components in later analysis, such as regression? If so, what do they discover?

The study conducted Exploratory Factor Analysis (EFA) to create a valid technostress measurement tool for primary school teachers, without extending to regression analysis. Future research may employ these EFA-derived components in regression to assess how technostress influences aspects like job satisfaction, performance, and wellbeing, potentially uncovering significant effects on teachers' professional experiences.

7) What overall conclusions does Principal Component Analysis allow them to draw?

The Principal Component Analysis (PCA) within the study allowed the researchers to draw several conclusions:

Dimensionality of Technostress: PCA helped in identifying five distinct components of technostress among primary school teachers: technical, professional, personal, social, and learning-teaching process oriented.

Variance Explanation: The components explained a significant portion of the variance (71.1%), indicating that the instrument captures a substantial part of the technostress construct.

Item Retention and Rejection: Through PCA, items with factor loadings above a certain threshold were retained, ensuring that only the most relevant items were included in the final measurement instrument.

Internal Consistency: PCA facilitated the assessment of internal consistency of the scale, where Cronbach's alpha values were found to be satisfactory for all the components, suggesting that the items within each component are measuring the same underlying construct.

Instrument Validity: The PCA provided evidence for the construct validity of the instrument, suggesting that it could be a reliable tool for measuring technostress in the context of Malaysian primary school teachers.

QUESTION 2

Define the matrices and vectors.

```
Z <- matrix(c(1, 1, 1, 1, 9,5,-3,11), nrow=4, byrow=FALSE)
Z
```

```
##      [,1] [,2]
## [1,]    1    9
## [2,]    1    5
## [3,]    1   -3
## [4,]    1   11
```

```
Y <- matrix(c(-1, 6, 0, 8), nrow=4, byrow=FALSE)
Y
```

```
##      [,1]
## [1,]   -1
## [2,]    6
## [3,]    0
## [4,]    8
```

```
M <- matrix(c(1, 11, 0,
              42, 52, 35,
              0, 9, 3), nrow=3, byrow=TRUE)
M
```

```
##      [,1] [,2] [,3]
## [1,]    1   11    0
## [2,]   42   52   35
## [3,]    0    9    3
```

```
N <- matrix(c(-10,-10,0,
              0,10,20,
              10,20,10), nrow=3, byrow=TRUE)
```

```
N
```

```
##      [,1] [,2] [,3]
## [1,] -10 -10  0
## [2,]  0  10  20
## [3,]  10  20  10
```

```
v <- matrix(c(-11, 11, 22), nrow=3)
```

```
v
```

```
##      [,1]
## [1,] -11
## [2,]  11
## [3,]  22
```

```
w <- matrix(c(8,-2,4), nrow=3)
```

```
w
```

```
##      [,1]
## [1,]  8
## [2,] -2
## [3,]  4
```

Dot product of vectors v and w

```
dot_product_v_w <- sum(v * w)
```

```
dot_product_v_w
```

```
## [1] -22
```

Result of scalar multiplication of -3 with vector w

```
result_neg_3_times_w <- -3 * w
```

```
result_neg_3_times_w
```

```
##      [,1]
## [1,] -24
## [2,]  6
## [3,] -12
```

Result of matrix-vector multiplication of M and v

```
result_M_times_v <- M %*% v
```

```
result_M_times_v
```

```
##      [,1]
## [1,] 110
```

```
## [2,] 880
## [3,] 165
```

Result of matrix addition of M and N

```
result_M_plus_N <- M + N
result_M_plus_N
```

```
##      [,1] [,2] [,3]
## [1,]  -9   1   0
## [2,]  42  62  55
## [3,]  10  29  13
```

Result of matrix subtraction of M and N

```
result_M_minus_N <- M - N
result_M_minus_N
```

```
##      [,1] [,2] [,3]
## [1,]   11  21   0
## [2,]   42  42  15
## [3,]  -10 -11  -7
```

Result of Z transpose multiplied by Z

```
result_Z_transpose_times_Z <- crossprod(Z)
result_Z_transpose_times_Z
```

```
##      [,1] [,2]
## [1,]    4   22
## [2,]   22  236
```

Inverse of the matrix Z_transpose_times_Z

```
inverse_Z_transpose_times_Z <- solve(result_Z_transpose_times_Z)
inverse_Z_transpose_times_Z
```

```
##      [,1] [,2]
## [1,] 0.51304348 -0.047826087
## [2,] -0.04782609 0.008695652
```

Result of Z transpose multiplied by Y

```
result_Z_transpose_times_Y <- t(Z) %*% Y
result_Z_transpose_times_Y
```

```
##      [,1]
## [1,]   13
## [2,]  109
```

Coefficient matrix B for the linear regression model

```
coefficients_B <- inverse_Z_transpose_times_Z %*% result_Z_transpose_times_Y
coefficients_B

##           [,1]
## [1,]  1.456522
## [2,]  0.326087
```

Determinant of the matrix Z_transpose_times_Z

```
determinant_Z_transpose_times_Z <- det(result_Z_transpose_times_Z)
determinant_Z_transpose_times_Z

## [1] 460
```

Question 3 - Missing Values

Treating missing values involves various strategies, each suitable for different scenarios depending on the nature of the data, the extent of missingness, and the analysis goals. Here's a comprehensive list of methods for treating missing values:

1. Deletion Methods

- Listwise Deletion (Complete Case Analysis): Removing entire records where any single value is missing.
- Pairwise Deletion: Using only the available data for each pairwise calculation, leading to different analyses having different sample sizes.

2. Single Imputation Methods

- Mean/Median/Mode Imputation: Filling in missing values with the average (mean), middle value (median), or most frequent value (mode) of the observed data in the same variable.
- Random Sampling Imputation: Replacing missing values by randomly selecting observed values from the same variable.
- Regression Imputation: Estimating missing values using a regression model based on other variables in the dataset.

- Last Observation Carried Forward (LOCF) & Next Observation Carried Backward (NOCB): Imputing missing values by carrying forward the last observed value or carrying backward the next observed value, often used in time series data.
- Hot Deck Imputation: Replacing missing values with observed values from similar records, where "similarity" is defined based on certain criteria.
- Cold Deck Imputation: Filling missing values with constants or values from other datasets that are assumed to be similar.
- Interpolation and Extrapolation: Estimating missing values using the linear trend of the data points around the missing value.

3. K-Nearest Neighbors (KNN) Imputation

Utilizes the KNN algorithm to impute missing values based on the 'k' most similar instances.

4. Advanced Imputation Methods

- Multiple Imputation: Creating multiple complete datasets by imputing missing values several times, each time creating a complete dataset, and then combining the results for analysis.
- Expectation-Maximization (EM) Algorithm: An iterative process that estimates the most likely values for missing data based on the observed data, often used in conjunction with other statistical models.
- Multiple Imputation by Chained Equations (MICE): A flexible approach that models each variable with missing values as a function of other variables in a round-robin fashion.

5. Machine Learning-Based Methods

- Predictive Modeling: Using machine learning algorithms (e.g., decision trees, random forests, neural networks) to predict and impute missing values based on patterns found in the data.
- Deep Learning Techniques: Employing sophisticated neural network architectures like autoencoders to model the data distribution and impute missing values.

6. Using Indicators

- Missing Indicator Method: Creating a binary variable for each variable with missing values to indicate whether the data was missing, allowing the analysis to incorporate the information about missingness.

7. Imputation Using External Data

- Incorporating Information from External Sources: Imputing missing values based on similar data points or aggregate statistics from external datasets or authoritative sources.

8. Domain-Specific Methods

Custom Imputation: Developing imputation methods tailored to the specific context or domain of the data, taking into account domain knowledge and specific data characteristics.

Selection Criteria

The choice among these methods depends on various factors, including:

- The mechanism of missingness (MCAR, MAR, MNAR).
- The proportion of missing data and its pattern.
- The type of data (continuous, categorical) and its distribution.
- The intended use of the data (exploratory analysis, predictive modeling).
- The complexity of the relationships in the data and computational resources.
- Each method has its strengths and limitations, and the best approach often involves a combination of techniques, sensitivity analysis, and considering the implications of the chosen method on the analysis results.

Examples

- Deletion: In a survey dataset, if a key question has a high non-response rate, listwise deletion can lead to the loss of a significant portion of data, potentially skewing results towards those who responded.
 - Mean Imputation: In a classroom test score dataset, replacing missing scores with the class average can make it appear as though more students scored around the average, reducing the variability and potentially masking underperforming or overperforming students.
 - KNN Imputation: For a dataset with customer demographics and purchasing behavior, using KNN imputation can help accurately fill in missing income values by finding similar customers, preserving the underlying structure of the data.
 - Multiple Imputation: In medical research, where multiple variables can have missing values, MICE can provide a robust way to handle missingness, allowing for the analysis to account for the uncertainty and variability of imputed data.
 - Each of these methods has its place and utility depending on the nature of the data, the mechanism of missingness (Missing Completely At Random, Missing At Random, Missing Not At Random), and the goals of the analysis. Choosing the right strategy is critical to minimizing bias and making the most informed decisions based on the available data.
-

Question 4

In linear regression analysis, it's crucial to check that the assumptions underlying the model hold true for your data. These assumptions are linearity, independence, homoscedasticity, and normality of residuals. Here's how you can use R to check for these assumptions:

1. Linearity

The relationship between the independent variables and the dependent variable should be linear.

#Plotting Residuals vs. Fitted Values:

```
model <- lm(dependent_variable ~ independent_variable1 + independent_variable2, data = dataset)
```

```
plot(model, which = 1)
```

This command plots the residuals (the differences between observed and predicted values) against the fitted (predicted) values. If the relationship is linear, there should be no discernible pattern in the plot.

Adding a Linearity Term and Testing:

We can also add a squared term of the predictor to the model and test if it's significant, indicating non-linearity.

```
model_nonlinear <- lm(dependent_variable ~ independent_variable + I(independent_variable^2), data = dataset)
```

```
summary(model_nonlinear)
```

2. Independence

Observations should be independent of each other.

Durbin-Watson Test:

This test checks for autocorrelation in the residuals. A value close to 2 suggests no autocorrelation.

```
library(lmtest)
```

```
dwtest(model)
```

3. Homoscedasticity

The variance of error terms (residuals) should be constant across all levels of the independent variables.

Residuals vs. Fitted Values Plot:

```
plot(model, which = 1)
```

In this plot, you're looking for a constant spread of residuals across the range of fitted values. A funnel shape (widening or narrowing as you move along the fitted values) indicates heteroscedasticity.

Breusch-Pagan Test:

This test can formally check for heteroscedasticity.

```
library(lmtest)
```

```
bptest(model)
```

4. Normality of Residuals

The residuals should be normally distributed.

Histogram of Residuals:

```
hist(resid(model))
```

This provides a visual check of the normality of residuals.

Q-Q Plot:

```
qqnorm(resid(model))
```

```
qqline(resid(model))
```

This plot shows if the distribution of residuals follows a straight line well, which would confirm their normality.

Shapiro-Wilk Test:

This is a formal test for normality. However, it's sensitive to large sample sizes.

```
shapiro.test(resid(model))
```

Additional Tips

- Always start with visual inspections before moving on to formal tests. Plots can give you a good intuition about possible violations of assumptions.
 - Be mindful of the limitations of formal tests; they can be sensitive to sample size. Very large samples may lead to significant test results for even trivial deviations from assumptions.
 - Consider transformations or alternative models if assumptions are not met. For instance, transforming the dependent variable or using robust regression techniques can sometimes help.
 - Using R to check these assumptions involves a combination of plotting functions and statistical tests, which can provide a comprehensive overview of whether your data meets the necessary criteria for linear regression.
-

Question 5

Ridge and Lasso regressions are two forms of regularized linear regression that include additional constraints on the coefficients to prevent overfitting and manage multicollinearity. Here are their advantages and disadvantages:

Ridge Regression (L2 Regularization):

Advantages:

- It shrinks the coefficients towards zero but does not set any to zero. This results in a model where all variables are included in the model, but their influence is moderated.
- Good for situations where many variables have a small/medium effect on the outcome.
- Can handle multicollinearity well by distributing the coefficient weights among correlated predictors.

Disadvantages:

- It does not perform variable selection; it includes all predictors in the final model, which can be problematic when the goal is interpretability or when only a subset of predictors is truly important.
- The shrinkage parameter (λ) needs to be set optimally via cross-validation, which can be computationally expensive.

Lasso Regression (L1 Regularization):

Advantages:

- Capable of reducing the coefficients for the least important variables to zero, thus performing variable selection and providing a more interpretable model with only a subset of the predictors.
- Useful when you have many variables and expect only a few to influence the outcome.

Disadvantages:

- When there are highly correlated variables, Lasso tends to select one variable from a group and ignore the others.
- If the number of predictors is greater than the number of observations, Lasso selects at most n predictors as non-zero, even if all predictors are relevant.

When to use Ridge vs. Lasso Regression:

Use Ridge Regression when:

- We have a dataset with many multicollinear predictors.
- We want to include all predictors in the model and are not concerned with model simplicity or interpretability.
- We believe that many predictors contribute to the outcome.

Use Lasso Regression when:

- We have a large number of predictors and believe that only a few are actually important.

- We are interested in a more interpretable model that is easier to understand.
 - We want to automatically select features (i.e., variable selection) in the model-building process.
 - Both Ridge and Lasso regression methods can be blended to form Elastic Net regression, which may incorporate the advantages of both methods, depending on the context and the specific dataset at hand.
-

Question 6

- In factor analysis, rotation methods can be used to make the output more interpretable by simplifying the structure of the factor loadings. There are two main types of rotations: orthogonal and oblique.
 - Orthogonal rotations, such as Varimax, assume that the factors are uncorrelated (independent of each other). These rotations preserve the orthogonality of the factors, meaning the axes remain at 90 degrees to each other.
 - Oblique rotations, such as Promax, Direct Oblimin, or Quartimin, allow for correlations between the factors. These rotations do not preserve the orthogonality and can be used when the factors are suspected or known to be correlated.
 - Given that the researcher has found some factors to be correlated with each other, the appropriate choice would be an oblique rotation. Oblique rotations would allow the researcher to understand the nature of the correlations among factors, which could be important for the interpretation of the factors and the underlying theoretical constructs they represent.
-

Question 7

Exploratory Factor Analysis (EFA) and Principal Component Analysis (PCA) are both techniques used for dimensionality reduction, but they have different objectives and interpretations.

Exploratory Factor Analysis (EFA):

Advantages:

- EFA seeks to identify the underlying relationships between measured variables by modeling the observed variables as linear combinations of potential factors, plus error terms. It's focused on the latent structure and common variance among variables.
- EFA is useful when you want to identify groups of variables that are influenced by common factors (latent variables).
- EFA can include rotation methods to improve the interpretability of the factors.

Disadvantages:

- EFA assumes that the underlying structure in the data is due to common factors and unique factors (including error), which may not always be appropriate.
- It can be difficult to determine the correct number of factors to extract.
- EFA models are not unique; different methods can produce different solutions.

Principal Component Analysis (PCA):

Advantages:

- PCA is a technique that converts a set of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. It's focused on maximizing variance and does not distinguish between common and unique variance.
- PCA is a data reduction technique that can summarize the covariance structure with as few principal components as needed to explain most of the variance in the data.
- PCA does not require any assumptions about the underlying structure of the data, such as factor models.

Disadvantages:

- PCA does not model error terms explicitly and treats all variance as being important, which can be an issue if the goal is to identify underlying latent variables.
- The principal components can sometimes be difficult to interpret, as they are linear combinations of all original variables.
- PCA is sensitive to the scaling of the variables; variables must often be standardized, especially when they are measured on different scales.

When to Use EFA vs. PCA:

- Use EFA when the research goal is to identify the underlying latent variables that give rise to the correlations between observed variables. This is common in psychological testing, survey research, and other fields where the concepts of interest are not directly measured.

- Use PCA when the goal is to reduce the dimensionality of the data, such as in image compression or when preparing data for predictive modeling, where the focus is on retaining as much variability as possible, not necessarily uncovering latent variables.
-

Question 8

To analyze a study predicting a student's grade in a class using linear regression, you would need to follow a structured approach. Here's a general outline for a statistical analysis plan:

1. Objective of the Study

- Clearly state the objective, e.g., to predict the final grade of students in a specific class based on various predictors such as study hours, attendance, previous grades, etc.

2. Data Collection

- Describe how the data will be collected, including the sample size, the population from which the sample is drawn, and any inclusion or exclusion criteria for participants.
- Specify the variables to be collected for each student, such as demographic information, educational background, and potentially relevant behavioral factors.

3. Variables of Interest

- Dependent Variable: The student's grade in the class (usually a continuous variable).
- Independent Variables: Factors believed to influence the grade, which might include study hours, class attendance, participation, previous academic performance, etc.

4. Statistical Methods

- Describe the linear regression model, including the equation $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n + \epsilon$, where y is the student's grade, β_0 is the intercept, $\beta_1, \beta_2, \dots, \beta_n$ are the coefficients for each predictor x_1, x_2, \dots, x_n , and ϵ is the error term.
- Explain the assumptions of linear regression, such as linearity, independence, homoscedasticity, and normality of residuals, and how you will assess these assumptions in your data.
- Detail any model diagnostics you will perform, such as checking for multicollinearity using variance inflation factors (VIF), examining residual plots for patterns, and conducting influence diagnostics.

5. Model Evaluation

- Specify the criteria for evaluating the model's performance, such as R-squared, adjusted R-squared, mean squared error (MSE), and root mean squared error (RMSE).
- Discuss any cross-validation techniques you might use to validate the model's predictive power, such as k-fold cross-validation.

6. Sensitivity Analyses

- Describe any planned sensitivity analyses to assess the robustness of your findings, such as repeating the analysis with different sets of predictors or after removing outliers.

Questions for Further Information:

- To refine the analysis plan, you might need answers to the following questions:
- Data Specifics: What specific data is available? Are there any constraints on data collection?
- Time Frame: Over what period will the data be collected? Is this a single term/semester or multiple?
- Contextual Factors: Are there contextual factors that might influence grades and should be considered, such as online vs. in-person classes, subject difficulty, etc.?
- Ethical Considerations: Are there any ethical considerations in data collection, especially concerning student privacy and consent?
- Software and Tools: What statistical software or tools will be used for the analysis?
- By addressing these points in your statistical analysis plan, you ensure a comprehensive approach to predicting student grades using linear regression, accounting for various potential influences and methodological considerations.

QUESTION 9

Load necessary libraries.

```
library(readr)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##   filter, lag
```



```
## The following objects are masked from 'package:base':
##
## intersect, setdiff, setequal, union
```

Load the dataset

```
house_prices <- read_csv("/Users/mrunalipatil/Downloads/home_prices.csv")

## Rows: 545 Columns: 13
## — Column specification —————
## Delimiter: ","
## chr (7): On_mainroad, Has_guestroom, Has_basement, Has_hotwaterheating, Ha
S_...
## dbl (6): price_of_house, area_of_house, number_of_bedrooms, number_of_bath
ro...
##
## ⓘ Use `spec()` to retrieve the full column specification for this data.
## ⓘ Specify the column types or set `show_col_types = FALSE` to quiet this
message.

print(house_prices)


## # A tibble: 545 × 13
##   price_of_house area_of_house number_of_bedrooms number_of_bathrooms
##           <dbl>         <dbl>             <dbl>             <dbl>
## 1      13300000         7420                4                 2
## 2      12250000         8960                4                 4
## 3      12250000         9960                3                 2
## 4      12215000         7500                4                 2
## 5      11410000         7420                4                 1
## 6      10850000         7500                3                 3
## 7      10150000         8580                4                 3
## 8      10150000        16200                5                 3
## 9       9870000         8100                4                 1
## 10     9800000         5750                3                 2
## # ⓘ 535 more rows
## # ⓘ 9 more variables: Number_of_house_stories <dbl>, On_mainroad <chr>,
## #   Has_guestroom <chr>, Has_basement <chr>, Has_hotwaterheating <chr>,
## #   Has_airconditioning <chr>, Number_of_parking_spaces <dbl>,
## #   in_preferred_area <chr>, is_furnished <chr>

dim(house_prices)

## [1] 545 13

head(house_prices)

## # A tibble: 6 × 13
##   price_of_house area_of_house number_of_bedrooms number_of_bathrooms
##           <dbl>         <dbl>             <dbl>             <dbl>
```

```
## 1      13300000      7420      4      2
## 2      12250000      8960      4      4
## 3      12250000      9960      3      2
## 4      12215000      7500      4      2
## 5      11410000      7420      4      1
## 6      10850000      7500      3      3
## #  9 more variables: Number_of_house_stories <dbl>, On_mainroad <chr>,
## #   Has_guestroom <chr>, Has_basement <chr>, Has_hotwaterheating <chr>,
## #   Has_airconditioning <chr>, Number_of_parking_spaces <dbl>,
## #   in_preferred_area <chr>, is_furnished <chr>
```

Check for any missing values in the entire dataset

```
total_missing_values <- sum(is.na(house_prices))
print(paste("Total missing values in the dataset:", total_missing_values))

## [1] "Total missing values in the dataset: 0"
```

Check for missing values in each column

```
missing_values_by_column <- colSums(is.na(house_prices))
print("Missing values by column:")

## [1] "Missing values by column:"

print(missing_values_by_column)

##           price_of_house      area_of_house      number_of_bedrooms
##                0                0                0
##  number_of_bathrooms  Number_of_house_stories      On_mainroad
##                0                0                0
##           Has_guestroom      Has_basement      Has_hotwaterheating
##                0                0                0
##  Has_airconditioning  Number_of_parking_spaces      in_preferred_area
##                0                0                0
##           is_furnished
##                0
```

Loop through each column in the dataset

```
for (col_name in names(house_prices)) {
  # Get unique values for the column
  unique_values <- unique(house_prices[[col_name]])

  # Print unique values along with the column name
  cat("Unique values in", col_name, ":", unique_values, "\n\n")
}

## Unique values in price_of_house : 13300000 12250000 12215000 11410000 1085
## 0000 10150000 9870000 9800000 9681000 9310000 9240000 9100000 8960000 8890000
```

8855000 8750000 8680000 8645000 8575000 8540000 8463000 8400000 8295000 81900
00 8120000 8080940 8043000 7980000 7962500 7910000 7875000 7840000 7700000 75
60000 7525000 7490000 7455000 7420000 7350000 7343000 7245000 7210000 7140000
7070000 7035000 7e+06 6930000 6895000 6860000 6790000 6755000 6720000 6685000
6650000 6629000 6615000 6580000 6510000 6475000 6440000 6419000 6405000 63000
00 6293000 6265000 6230000 6195000 6160000 6125000 6107500 6090000 6083000 60
20000 5950000 5943000 5880000 5873000 5866000 5810000 5803000 5775000 5740000
5652500 5600000 5565000 5530000 5523000 5495000 5460000 5425000 5390000 53830
00 5320000 5285000 5250000 5243000 5229000 5215000 5145000 5110000 5075000 50
40000 5033000 5005000 4970000 4956000 4935000 4907000 4900000 4893000 4865000
4830000 4795000 4767000 4760000 4753000 4690000 4655000 4620000 4613000 45850
00 4550000 4543000 4515000 4480000 4473000 4445000 4410000 4403000 4382000 43
75000 4340000 4319000 4305000 4277000 4270000 4235000 4200000 4193000 4165000
4130000 4123000 4098500 4095000 4060000 4025000 4007500 3990000 3920000 38850
00 3850000 3836000 3815000 3780000 3773000 3745000 3710000 3703000 3675000 36
40000 3633000 3605000 3570000 3535000 3500000 3493000 3465000 3430000 3423000
3395000 3360000 3353000 3332000 3325000 3290000 3255000 3234000 3220000 31500
00 3143000 3129000 3118850 3115000 3087000 3080000 3045000 3010000 3003000 29
75000 2961000 2940000 2870000 2852500 2835000 2800000 2730000 2695000 2660000
2653000 2604000 2590000 2520000 2485000 2450000 2408000 2380000 2345000 23100
00 2275000 2240000 2233000 2135000 2100000 1960000 1890000 1855000 1820000 17
67150 1750000

##

Unique values in area_of_house : 7420 8960 9960 7500 8580 16200 8100 5750
13200 6000 6550 3500 7800 6600 8500 4600 6420 4320 7155 8050 4560 8800 6540 8
875 7950 5500 7475 7000 4880 5960 6840 7482 9000 6360 6480 4300 7440 6325 515
0 11440 7680 8880 6240 11175 7700 12090 4000 5020 4040 4260 6500 5700 10500 3
760 8250 6670 3960 7410 5000 6750 4800 7200 4100 6400 6350 4500 5450 3240 661
5 8372 9620 6800 8000 6900 3700 7020 7231 6254 7320 6525 15600 7160 11460 582
8 5200 5400 4640 5800 6660 4700 5136 4400 3300 3650 6100 2817 7980 3150 6210
6825 6710 6450 10269 8400 5300 3800 9800 8520 6050 7085 3180 3410 3000 11410
5720 3540 7600 10700 8150 4410 7686 2800 5948 4200 4520 4095 4120 4770 6300 2
970 6720 4646 12900 3420 4995 4350 4160 6040 6862 4815 9166 6321 10240 6440 5
170 3630 9667 3745 3880 5680 2870 5010 4510 3840 3640 2550 5320 5360 3520 499
0 3510 3450 9860 5885 3162 3750 3968 4900 2880 4920 4950 3900 1905 4075 4032
10360 3400 2175 4360 7770 6650 2787 5040 5850 2610 2953 2747 2325 4079 2145 4
840 4080 4046 4632 5985 6060 3600 3680 5600 5900 4992 4340 3460 4050 7260 329
0 3816 8080 3780 7152 3850 2015 2176 3350 4820 5830 2856 2520 6930 3480 6020
3584 3120 5640 4280 3570 4130 2850 2275 4240 2135 3036 3990 7424 7350 3512 95
00 5880 12944 3060 3185 1950 4785 4960 4750 3720 3100 2700 4775 2500 3792 393
0 4370 2684 3986 1650 3069 5495 2398 2160 3090 2835 5076 4352 3660 3040 2640
2650 3934 2000 2430 2910 3210 3635 2475 3264 1836 3970 2400 3360 1700 3649 29
90 3620

##

Unique values in number_of_bedrooms : 4 3 5 2 6 1

##

Unique values in number_of_bathrooms : 2 4 1 3

##

Unique values in Number_of_house_stories : 3 4 2 1

##

```
## Unique values in On_mainroad : yes no
##
## Unique values in Has_guestroom : no yes
##
## Unique values in Has_basement : no yes
##
## Unique values in Has_hotwaterheating : no yes
##
## Unique values in Has_airconditioning : yes no
##
## Unique values in Number_of_parking_spaces : 2 3 0 1
##
## Unique values in in_preferred_area : yes no
##
## Unique values in is_furnished : furnished semi-furnished unfurnished
```

Select only numeric columns

```
numeric_data <- house_prices[sapply(house_prices, is.numeric)]
```

Calculate the correlation matrix for numeric variables

```
cor_matrix <- cor(numeric_data, use = "complete.obs") # 'use' argument handles missing values
```

View the correlation matrix

```
print(cor_matrix)
```

##	price_of_house	area_of_house	number_of_bedrooms
## price_of_house	1.0000000	0.53599735	0.3664940
## area_of_house	0.5359973	1.00000000	0.1518585
## number_of_bedrooms	0.3664940	0.15185849	1.0000000
## number_of_bathrooms	0.5175453	0.19381953	0.3739302
## Number_of_house_stories	0.4207124	0.08399605	0.4085642
## Number_of_parking_spaces	0.3843936	0.35298048	0.1392699
##	number_of_bathrooms	Number_of_house_stories	
## price_of_house	0.5175453	0.42071237	
## area_of_house	0.1938195	0.08399605	
## number_of_bedrooms	0.3739302	0.40856424	
## number_of_bathrooms	1.0000000	0.32616471	
## Number_of_house_stories	0.3261647	1.00000000	
## Number_of_parking_spaces	0.1774958	0.04554709	
##	Number_of_parking_spaces		
## price_of_house	0.38439365		
## area_of_house	0.35298048		
## number_of_bedrooms	0.13926990		
## number_of_bathrooms	0.17749582		

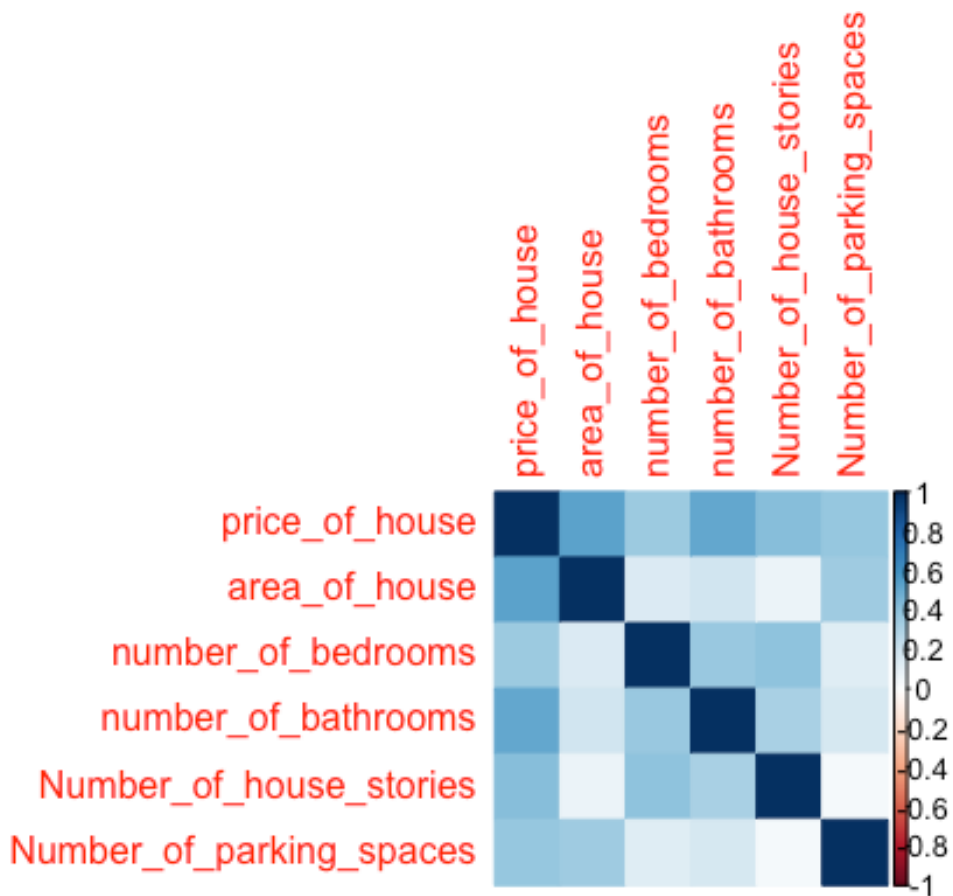
```
## Number_of_house_stories      0.04554709
## Number_of_parking_spaces     1.00000000
```

Visualizing the correlation matrix

```
library(corrplot)

## corrplot 0.92 loaded

corrplot(cor_matrix, method = "color")
```



Heatmap Analysis:

House Price Correlations: The price_of_house seems to be most strongly correlated with area_of_house. This indicates that the larger the area of the house, the higher the price, which is a common trend in real estate.

Feature Interrelations: number_of_bedrooms and number_of_bathrooms appear to have a strong positive correlation with each other, suggesting that houses with more bedrooms also tend to have more bathrooms.

The number_of_house_stories also have a noticeable correlation with both number_of_bedrooms and number_of_bathrooms, where more stories can accommodate more rooms and facilities.

Weaker Correlations: number_of_parking_spaces have a weaker correlation with price_of_house compared to other variables, which might indicate that while parking space is a factor in housing prices, its impact is not as strong as that of the area or the number of rooms.

Load necessary library for VIF

```
library(car)

## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##      recode
```

Prepare the data for VIF calculation

```
predictors <- house_prices %>% select(-price_of_house)
```

Calculate VIF

```
vif_result <- vif(lm(price_of_house ~ ., data = house_prices))
```

Print VIF scores

```
print(vif_result)
```

	GVIF	Df	GVIF^(1/(2*Df))
## area_of_house	1.325250	1	1.151195
## number_of_bedrooms	1.369477	1	1.170246
## number_of_bathrooms	1.286621	1	1.134293
## Number_of_house_stories	1.478055	1	1.215753
## On_mainroad	1.172728	1	1.082926
## Has_guestroom	1.212838	1	1.101289
## Has_basement	1.323050	1	1.150239
## Has_hotwaterheating	1.041506	1	1.020542
## Has_airconditioning	1.211840	1	1.100836
## Number_of_parking_spaces	1.212837	1	1.101289
## in_preferred_area	1.149196	1	1.072006
## is_furnished	1.109639	2	1.026350

VIF Scores Analysis:

Low Multicollinearity: All VIF scores are close to 1 (ranging from about 1.02 to 1.22 for the square root of the GVIF), which shows that there is no significant multicollinearity among

these variables. It indicates that every variable provides a regression model with distinct information.

Stability of Regression Coefficients: The regression coefficients for these variables in a model predicting house prices are likely to be stable, and changes in the model should not lead to large fluctuations in these coefficients.

Concerns and Corrections: There are no immediate concerns regarding multicollinearity among the variables from the information given by heatmap and VIF Scores. The relationships between the variables are as expected, with no red flags requiring corrective action. Therefore, there is no need to remove any variables to correct for multicollinearity, nor we need to combine variables or apply techniques like PCA or regularization due to multicollinearity issues.

For building a Regression Model, we can proceed with the understanding that each one brings unique information to the table. The next steps analysis likely includes fitting a regression model to the data and then validating the model's assumptions, checking for outliers and leverage points.

Performing Linear Regression

```
model <- lm(price_of_house ~ ., data = house_prices)
```

Summarizing the model

```
summary(model)
```

```
##
## Call:
## lm(formula = price_of_house ~ ., data = house_prices)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2619718 -657322  -68409   507176  5166695
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    42771.69   264313.31    0.162  0.871508
## area_of_house      244.14     24.29   10.052 < 2e-16 ***
## number_of_bedrooms 114787.56   72598.66    1.581  0.114445
## number_of_bathrooms 987668.11  103361.98    9.555 < 2e-16 ***
## Number_of_house_stories 450848.00   64168.93    7.026 6.55e-12 ***
## On_mainroadyes    421272.59  142224.13    2.962  0.003193 **
## Has_guestroomyes   300525.86  131710.22    2.282  0.022901 *
## Has_basementyes    350106.90  110284.06    3.175  0.001587 **
## Has_hotwaterheatingyes 855447.15  223152.69    3.833  0.000141 ***
## Has_airconditioningyes 864958.31  108354.51    7.983 8.91e-15 ***
## Number_of_parking_spaces 277107.10   58525.89    4.735 2.82e-06 ***
## in_preferred_areayes 651543.80  115682.34    5.632 2.89e-08 ***
## is_furnishedsemi-furnished -46344.62  116574.09   -0.398  0.691118
```

```
## is_furnishedunfurnished      -411234.39  126210.56  -3.258 0.001192 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1068000 on 531 degrees of freedom
## Multiple R-squared:  0.6818, Adjusted R-squared:  0.674
## F-statistic: 87.52 on 13 and 531 DF,  p-value: < 2.2e-16
```

R-squared and Adjusted R-squared:

The Multiple R-squared value is 0.6818, indicating that approximately 68.18% of the variance in house prices is explained by the model. This is a relatively high value, showing that the model has a good fit.

The Adjusted R-squared is 0.674, which is slightly lower than the Multiple R-squared but still indicates a good fit. The adjustment accounts for the number of predictors in the model, providing a more accurate measure of the model's explanatory power when comparing models with different numbers of predictors.

F-Test: The F-statistic is 87.52, with an associated p-value of less than 2.2e-16. This is extremely low, well below a significance level of 0.05, indicating that the model is statistically significant. In other words, there is a very low probability that the model's predictive capability is due to chance, and it suggests that the model provides a better fit to the data than a model with no independent variables.

Significant Variables: Variables with a p-value less than 0.05 are considered statistically significant. In this model, the significant variables are:

Get the summary of the model

```
model_summary <- summary(model)
```

Get the coefficients table

```
coefficients_table <- model_summary$coefficients
```

Filter for significant variables at the 5% significance level (p-value < 0.05)

```
significant_variables <- coefficients_table[coefficients_table[, "Pr(>|t|)"]
< 0.05, ]
```

Print the names and coefficients of the significant variables

```
print(significant_variables)
```


	Estimate	Std. Error	t value	Pr(> t)
## area_of_house	244.1394	24.28881	10.051516	7.068685e-22
## number_of_bathrooms	987668.1073	103361.97842	9.555430	4.527066e-20
## Number_of_house_stories	450848.0029	64168.93089	7.025955	6.547634e-12
## On_mainroadyes	421272.5887	142224.12884	2.962033	3.192999e-03
## Has_guestroomyes	300525.8596	131710.21941	2.281720	2.290075e-02
## Has_basementyes	350106.9041	110284.05681	3.174592	1.587481e-03
## Has_hotwaterheatingyes	855447.1454	223152.68961	3.833461	1.414948e-04
## Has_airconditioningyes	864958.3113	108354.51133	7.982670	8.906006e-15
## Number_of_parking_spaces	277107.1013	58525.88835	4.734778	2.817182e-06
## in_preferred_areayes	651543.7999	115682.33692	5.632180	2.888201e-08
## is_furnishedunfurnished	-411234.3862	126210.55804	-3.258320	1.192433e-03

Interpretation of Beta Coefficients: The beta coefficient for area_of_house is 244.14, which means that with every additional square foot of area, the expected price of the house increases by approximately 244.14 units of currency. With a beta coefficient of 987668.11 for number_of_bathrooms, the model predicts that adding one more bathroom to a house increases its price by roughly 987,668 units of currency. The is_furnishedunfurnished variable has a negative beta coefficient of -411234.39, indicating that an unfurnished house is expected to sell for about 411,234 units of currency less than its furnished or semi-furnished counterparts.

Conclusion: The model is a good fit, with significant predictors explaining a substantial proportion of the variance in house prices. The significant variables and their coefficients can inform stakeholders of the features that contribute to house valuations, such as the size of the house, number of amenities, and location-related attributes.

Load the glmnet package

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

Prepare the matrix of predictors and the response variable

```
x <- as.matrix(house_prices[, -which(names(house_prices) == "price_of_house")
]) # predictors
y <- house_prices$price_of_house # response
```

Fit the lasso model

```
lasso_model <- glmnet(x, y, alpha = 1)
```

```
## Warning in storage.mode(xd) <- "double": NAs introduced by coercion
```

Use cross-validation to choose the best lambda

[illegible]

Look at the coefficients at the best lambda

```
best_lambda <- cv_lasso$lambda.min
lasso_coefficients <- coef(cv_lasso, s = best_lambda)

print(lasso_coefficients)

## 13 x 1 sparse Matrix of class "dgCMatrix"
##                               s1
## (Intercept)                -113195.6416
## area_of_house                 329.6536
## number_of_bedrooms            164618.8407
## number_of_bathrooms           1129246.1739
## Number_of_house_stories       544079.5016
## On_mainroad                   .
## Has_guestroom                 .
## Has_basement                  .
## Has_hotwaterheating           .
## Has_airconditioning           .
## Number_of_parking_spaces      373563.0022
## in_preferred_area             .
## is_furnished                  .
```

Benefits of Lasso Regression:

Feature Selection: Lasso performs L1 regularization which can shrink some coefficients to zero, effectively performing feature selection and removing some variables from the model.

Model Interpretability: By eliminating irrelevant features, Lasso helps in creating more interpretable models, which is beneficial when you want to explain your model to stakeholders.

Handling Multicollinearity: Lasso can handle multicollinearity well by keeping only one variable from a group of highly correlated variables and eliminating the others.

Prediction Accuracy: In situations where the dataset has numerous features, Lasso can improve prediction accuracy by avoiding overfitting.

Costs of Lasso Regression: Bias: Lasso introduces bias into the model with its shrinkage method, which can be an issue if the true relationship is complex.

Selection of Lambda: The selection of the regularization parameter, lambda, is crucial. If lambda is too large, it can overshrink the coefficients. Determining the right lambda usually requires cross-validation, which can be computationally intensive.

Loss of Variables: Lasso might exclude variables that are important, especially when there is a lot of noise in the dataset.

Interpretation of Coefficients: The interpretation of the coefficients becomes less straightforward because they are shrunk towards zero, and the scale of the predictors affects the shrinkage.

Load necessary libraries

```
library(ggplot2)
library(readr)
```

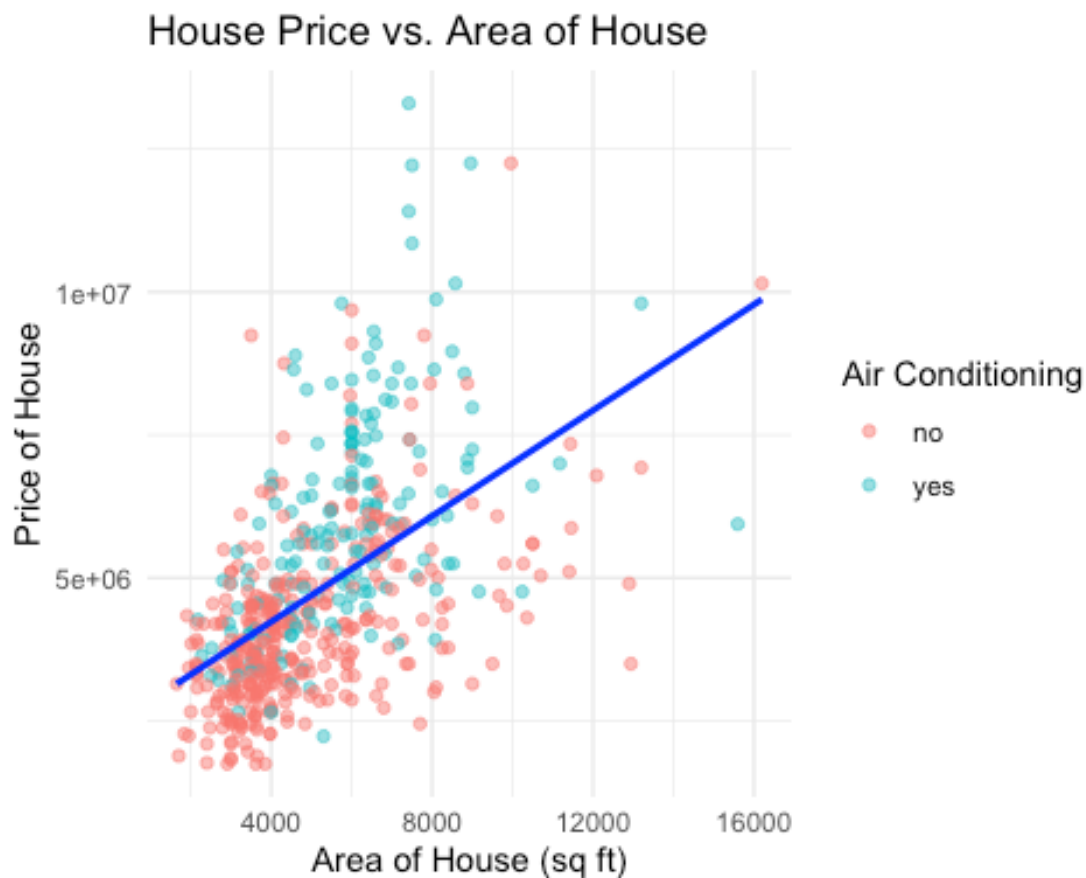
Visualization 1: Price vs. Area of House

```
p1 <- ggplot(house_prices, aes(x = area_of_house, y = price_of_house)) +
  geom_point(aes(color = factor(Has_airconditioning)), alpha = 0.5) +
  geom_smooth(method = "lm", color = "blue", se = FALSE) +
  labs(title = "House Price vs. Area of House",
       x = "Area of House (sq ft)",
       y = "Price of House",
       color = "Air Conditioning") +
  theme_minimal()
```

Print the plots

```
print(p1)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Convert numerical variables to factors

```
house_prices$number_of_bedrooms <- as.factor(house_prices$number_of_bedrooms)
house_prices$number_of_bathrooms <- as.factor(house_prices$number_of_bathrooms)
house_prices$Number_of_house_stories <- as.factor(house_prices$Number_of_house_stories)
house_prices$Number_of_parking_spaces <- as.factor(house_prices$Number_of_parking_spaces)
```

Verify the changes

```
str(house_prices)

## spc_tbl_ [545 × 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ price_of_house      : num [1:545] 13300000 12250000 12250000 12215000 11410000 ...
##  $ area_of_house       : num [1:545] 7420 8960 9960 7500 7420 7500 8580 16200 8100 5750 ...
##  $ number_of_bedrooms   : Factor w/ 6 levels "1","2","3","4",...: 4 4 3 4 4 3 4 5 4 3 ...
##  $ number_of_bathrooms  : Factor w/ 4 levels "1","2","3","4": 2 4 2 2 1 3 3 3 1 2 ...
##  $ Number_of_house_stories : Factor w/ 4 levels "1","2","3","4": 3 4 2 2 2 1 4 2 2 4 ...
##  $ On_mainroad          : chr [1:545] "yes" "yes" "yes" "yes" ...
##  $ Has_guestroom        : chr [1:545] "no" "no" "no" "no" ...
##  $ Has_basement         : chr [1:545] "no" "no" "yes" "yes" ...
##  $ Has_hotwaterheating  : chr [1:545] "no" "no" "no" "no" ...
##  $ Has_airconditioning  : chr [1:545] "yes" "yes" "no" "yes" ...
##  $ Number_of_parking_spaces: Factor w/ 4 levels "0","1","2","3": 3 4 3 4 3 3 3 1 3 2 ...
##  $ in_preferred_area    : chr [1:545] "yes" "no" "yes" "yes" ...
##  $ is_furnished         : chr [1:545] "furnished" "furnished" "semi-furnished" "furnished" ...
##  - attr(*, "spec")=
##    .. cols(
##    ..   price_of_house = col_double(),
##    ..   area_of_house = col_double(),
##    ..   number_of_bedrooms = col_double(),
##    ..   number_of_bathrooms = col_double(),
##    ..   Number_of_house_stories = col_double(),
##    ..   On_mainroad = col_character(),
##    ..   Has_guestroom = col_character(),
##    ..   Has_basement = col_character(),
##    ..   Has_hotwaterheating = col_character(),
##    ..   Has_airconditioning = col_character(),
##    ..   Number_of_parking_spaces = col_double(),
##    ..   in_preferred_area = col_character(),
##    ..   is_furnished = col_character()
```

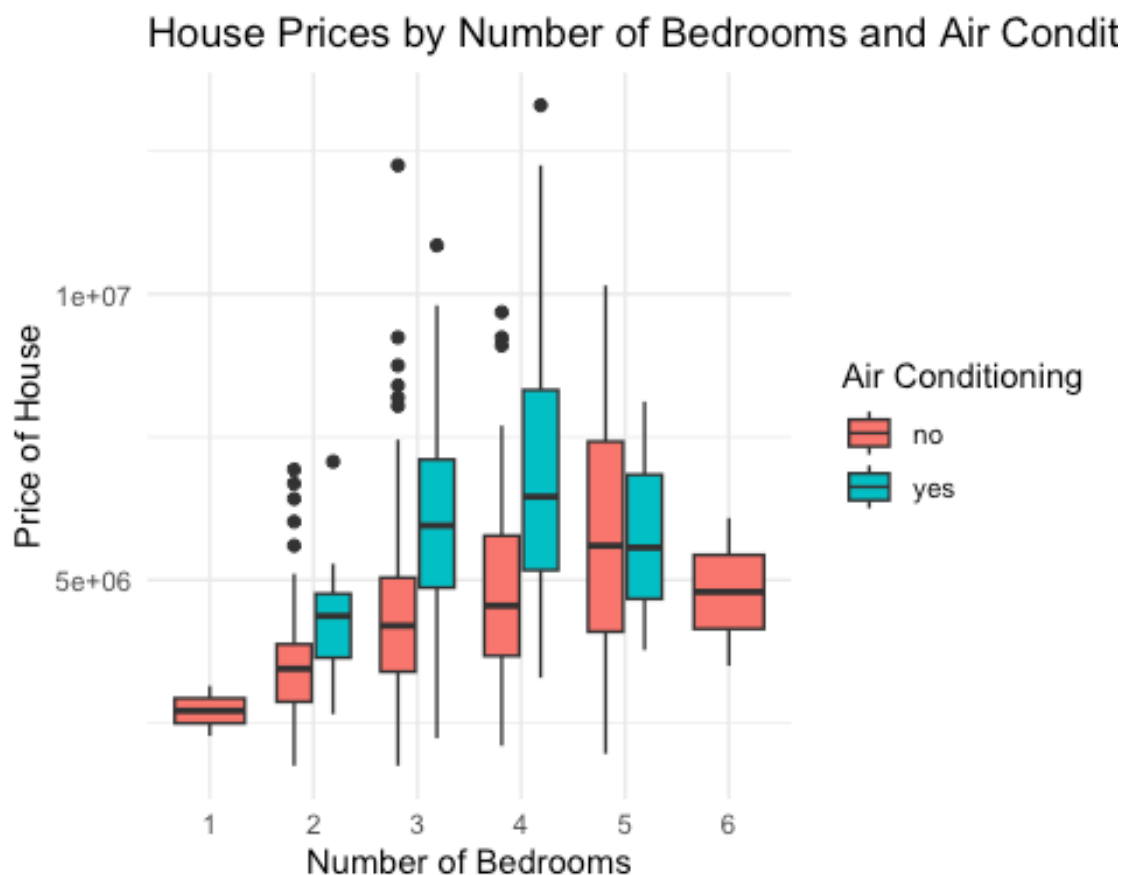
```
## .. )
## - attr(*, "problems")=<externalptr>
```

Load necessary library

```
library(ggplot2)
```

Box plot for Number of Bedrooms vs House Prices, grouped by Air Conditioning

```
ggplot(house_prices, aes(x = number_of_bedrooms, y = price_of_house, fill = factor(Has_airconditioning))) +
  geom_boxplot() +
  labs(title = "House Prices by Number of Bedrooms and Air Conditioning",
       x = "Number of Bedrooms",
       y = "Price of House",
       fill = "Air Conditioning") +
  theme_minimal()
```



The plot is a boxplot that displays the range of house prices categorized by the number of bedrooms, with a separate color indicating whether the house has air conditioning.

Prices increase with the number of bedrooms, and for any given number of bedrooms, houses with air conditioning tend to be priced higher.

Outliers are present, especially in houses with more bedrooms, indicating significant variation in prices at higher bedroom counts.

Box plot for Number of Bathrooms vs House Prices, grouped by Air Conditioning

```
ggplot(house_prices, aes(x = number_of_bathrooms, y = price_of_house, fill = factor(Has_airconditioning))) +  
  geom_boxplot() +  
  labs(title = "House Prices by Number of Bathrooms and Air Conditioning",  
       x = "Number of Bathrooms",  
       y = "Price of House",  
       fill = "Air Conditioning") +  
  theme_minimal()
```



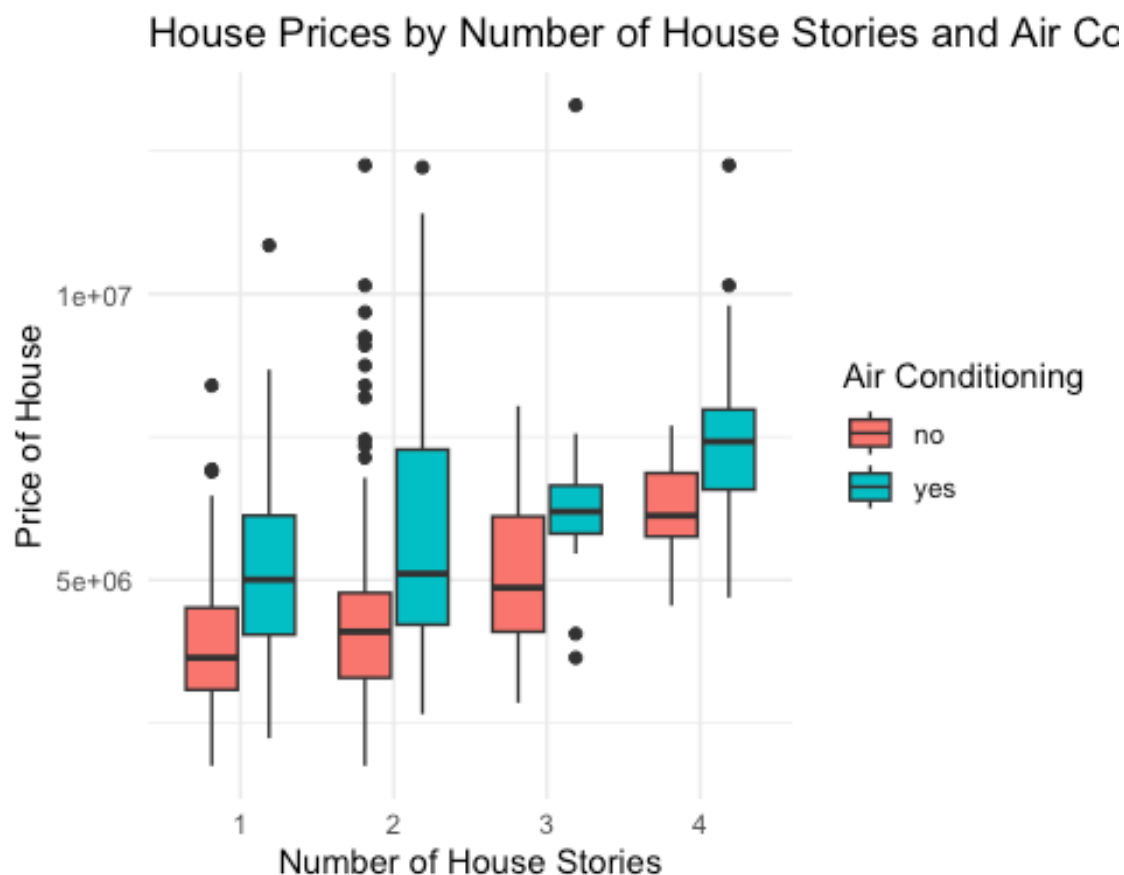
The plot is a boxplot that illustrates house prices in relation to the number of bathrooms, differentiated by whether a house has air conditioning or not.

As the number of bathrooms increases, the median house price generally increases.

Houses with air conditioning appear to have a higher median price compared to those without, across all bathroom counts.

Box plot for Number of House Stories vs House Prices, grouped by Air Conditioning

```
ggplot(house_prices, aes(x = Number_of_house_stories, y = price_of_house, fill = factor(Has_airconditioning))) +  
  geom_boxplot() +  
  labs(title = "House Prices by Number of House Stories and Air Conditioning")  
,  
  x = "Number of House Stories",  
  y = "Price of House",  
  fill = "Air Conditioning") +  
  theme_minimal()
```



The plot is a boxplot depicting the distribution of house prices based on the number of stories in a house, with a distinction between houses with and without air conditioning.

The median house prices don't show a consistent trend with the increase in the number of stories. Houses with air conditioning generally have higher median prices across the

different numbers of stories, but the difference is not as pronounced in houses with more stories.

Box plot for Number of Parking Spaces vs House Prices, grouped by Air Conditioning

```
ggplot(house_prices, aes(x = Number_of_parking_spaces, y = price_of_house, fill = factor(Has_airconditioning))) +  
  geom_boxplot() +  
  labs(title = "House Prices by Number of Parking Spaces and Air Conditioning",  
        x = "Number of Parking Spaces",  
        y = "Price of House",  
        fill = "Air Conditioning") +  
  theme_minimal()
```



The plot is a boxplot visualizing house prices as they relate to the number of parking spaces, with color coding to distinguish between houses with and without air conditioning. Interestingly, the median house prices appear higher for homes with no parking spaces compared to those with one or two, and there's a significant jump in price for houses with three parking spaces. Air conditioning seems to be associated with higher median prices, but not uniformly across all categories of parking spaces.

QUESTION 10

Load necessary libraries

```
library(readr) # For reading CSV files
library(stats) # For PCA
library(tidyverse)

## — Attaching core tidyverse packages — tidyverse 2.0.0 —
## ✓ forcats 1.0.0      ✓ stringr 1.5.0
## ✓ lubridate 1.9.3    ✓ tibble 3.2.1
## ✓ purrr 1.0.2       ✓ tidyr 1.3.0
## — Conflicts — tidyverse_conflicts() —
## ✗ tidyr::expand() masks Matrix::expand()
## ✗ dplyr::filter() masks stats::filter()
## ✗ dplyr::lag() masks stats::lag()
## ✗ tidyr::pack() masks Matrix::pack()
## ✗ car::recode() masks dplyr::recode()
## ✗ purrr::some() masks car::some()
## ✗ tidyr::unpack() masks Matrix::unpack()
## ⓘ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(factoextra)

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

library(dplyr) # For data manipulation
```

#Read the data

```
data <- read_csv("/Users/mrunalipatil/Downloads/16PF.csv")

## Rows: 49159 Columns: 163
## — Column specification —
## Delimiter: ","
## dbf (163): A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, B1, B2, B3, B4, B5, B6, ...
## ⓘ Use `spec()` to retrieve the full column specification for this data.
## ⓘ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
dim(data)
```

```
## [1] 49159 163
```

Replace 0s with NA (to denote missing values)

```
data <- data %>% mutate(across(everything(), ~na_if(.x, 0)))
```

#Check NA for all variables

```
sum(is.na(data))
```

```
## [1] 98919
```

#Check for NA percentages in each column

```
na_percentages <- colSums(is.na(data)) / nrow(data) * 100  
print(na_percentages)
```

```
##      A1      A2      A3      A4      A5      A6      A7  
A8  
## 1.4280193 1.4849773 1.2551110 1.4748062 0.9316707 1.0476210 0.6590858 1.45  
24299  
##      A9      A10      B1      B2      B3      B4      B5  
B6  
## 1.5337985 1.1920503 1.4361561 1.1879819 1.0883053 0.9743892 1.5460038 1.58  
05854  
##      B7      B8      B9      B10      B11      B12      B13  
C1  
## 1.4707378 1.0557578 1.4788747 1.3425822 0.6712911 1.2530768 1.0313473 0.97  
03208  
##      C2      C3      C4      C5      C6      C7      C8  
C9  
## 1.1432291 1.6192355 1.1086475 0.6916333 1.4442930 1.4646352 0.6692569 1.03  
54157  
##      C10      D1      D2      D3      D4      D5      D6  
D7  
## 1.2774873 0.8503021 1.0191420 1.2123924 1.3954718 0.7709677 1.5988934 0.91  
94654  
##      D8      D9      D10      E1      E2      E3      E4  
E5  
## 1.1167843 0.7994467 1.2978295 1.0232104 1.4626009 1.4036087 0.9947314 1.45  
03957  
##      E6      E7      E8      E9      E10      F1      F2  
F3  
## 1.1656055 1.3975061 1.1534002 1.3100348 1.2530768 1.4137798 1.2795216 0.88  
08153  
##      F4      F5      F6      F7      F8      F9      F10  
G1  
## 1.0130393 1.1696739 1.4117456 1.3283427 1.3059664 1.3242743 1.4605667 1.16  
35713
```

##	G2	G3	G4	G5	G6	G7	G8
G9							
##	1.4402246	1.3405480	1.4910800	1.3364796	0.8665758	0.7974125	1.4666694 0.9764234
##	G10	H1	H2	H3	H4	H5	H6
H7							
##	1.1066132	1.3954718	1.3791981	1.2164609	1.3202059	1.3669928	1.2225635 1.3425822
##	H8	H9	H10	I1	I2	I3	I4
I5							
##	1.3486849	1.4727720	1.4280193	1.1635713	1.3588560	1.4666694	0.9967656 1.4788747
##	I6	I7	I8	I9	I10	J1	J2
J3							
##	1.4137798	1.2042556	0.6916333	1.5358327	1.2205293	1.1981529	0.8991233 1.3812323
##	J4	J5	J6	J7	J8	J9	J10
K1							
##	1.0557578	1.0598263	1.3975061	1.0333815	1.3690270	1.3669928	1.3710613 1.5032853
##	K2	K3	K4	K5	K6	K7	K8
K9							
##	1.0455868	0.9886287	1.3995403	1.5297301	1.5154906	1.5521064	1.4992168 1.1635713
##	K10	L1	L2	L3	L4	L5	L6
L7							
##	0.8909864	1.0781342	1.4564983	1.2266319	1.1005106	1.5093879	0.8340284 1.0557578
##	L8	L9	L10	M1	M2	M3	M4
M5							
##	1.3425822	1.1350923	1.0028682	0.8238573	1.6558514	1.3425822	1.4076771 1.1554344
##	M6	M7	M8	M9	M10	N1	N2
N3							
##	1.4320877	1.5500722	1.4971826	1.4463272	1.4320877	1.3547875	1.1778108 1.3425822
##	N4	N5	N6	N7	N8	N9	N10
O1							
##	1.4117456	0.9540471	1.0496552	0.8055493	1.0699974	1.3853008	0.7262149 1.3954718
##	O2	O3	O4	O5	O6	O7	O8
O9							
##	1.3975061	1.4381904	1.1839134	1.1595028	1.5460038	1.4809089	1.3568217 1.2530768
##	O10	P1	P2	P3	P4	P5	P6
P7							
##	1.5134563	1.2245977	0.9235338	1.0842369	1.4259851	1.3629244	1.2652820 1.1147501
##	P8	P9	P10				
##	0.9967656	0.7730019	1.1513660				

Count the number of rows before imputation

```
n_rows_before <- nrow(data)
```

Function to impute NA values

```
impute_na <- function(data, method = "mean") {  
  if (method == "mean") {  
    # Impute using mean  
    imputed_data <- data %>% mutate(across(where(is.numeric), ~ifelse(is.na(.  
) , mean(., na.rm = TRUE), .)))  
  } else if (method == "median") {  
    # Impute using median  
    imputed_data <- data %>% mutate(across(where(is.numeric), ~ifelse(is.na(.  
) , median(., na.rm = TRUE), .)))  
  } else if (method == "mode") {  
    # Impute using mode (most frequent value)  
    mode_impute <- function(x) {  
      ux <- unique(x)  
      ux[which.max(tabulate(match(x, ux)))]  
    }  
    imputed_data <- data %>% mutate(across(where(is.numeric), ~ifelse(is.na(.  
) , mode_impute(.), .)))  
  } else {  
    stop("Invalid method. Choose 'mean', 'median', or 'mode'.")  
  }  
  return(imputed_data)  
}
```

Impute NA values using the desired method

```
imputed_data <- impute_na(data, method = "mean") # You can change "mean" to "  
median" or "mode"
```

Check the result

```
summary(imputed_data)
```

##	A1	A2	A3	A4
##	Min. :1.000	Min. :1.000	Min. :1.000	Min. :1.000
##	1st Qu.:3.000	1st Qu.:3.000	1st Qu.:3.000	1st Qu.:3.000
##	Median :4.000	Median :4.000	Median :4.000	Median :4.000
##	Mean :3.702	Mean :3.849	Mean :3.847	Mean :3.688
##	3rd Qu.:4.000	3rd Qu.:4.000	3rd Qu.:5.000	3rd Qu.:4.000
##	Max. :5.000	Max. :5.000	Max. :5.000	Max. :5.000
##	A5	A6	A7	A8
##	Min. :1.000	Min. :1.000	Min. :1.000	Min. :1.000
##	1st Qu.:3.000	1st Qu.:3.000	1st Qu.:4.000	1st Qu.:2.000
##	Median :4.000	Median :4.000	Median :4.000	Median :3.000

## Mean :3.865	Mean :3.709	Mean :3.896	Mean :3.004
## 3rd Qu.:4.000	3rd Qu.:4.000	3rd Qu.:4.000	3rd Qu.:4.000
## Max. :5.000	Max. :5.000	Max. :5.000	Max. :5.000
## A9	A10	B1	B2
## Min. :1.000	Min. :1.000	Min. :1.000	Min. :1.000
## 1st Qu.:1.000	1st Qu.:2.000	1st Qu.:3.000	1st Qu.:3.000
## Median :2.000	Median :2.000	Median :4.000	Median :4.000
## Mean :2.174	Mean :2.458	Mean :3.731	Mean :3.537
## 3rd Qu.:3.000	3rd Qu.:3.000	3rd Qu.:4.000	3rd Qu.:4.000
## Max. :5.000	Max. :5.000	Max. :5.000	Max. :5.000
## B3	B4	B5	B6
## Min. :1.000	Min. :1.000	Min. :1.000	Min. :1.000
## 1st Qu.:4.000	1st Qu.:4.000	1st Qu.:4.000	1st Qu.:3.000
## Median :4.000	Median :4.000	Median :4.000	Median :4.000
## Mean :4.189	Mean :4.358	Mean :4.039	Mean :3.553
## 3rd Qu.:5.000	3rd Qu.:5.000	3rd Qu.:5.000	3rd Qu.:4.000
## Max. :5.000	Max. :5.000	Max. :5.000	Max. :5.000
## B7	B8	B9	B10
## Min. :1.000	Min. :1.000	Min. :1.000	Min. :1.000
## 1st Qu.:3.000	1st Qu.:3.000	1st Qu.:2.000	1st Qu.:2.000
## Median :4.000	Median :4.000	Median :3.000	Median :2.000
## Mean :3.558	Mean :3.832	Mean :3.033	Mean :2.645
## 3rd Qu.:4.000	3rd Qu.:4.000	3rd Qu.:4.000	3rd Qu.:4.000
## Max. :5.000	Max. :5.000	Max. :5.000	Max. :5.000
## B11	B12	B13	C1
## Min. :1.000	Min. :1.000	Min. :1.000	Min. :1.000
## 1st Qu.:2.000	1st Qu.:1.000	1st Qu.:1.000	1st Qu.:2.000
## Median :3.000	Median :2.000	Median :2.000	Median :3.000
## Mean :2.742	Mean :1.936	Mean :2.175	Mean :2.837
## 3rd Qu.:4.000	3rd Qu.:2.000	3rd Qu.:3.000	3rd Qu.:4.000
## Max. :5.000	Max. :5.000	Max. :5.000	Max. :5.000
## C2	C3	C4	C5
## Min. :1.000	Min. :1.000	Min. :1.000	Min. :1.000
## 1st Qu.:3.000	1st Qu.:3.000	1st Qu.:3.000	1st Qu.:2.000
## Median :4.000	Median :4.000	Median :4.000	Median :3.000
## Mean :3.686	Mean :3.427	Mean :3.387	Mean :2.952
## 3rd Qu.:4.000	3rd Qu.:4.000	3rd Qu.:4.000	3rd Qu.:4.000
## Max. :5.000	Max. :5.000	Max. :5.000	Max. :5.000
## C6	C7	C8	C9
## Min. :1.000	Min. :1.000	Min. :1.000	Min. :1.000
## 1st Qu.:2.000	1st Qu.:2.000	1st Qu.:1.000	1st Qu.:1.000
## Median :3.000	Median :3.000	Median :2.000	Median :2.000
## Mean :2.939	Mean :2.791	Mean :2.328	Mean :2.362
## 3rd Qu.:4.000	3rd Qu.:4.000	3rd Qu.:3.000	3rd Qu.:3.000
## Max. :5.000	Max. :5.000	Max. :5.000	Max. :5.000
## C10	D1	D2	D3
## Min. :1.000	Min. :1.000	Min. :1.000	Min. :1.000
## 1st Qu.:2.000	1st Qu.:3.000	1st Qu.:3.000	1st Qu.:3.000
## Median :2.000	Median :4.000	Median :3.000	Median :4.000
## Mean :2.549	Mean :3.484	Mean :3.343	Mean :3.593

##	3rd Qu.:3.000	3rd Qu.:4.000	3rd Qu.:4.000	3rd Qu.:4.000	
##	Max. :5.000	Max. :5.000	Max. :5.000	Max. :5.000	
##	D4	D5	D6	D7	
##	Min. :1.000	Min. :1.000	Min. :1.000	Min. :1.000	
##	1st Qu.:3.000	1st Qu.:3.000	1st Qu.:3.000	1st Qu.:2.000	
##	Median :4.000	Median :4.000	Median :4.000	Median :3.000	
##	Mean :3.525	Mean :3.663	Mean :3.609	Mean :2.677	
##	3rd Qu.:4.000	3rd Qu.:4.000	3rd Qu.:4.000	3rd Qu.:3.000	
##	Max. :5.000	Max. :5.000	Max. :5.000	Max. :5.000	
##	D8	D9	D10	E1	
##	Min. :1.000	Min. :1.000	Min. :1.000	Min. :1.000	
##	1st Qu.:2.000	1st Qu.:2.000	1st Qu.:2.000	1st Qu.:2.000	
##	Median :2.000	Median :3.000	Median :2.000	Median :2.574	
##	Mean :2.124	Mean :2.885	Mean :2.495	Mean :2.574	
##	3rd Qu.:2.124	3rd Qu.:4.000	3rd Qu.:3.000	3rd Qu.:3.000	
##	Max. :5.000	Max. :5.000	Max. :5.000	Max. :5.000	
##	E2	E3	E4	E5	E6
##	Min. :1.000	Min. :1.000	Min. :1.000	Min. :1.000	Min. :1.000
##	1st Qu.:2.000	1st Qu.:3.000	1st Qu.:2.000	1st Qu.:4.000	1st Qu.:2.000
##	Median :3.000	Median :4.000	Median :3.000	Median :4.000	Median :3.000
##	Mean :2.918	Mean :3.842	Mean :2.765	Mean :3.929	Mean :2.83
##	3rd Qu.:4.000	3rd Qu.:5.000	3rd Qu.:4.000	3rd Qu.:4.000	3rd Qu.:4.000
##	Max. :5.000	Max. :5.000	Max. :5.000	Max. :5.000	Max. :5.000
##	E7	E8	E9	E10	
##	Min. :1.000	Min. :1.000	Min. :1.000	Min. :1.000	
##	1st Qu.:1.000	1st Qu.:2.000	1st Qu.:1.000	1st Qu.:1.000	
##	Median :2.000	Median :3.000	Median :2.000	Median :2.000	
##	Mean :2.207	Mean :3.141	Mean :2.105	Mean :2.352	
##	3rd Qu.:3.000	3rd Qu.:4.000	3rd Qu.:3.000	3rd Qu.:3.000	
##	Max. :5.000	Max. :5.000	Max. :5.000	Max. :5.000	
##	F1	F2	F3	F4	
##	Min. :1.000	Min. :1.000	Min. :1.000	Min. :1.000	
##	1st Qu.:2.000	1st Qu.:3.000	1st Qu.:1.000	1st Qu.:3.000	
##	Median :3.000	Median :4.000	Median :2.000	Median :4.000	
##	Mean :3.104	Mean :3.675	Mean :2.574	Mean :3.647	
##	3rd Qu.:4.000	3rd Qu.:4.000	3rd Qu.:4.000	3rd Qu.:4.000	
##	Max. :5.000	Max. :5.000	Max. :5.000	Max. :5.000	
##	F5	F6	F7	F8	F9
##	Min. :1.000	Min. :1.00	Min. :1.000	Min. :1.000	Min. :1.000
##	1st Qu.:2.000	1st Qu.:2.00	1st Qu.:2.000	1st Qu.:3.000	1st Qu.:2.000
##	Median :4.000	Median :2.00	Median :3.000	Median :4.000	Median :2.000

##	Mean	:3.381	Mean	:2.59	Mean	:2.913	Mean	:3.559	Mean	:2.598
##	3rd Qu.:	4.000	3rd Qu.:	3.00	3rd Qu.:	4.000	3rd Qu.:	4.000	3rd Qu.:	3.000
##	Max.	:5.000	Max.	:5.00	Max.	:5.000	Max.	:5.000	Max.	:5.000
##	F10		G1		G2		G3		G4	
##	Min.	:1.000	Min.	:1.000	Min.	:1.00	Min.	:1.000	Min.	:1.000
##	1st Qu.:	2.000	1st Qu.:	3.000	1st Qu.:	2.00	1st Qu.:	2.000	1st Qu.:	2.000
##	Median	:3.000	Median	:4.000	Median	:3.00	Median	:3.000	Median	:4.000
##	Mean	:3.267	Mean	:3.474	Mean	:2.96	Mean	:3.195	Mean	:3.357
##	3rd Qu.:	4.000	3rd Qu.:	4.000	3rd Qu.:	4.00	3rd Qu.:	4.000	3rd Qu.:	4.000
##	Max.	:5.000	Max.	:5.000	Max.	:5.00	Max.	:5.000	Max.	:5.000
##	G5		G6		G7		G8			
##	Min.	:1.000	Min.	:1.000	Min.	:1.000	Min.	:1.000		
##	1st Qu.:	3.000	1st Qu.:	2.000	1st Qu.:	2.000	1st Qu.:	2.000		
##	Median	:4.000	Median	:3.000	Median	:3.000	Median	:2.000		
##	Mean	:3.425	Mean	:2.858	Mean	:2.801	Mean	:2.412		
##	3rd Qu.:	4.000	3rd Qu.:	4.000	3rd Qu.:	4.000	3rd Qu.:	3.000		
##	Max.	:5.000	Max.	:5.000	Max.	:5.000	Max.	:5.000		
##	G9		G10		H1		H2			
##	Min.	:1.000	Min.	:1.000	Min.	:1.000	Min.	:1.000		
##	1st Qu.:	2.000	1st Qu.:	2.000	1st Qu.:	3.000	1st Qu.:	4.000		
##	Median	:4.000	Median	:3.000	Median	:4.000	Median	:4.000		
##	Mean	:3.469	Mean	:3.117	Mean	:3.919	Mean	:3.989		
##	3rd Qu.:	4.000	3rd Qu.:	4.000	3rd Qu.:	5.000	3rd Qu.:	5.000		
##	Max.	:5.000	Max.	:5.000	Max.	:5.000	Max.	:5.000		
##	H3		H4		H5		H6		H7	
##	Min.	:1.000	Min.	:1.000	Min.	:1.000	Min.	:1.000	Min.	:1.000
##	1st Qu.:	3.000	1st Qu.:	1.000	1st Qu.:	2.000	1st Qu.:	3.000	1st Qu.:	2.000
##	Median	:4.000	Median	:2.000	Median	:3.141	Median	:4.000	Median	:2.000
##	Mean	:3.578	Mean	:2.283	Mean	:3.141	Mean	:3.678	Mean	:2.49
##	3rd Qu.:	5.000	3rd Qu.:	3.000	3rd Qu.:	4.000	3rd Qu.:	5.000	3rd Qu.:	3.000
##	Max.	:5.000	Max.	:5.000	Max.	:5.000	Max.	:5.000	Max.	:5.000
##	H8		H9		H10		I1			
##	Min.	:1.000	Min.	:1.000	Min.	:1.000	Min.	:1.000		
##	1st Qu.:	2.000	1st Qu.:	1.000	1st Qu.:	2.000	1st Qu.:	2.000		
##	Median	:2.000	Median	:2.000	Median	:2.000	Median	:2.000		

##	Mean	:2.561	Mean	:1.979	Mean	:2.535	Mean	:2.722		
##	3rd Qu.:	4.000	3rd Qu.:	2.000	3rd Qu.:	3.000	3rd Qu.:	4.000		
##	Max.	:5.000	Max.	:5.000	Max.	:5.000	Max.	:5.000		
##	I2		I3		I4		I5			
##	Min.	:1.000	Min.	:1.000	Min.	:1.000	Min.	:1.000		
##	1st Qu.:	3.000	1st Qu.:	2.000	1st Qu.:	2.000	1st Qu.:	3.000		
##	Median	:4.000	Median	:3.000	Median	:3.000	Median	:4.000		
##	Mean	:3.377	Mean	:3.205	Mean	:2.882	Mean	:3.509		
##	3rd Qu.:	4.000	3rd Qu.:	4.000	3rd Qu.:	4.000	3rd Qu.:	4.000		
##	Max.	:5.000	Max.	:5.000	Max.	:5.000	Max.	:5.000		
##	I6		I7		I8		I9			
##	Min.	:1.000	Min.	:1.000	Min.	:1.000	Min.	:1.000		
##	1st Qu.:	1.000	1st Qu.:	2.000	1st Qu.:	3.000	1st Qu.:	3.000		
##	Median	:2.000	Median	:3.000	Median	:4.000	Median	:4.000		
##	Mean	:2.121	Mean	:3.099	Mean	:3.295	Mean	:3.427		
##	3rd Qu.:	3.000	3rd Qu.:	4.000	3rd Qu.:	4.000	3rd Qu.:	4.000		
##	Max.	:5.000	Max.	:5.000	Max.	:5.000	Max.	:5.000		
##	I10		J1		J2		J3			
##	Min.	:1.000	Min.	:1.000	Min.	:1.000	Min.	:1.000		
##	1st Qu.:	3.000	1st Qu.:	3.000	1st Qu.:	3.000	1st Qu.:	3.000		
##	Median	:3.000	Median	:4.000	Median	:4.000	Median	:4.000		
##	Mean	:3.278	Mean	:3.603	Mean	:3.865	Mean	:3.584		
##	3rd Qu.:	4.000	3rd Qu.:	4.000	3rd Qu.:	5.000	3rd Qu.:	4.000		
##	Max.	:5.000	Max.	:5.000	Max.	:5.000	Max.	:5.000		
##	J4		J5		J6		J7		J8	
##	Min.	:1.000	Min.	:1.00	Min.	:1.000	Min.	:1.000	Min.	:1.000
##	1st Qu.:	3.000	1st Qu.:	3.00	1st Qu.:	2.000	1st Qu.:	3.000	1st Qu.:	2.000
##	Median	:4.000	Median	:3.00	Median	:3.000	Median	:4.000	Median	:3.000
##	Mean	:3.897	Mean	:3.37	Mean	:2.891	Mean	:3.383	Mean	:2.874
##	3rd Qu.:	5.000	3rd Qu.:	4.00	3rd Qu.:	4.000	3rd Qu.:	4.000	3rd Qu.:	4.000
##	Max.	:5.000	Max.	:5.00	Max.	:5.000	Max.	:5.000	Max.	:5.000
##	J9		J10		K1		K2		K3	
##	Min.	:1.000	Min.	:1.000	Min.	:1.000	Min.	:1.000	Min.	:1.000
##	1st Qu.:	2.000	1st Qu.:	2.000	1st Qu.:	2.000	1st Qu.:	2.000	1st Qu.:	2.000
##	Median	:2.000	Median	:2.000	Median	:3.000	Median	:3.000	Median	:3.000
##	Mean	:2.409	Mean	:2.419	Mean	:3.091	Mean	:3.027	Mean	:2.74
##	3rd Qu.:	3.000	3rd Qu.:	3.000	3rd Qu.:	4.000	3rd Qu.:	4.000	3rd Qu.:	4.000
##	Max.	:5.000	Max.	:5.000	Max.	:5.000	Max.	:5.000	Max.	:5.000

##	K4	K5	K6	K7	K8
##	Min. :1.00	Min. :1.000	Min. :1.000	Min. :1.000	Min. :1.000
##	1st Qu.:2.00	1st Qu.:2.000	1st Qu.:2.000	1st Qu.:2.000	1st Qu.:2.000
##	Median :4.00	Median :3.000	Median :3.205	Median :3.000	Median :3.000
##	Mean :3.31	Mean :3.196	Mean :3.205	Mean :3.078	Mean :2.843
##	3rd Qu.:4.00	3rd Qu.:4.000	3rd Qu.:4.000	3rd Qu.:4.000	3rd Qu.:4.000
##	Max. :5.00	Max. :5.000	Max. :5.000	Max. :5.000	Max. :5.000
##	K9	K10	L1	L2	L3
##	Min. :1.000	Min. :1.000	Min. :1.00	Min. :1.000	Min. :1.000
##	1st Qu.:2.000	1st Qu.:3.000	1st Qu.:2.00	1st Qu.:2.000	1st Qu.:2.000
##	Median :3.000	Median :4.000	Median :4.00	Median :2.000	Median :3.000
##	Mean :3.122	Mean :3.554	Mean :3.39	Mean :2.672	Mean :3.155
##	3rd Qu.:4.000	3rd Qu.:4.000	3rd Qu.:4.00	3rd Qu.:4.000	3rd Qu.:4.000
##	Max. :5.000	Max. :5.000	Max. :5.00	Max. :5.000	Max. :5.000
##	L4	L5	L6	L7	L8
##	Min. :1.00	Min. :1.000	Min. :1.00	Min. :1.000	Min. :1.000
##	1st Qu.:3.00	1st Qu.:3.000	1st Qu.:2.00	1st Qu.:2.000	1st Qu.:2.000
##	Median :4.00	Median :4.000	Median :3.21	Median :3.000	Median :2.000
##	Mean :3.74	Mean :3.557	Mean :3.21	Mean :3.029	Mean :2.698
##	3rd Qu.:4.00	3rd Qu.:4.000	3rd Qu.:4.00	3rd Qu.:4.000	3rd Qu.:4.000
##	Max. :5.00	Max. :5.000	Max. :5.00	Max. :5.000	Max. :5.000
##	L9	L10	M1	M2	
##	Min. :1.000	Min. :1.000	Min. :1.000	Min. :1.000	
##	1st Qu.:2.000	1st Qu.:3.000	1st Qu.:3.000	1st Qu.:3.000	
##	Median :3.000	Median :4.000	Median :4.000	Median :4.000	
##	Mean :2.931	Mean :3.424	Mean :3.861	Mean :3.897	
##	3rd Qu.:4.000	3rd Qu.:4.000	3rd Qu.:5.000	3rd Qu.:5.000	
##	Max. :5.000	Max. :5.000	Max. :5.000	Max. :5.000	
##	M3	M4	M5	M6	M7
##	Min. :1.000	Min. :1.000	Min. :1.000	Min. :1.000	Min. :1.000
##	1st Qu.:4.000	1st Qu.:3.000	1st Qu.:3.000	1st Qu.:1.000	1st Qu.:1.000

.00					
## Median :4.000	Median :4.000	Median :4.000	Median :2.000	Median :2	
.00					
## Mean :4.264	Mean :3.528	Mean :3.457	Mean :2.187	Mean :2	
.16					
## 3rd Qu.:5.000	3rd Qu.:4.000	3rd Qu.:4.000	3rd Qu.:3.000	3rd Qu.:3	
.00					
## Max. :5.000	Max. :5.000	Max. :5.000	Max. :5.000	Max. :5	
.00					
## M8	M9	M10	N1	N2	
## Min. :1.000	Min. :1.000	Min. :1.000	Min. :1.00	Min. :1.	
000					
## 1st Qu.:1.000	1st Qu.:1.000	1st Qu.:2.000	1st Qu.:2.00	1st Qu.:3.	
000					
## Median :2.000	Median :2.000	Median :2.000	Median :3.00	Median :4.	
000					
## Mean :2.226	Mean :2.173	Mean :2.371	Mean :2.87	Mean :3.	
573					
## 3rd Qu.:3.000	3rd Qu.:3.000	3rd Qu.:3.000	3rd Qu.:4.00	3rd Qu.:4.	
000					
## Max. :5.000	Max. :5.000	Max. :5.000	Max. :5.00	Max. :5.	
000					
## N3	N4	N5	N6		
## Min. :1.000	Min. :1.000	Min. :1.000	Min. :1.000		
## 1st Qu.:4.000	1st Qu.:3.000	1st Qu.:3.000	1st Qu.:3.000		
## Median :4.000	Median :3.363	Median :4.000	Median :4.000		
## Mean :3.934	Mean :3.363	Mean :3.704	Mean :3.726		
## 3rd Qu.:5.000	3rd Qu.:4.000	3rd Qu.:5.000	3rd Qu.:4.000		
## Max. :5.000	Max. :5.000	Max. :5.000	Max. :5.000		
## N7	N8	N9	N10	O1	
## Min. :1.000	Min. :1.000	Min. :1.0	Min. :1.000	Min. :1.0	
00					
## 1st Qu.:4.000	1st Qu.:3.000	1st Qu.:3.0	1st Qu.:2.000	1st Qu.:3.0	
00					
## Median :4.000	Median :4.000	Median :4.0	Median :2.000	Median :4.0	
00					
## Mean :4.165	Mean :3.647	Mean :3.5	Mean :2.639	Mean :3.6	
25					
## 3rd Qu.:5.000	3rd Qu.:4.000	3rd Qu.:4.0	3rd Qu.:4.000	3rd Qu.:4.0	
00					
## Max. :5.000	Max. :5.000	Max. :5.0	Max. :5.000	Max. :5.0	
00					
## O2	O3	O4	O5		
## Min. :1.000	Min. :1.000	Min. :1.000	Min. :1.000		
## 1st Qu.:2.000	1st Qu.:3.000	1st Qu.:3.000	1st Qu.:3.000		
## Median :3.000	Median :4.000	Median :4.000	Median :4.000		
## Mean :2.751	Mean :3.642	Mean :3.488	Mean :3.619		
## 3rd Qu.:4.000	3rd Qu.:4.000	3rd Qu.:4.000	3rd Qu.:4.000		
## Max. :5.000	Max. :5.000	Max. :5.000	Max. :5.000		
## O6	O7	O8	O9		

```
## Min. :1.000 Min. :1.000 Min. :1.000 Min. :1.000
## 1st Qu.:2.000 1st Qu.:2.000 1st Qu.:2.000 1st Qu.:2.000
## Median :3.000 Median :3.000 Median :3.199 Median :3.000
## Mean :2.994 Mean :2.884 Mean :3.199 Mean :3.033
## 3rd Qu.:4.000 3rd Qu.:4.000 3rd Qu.:4.000 3rd Qu.:4.000
## Max. :5.000 Max. :5.000 Max. :5.000 Max. :5.000
##      O10      P1      P2      P3      P4
## Min. :1.000 Min. :1.000 Min. :1.0 Min. :1.000 Min. :1.0
## 1st Qu.:3.000 1st Qu.:2.000 1st Qu.:2.0 1st Qu.:2.000 1st Qu.:2.0
## Median :4.000 Median :3.000 Median :2.0 Median :3.000 Median :3.0
## Mean :3.489 Mean :2.979 Mean :2.7 Mean :3.011 Mean :3.1
## 3rd Qu.:4.000 3rd Qu.:4.000 3rd Qu.:4.0 3rd Qu.:4.000 3rd Qu.:4.0
## Max. :5.000 Max. :5.000 Max. :5.0 Max. :5.000 Max. :5.0
##      P5      P6      P7      P8
## Min. :1.000 Min. :1.000 Min. :1.000 Min. :1.000
## 1st Qu.:2.000 1st Qu.:2.000 1st Qu.:2.000 1st Qu.:2.000
## Median :3.000 Median :3.000 Median :3.000 Median :3.000
## Mean :2.637 Mean :3.135 Mean :2.828 Mean :2.961
## 3rd Qu.:3.000 3rd Qu.:4.000 3rd Qu.:4.000 3rd Qu.:4.000
## Max. :5.000 Max. :5.000 Max. :5.000 Max. :5.000
##      P9      P10
## Min. :1.000 Min. :1.000
## 1st Qu.:3.000 1st Qu.:3.000
## Median :4.000 Median :4.000
## Mean :3.672 Mean :3.504
## 3rd Qu.:4.000 3rd Qu.:4.000
## Max. :5.000 Max. :5.000
```

Function to calculate unique counts for each column

```
unique_counts <- sapply(data, function(x) length(unique(na.omit(x))))
```

Print the unique counts

```
print(unique_counts)
```

```
## A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 B1 B2 B3 B4 B5 B6 B7 B8 B
## 9 B10
## 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## 5 5
## B11 B12 B13 C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 D1 D2 D3 D4 D5 D
## 6 D7
## 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
```

```

5      5
## D8 D9 D10 E1 E2 E3 E4 E5 E6 E7 E8 E9 E10 F1 F2 F3 F4 F5 F
6 F7
## 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
5      5
## F8 F9 F10 G1 G2 G3 G4 G5 G6 G7 G8 G9 G10 H1 H2 H3 H4 H5 H
6 H7
## 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
5      5
## H8 H9 H10 I1 I2 I3 I4 I5 I6 I7 I8 I9 I10 J1 J2 J3 J4 J5 J
6 J7
## 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
5      5
## J8 J9 J10 K1 K2 K3 K4 K5 K6 K7 K8 K9 K10 L1 L2 L3 L4 L5 L
6 L7
## 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
5      5
## L8 L9 L10 M1 M2 M3 M4 M5 M6 M7 M8 M9 M10 N1 N2 N3 N4 N5 N
6 N7
## 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
5      5
## N8 N9 N10 O1 O2 O3 O4 O5 O6 O7 O8 O9 O10 P1 P2 P3 P4 P5 P
6 P7
## 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
5      5
## P8 P9 P10
## 5 5 5

```

Count the number of rows after imputation

```
n_rows_after <- nrow(imputed_data)
```

Calculate the percentage of rows retained

```
percentage_retained <- (n_rows_after / n_rows_before) * 100
```

Print the percentage

```
print(paste("Percentage of rows retained after NA imputation:", percentage_re
tained, "%"))
```

```
## [1] "Percentage of rows retained after NA imputation: 100 %"
```

```
library(corrplot) # For correlation plots
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
```

```
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':  
##  
## lift
```

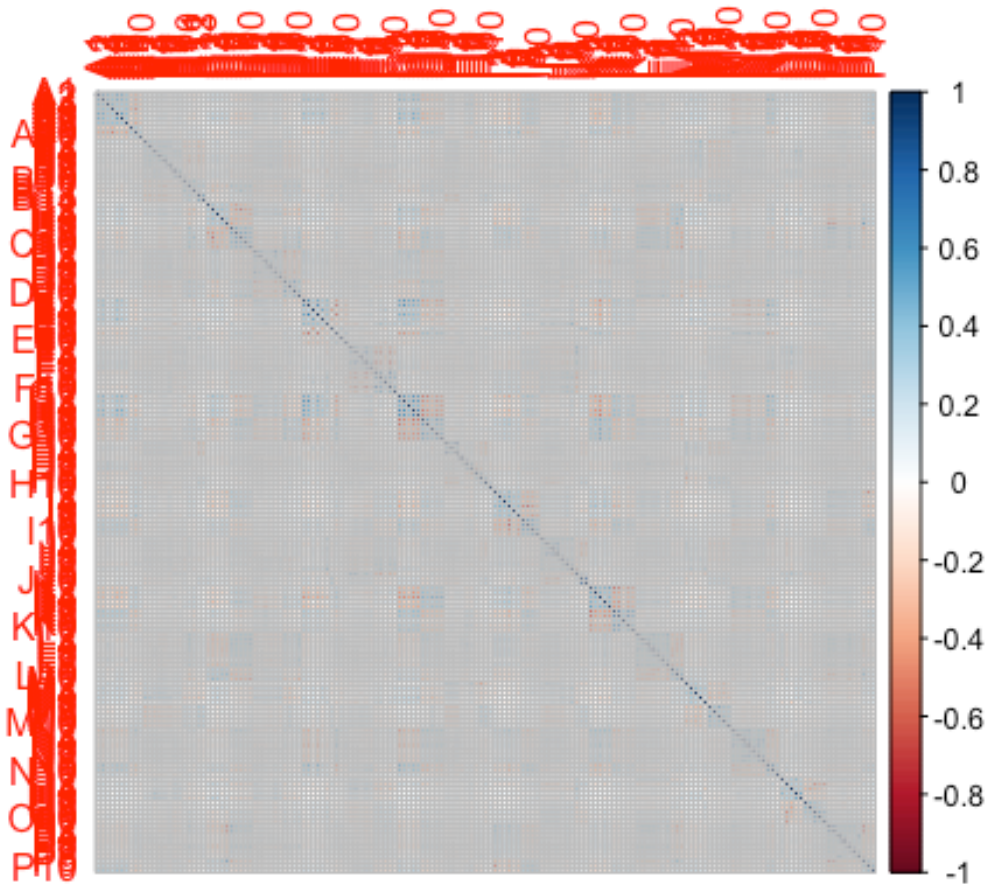
Calculate the correlation matrix

```
cor_matrix <- cor(imputed_data, use = "pairwise.complete.obs") # Handles missing values by using complete pairs  
  
# Set the threshold for high correlation  
threshold <- 0.75  
  
# Find highly correlated pairs (absolute value)  
highly_correlated_pairs <- which(abs(cor_matrix) > threshold, arr.ind = TRUE)  
  
# Create a data frame to store the variable pairs and their correlation values  
high_correlations <- data.frame(  
  Variable1 = rownames(cor_matrix)[highly_correlated_pairs[, 1]],  
  Variable2 = colnames(cor_matrix)[highly_correlated_pairs[, 2]],  
  Correlation = cor_matrix[highly_correlated_pairs],  
  stringsAsFactors = FALSE  
)  
  
high_correlations <- subset(high_correlations, Variable1 != Variable2)  
high_correlations <- high_correlations[!duplicated(t(apply(high_correlations, 1, sort))), ]  
  
# Print the highly correlated pairs  
print(high_correlations)  
  
##      Variable1 Variable2 Correlation  
## 75          H3         H1  0.8200898  
## 105         J10         J9  0.7615790  
  
high_correlations <- data.frame(  
  Variable1 = c("H3", "J10"),  
  Variable2 = c("H1", "J9"),  
  Correlation = c(0.8200898, 0.7615790)  
)  
  
variables_to_remove <- high_correlations$Variable2  
variables_to_remove <- unique(variables_to_remove)  
  
# Remove the columns from imputed_data  
imputed_data <- imputed_data[, !(names(imputed_data) %in% variables_to_remove)]  
print(dim(imputed_data))
```

```
## [1] 49159 161
```

Plot the correlation matrix

```
corrplot(cor_matrix, method = "circle")
```



PCA_PLOTS

Load necessary libraries.

```
library(factoextra)
library(FactoMineR)
library(ggplot2)

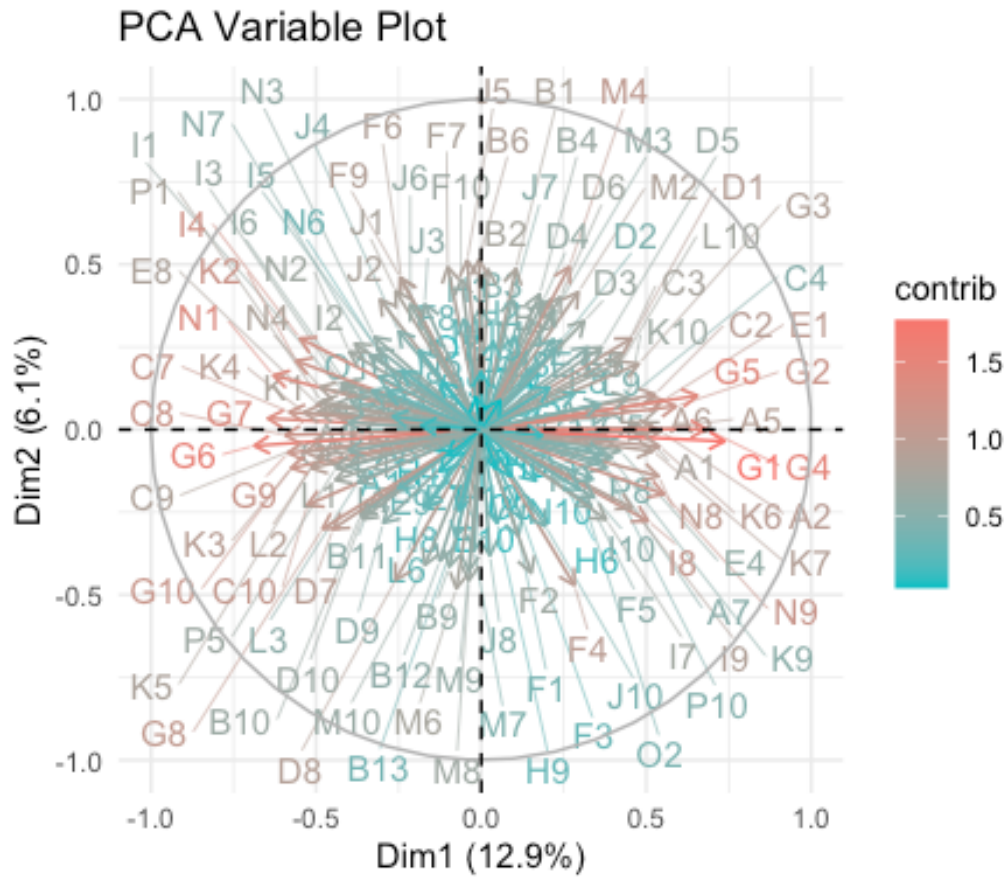
# Perform PCA
pca_result <- prcomp(imputed_data, center = TRUE, scale. = TRUE)

# Visualize the PCA results using factoextra
pca_plot <- function(pcaData) {
  fviz_pca_var(pcaData, col.var = "contrib",
    gradient.cols = c("#00BFC4", "#F8766D"),
    repel = TRUE) +
  theme_minimal() +
```

```
  labs(title = "PCA Variable Plot")
}
```

Call the function with your PCA results

```
pca_plot(pca_result)
```



The image depicts a PCA variable plot, showing how each variable contributes to the first two principal components of a dataset with variables labeled as B1, C3, G4, etc.

The length and direction of the vectors indicate the strength and combination of variables in the components, with the color gradient denoting the contribution level.

The first two components explain 19.5% of the total variance, suggesting further dimensions may be relevant for a comprehensive analysis.

PCA Scatter Plot Function

```
PCA_Scatter_Plot <- function(pca_result) {
  pca_data <- data.frame(PC1 = pca_result$x[, "PC1"], PC2 = pca_result$x[, "PC2"])

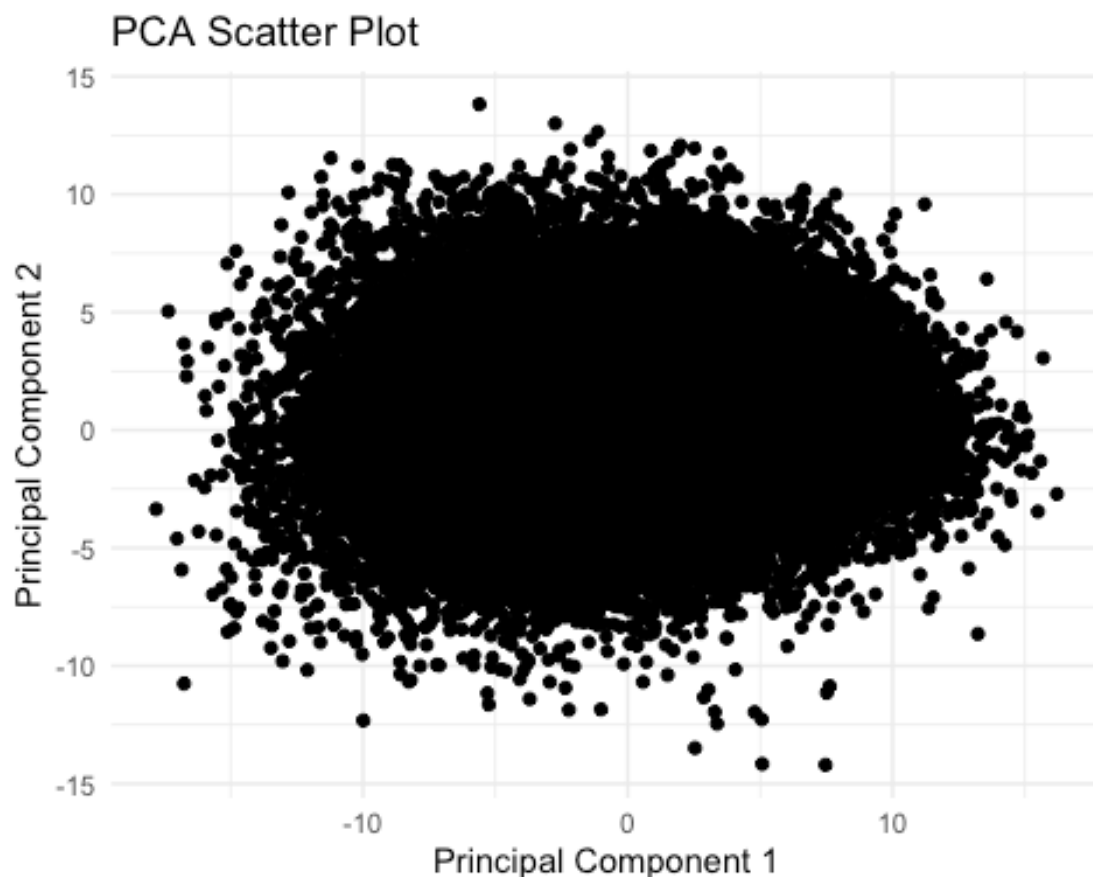
  ggplot(pca_data, aes(x = PC1, y = PC2)) +
```



```
geom_point() +  
theme_minimal() +  
labs(title = "PCA Scatter Plot",  
      x = "Principal Component 1",  
      y = "Principal Component 2")  
}
```

Plot PCA Scatter Plot

```
PCA_Scatter_Plot(pca_result)
```



The scatter plot displays data points in the space defined by the first two principal components, suggesting a PCA application.

The points are densely clustered around the center without clear separation, indicating no distinct groups or outliers in this two-dimensional PCA view.

The plot also implies that the first two principal components may not explain a significant portion of the variance, or the data might not be well-differentiated in the space of these components.

PCA Biplot Function

```

PCA_Biplot <- function(pca_result) {
  scores <- as.data.frame(pca_result$x)
  loadings <- pca_result$rotation

  # First two principal components scores
  scores_df <- data.frame(PC1 = scores[, "PC1"], PC2 = scores[, "PC2"])

  # First two principal components loadings
  loadings_df <- data.frame(Variable = rownames(loadings), PC1 = loadings[, "PC1"], PC2 = loadings[, "PC2"])

  # Create the base scatter plot for scores
  p <- ggplot() +
    geom_point(data = scores_df, aes(x = PC1, y = PC2), color = "pink") +
    theme_minimal() +
    labs(title = "PCA Biplot",
         x = "Principal Component 1",
         y = "Principal Component 2")

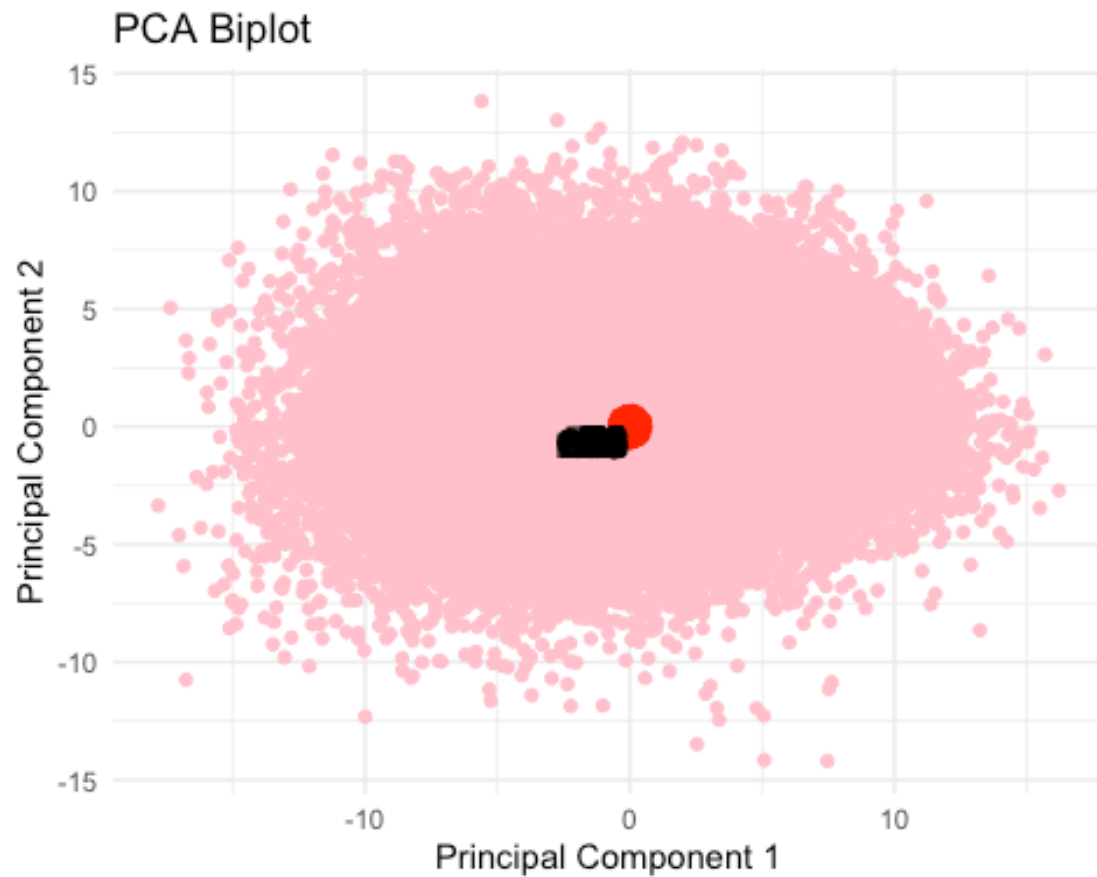
  # Add arrows for loadings
  p <- p + geom_segment(data = loadings_df, aes(x = 0, y = 0, xend = PC1, yend = PC2),
                        arrow = arrow(type = "closed", length = unit(0.1, "inches")),
                        color = "red") +
    geom_text(data = loadings_df, aes(x = PC1, y = PC2, label = Variable), hjust = 1.1, vjust = 1.1)

  return(p)
}

```

Plot PCA Biplot

```
PCA_Biplot(pca_result)
```

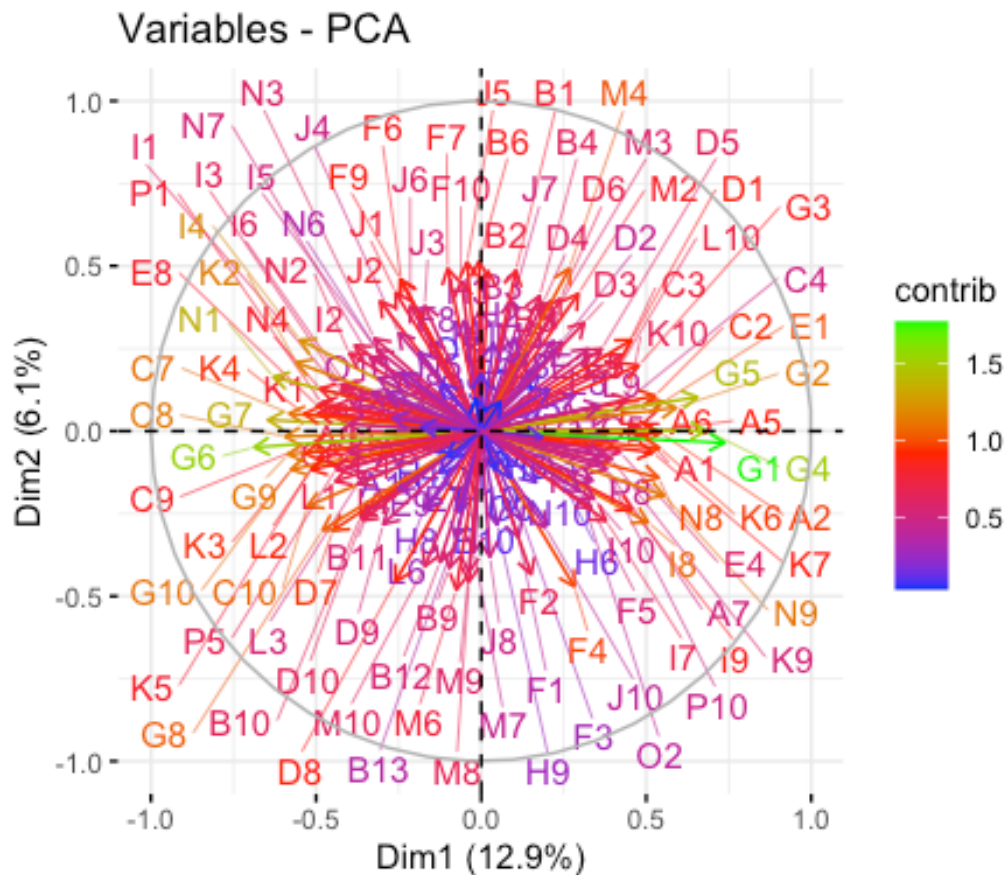


Create a biplot of variable contributions

```
biplot <- fviz_pca_var(pca_result, col.var = "contrib", gradient.cols = c("blue", "red", "green"), repel = TRUE)
```

Print or plot the biplot

```
print(biplot)
```



The PCA biplot visualizes the loadings of variables on the first two principal components, with a wide spread indicating diverse contributions to the dataset's variance.

The color gradient, from red through blue, represents the magnitude of each variable's contribution.

Principal Components 1 and 2 together explain 19.5% of the variance, and the clustering of variables along the axes suggests potential correlations or underlying factors.

Kaiser-Meyer-Olkin (KMO) Test for Sampling Adequacy

```
library(psych) # This package contains the KMO test
```

```
##
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':
##
##   %+%, alpha

## The following object is masked from 'package:car':
##
##   logit
```

```
kmo_result <- KMO(imputed_data)
```

Print the KMO statistic

```
print(kmo_result$KMO)
```

```
## NULL
```

Bartlett's Test of Sphericity

```
bartlett_result <- cortest.bartlett(imputed_data)
```

```
## R was not square, finding R from data
```

Print the test results

```
print(bartlett_result)
```

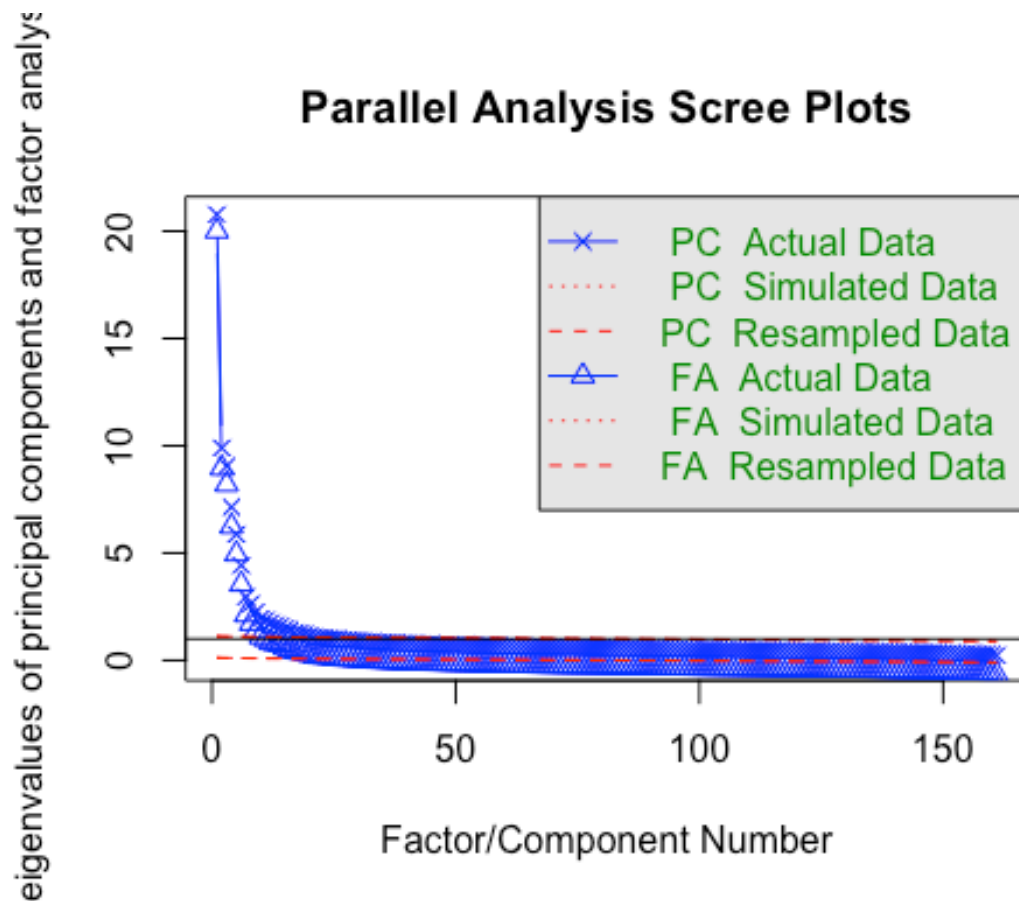
```
## $chisq
## [1] 3378446
##
## $p.value
## [1] 0
##
## $df
## [1] 12880
```

Perform Parallel Analysis

```
library(nFactors) # This package provides functions for parallel analysis
```

```
##
## Attaching package: 'nFactors'
## The following object is masked from 'package:lattice':
##
## parallel
```

```
set.seed(123) # Setting seed for reproducibility
parallel_result <- fa.parallel(imputed_data, fa = "both", n.iter = 100)
```



```
## Parallel analysis suggests that the number of factors = 26 and the number of components = 21
```

PRINCIPAL COMPONENT ANALYSIS

QUESTION 1

Performing PCA on cleaned data

```
pca_result <- prcomp(imputed_data, center = TRUE, scale. = TRUE)
```

Calculate explained variance for each principal component

```
explained_variance <- pca_result$sdev^2 / sum(pca_result$sdev^2)
```

Calculate cumulative explained variance

```
cumulative_variance <- cumsum(explained_variance)
```

```
components_needed <- which(cumulative_variance >= 0.80)[1]
```

Print the number of components needed to explain 80% of the variance

```
print(paste("Number of components needed to explain 80% of variance:", components_needed))
```

```
## [1] "Number of components needed to explain 80% of variance: 85"
```

Number of components needed to explain 80% of variance is 85.

In Principal Component Analysis (PCA), the number of components required to explain 80% of the variance is determined by cumulatively summing the variance explained by each component.

The process involves squaring the singular values from the PCA to get the variance explained by each component, then calculating the cumulative sum. Once this cumulative sum reaches or exceeds 80%, the corresponding number of components is identified.

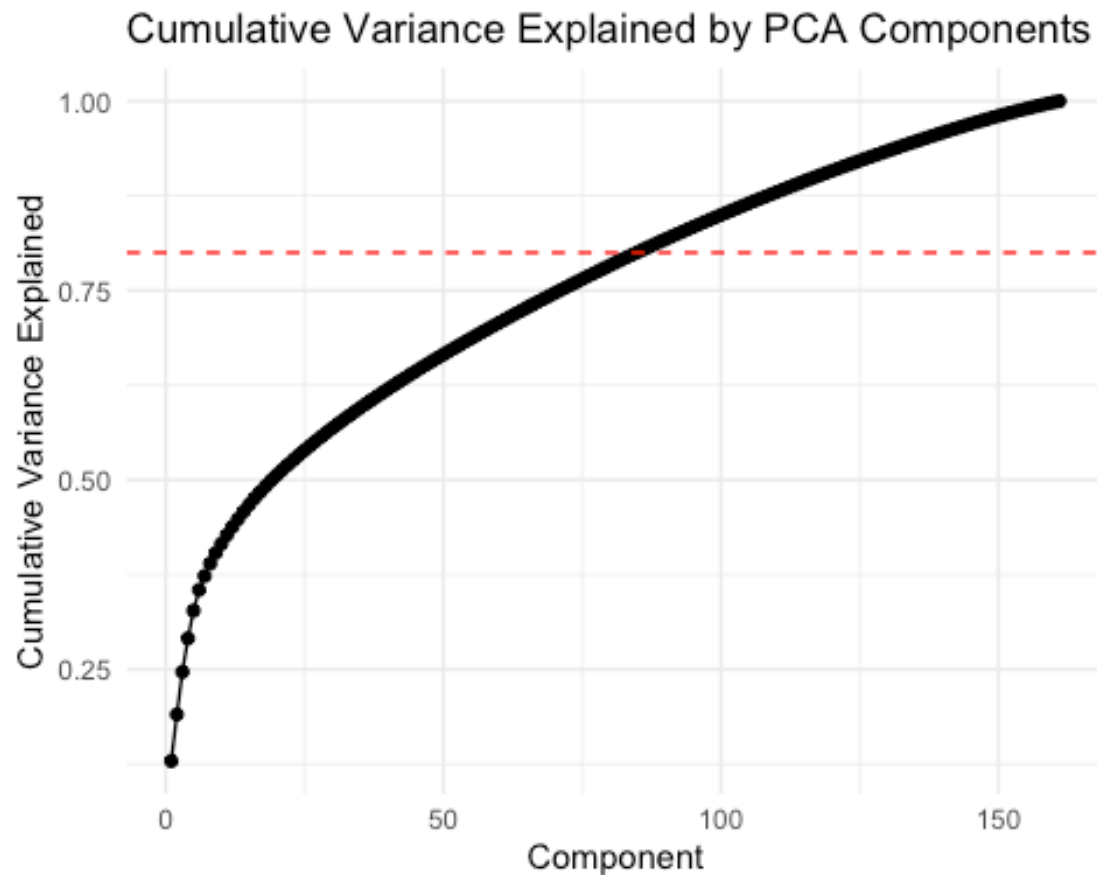
The R function `which(cumulative_variance >= 0.80)[1]` returns the smallest number of components needed to surpass this threshold, which in this case is 85 components.

Create a data frame for plotting

```
variance_df <- data.frame(Component = 1:length(explained_variance),  
                           CumulativeVariance = cumulative_variance)
```

Plot

```
ggplot(variance_df, aes(x = Component, y = CumulativeVariance)) +  
  geom_line() +  
  geom_point() +  
  geom_hline(yintercept = 0.80, linetype = "dashed", color = "red") +  
  theme_minimal() +  
  ggtitle("Cumulative Variance Explained by PCA Components") +  
  xlab("Component") +  
  ylab("Cumulative Variance Explained")
```



The scree plot shows the cumulative variance explained by each principal component in a PCA. The curve levels off after a certain point, which is often referred to as the 'elbow' or 'knee' of the plot, indicating a diminishing return on explained variance with additional components.

Based on the 'elbow' method, one might choose the number of components just before the curve flattens out, which appears to be around the 20th component, as indicated by where the slope of the line changes most abruptly before reaching the red dashed line that represents the 80% cumulative variance threshold.

QUESTION 2

Eigenvalue Method

```
eigenvalues <- pca_result$sdev^2 # Square of the singular values are the eigenvalues
num_components_eigenvalue_method <- sum(eigenvalues > 1)
num_components_eigenvalue_method

## [1] 24
```


Extract the variance explained by each principal component

```
var_explained <- pca_result$sdev^2 / sum(pca_result$sdev^2)
```

Load necessary libraries

```
library(factoextra)  
library(FactoMineR)
```

Perform PCA using the FactoMineR package

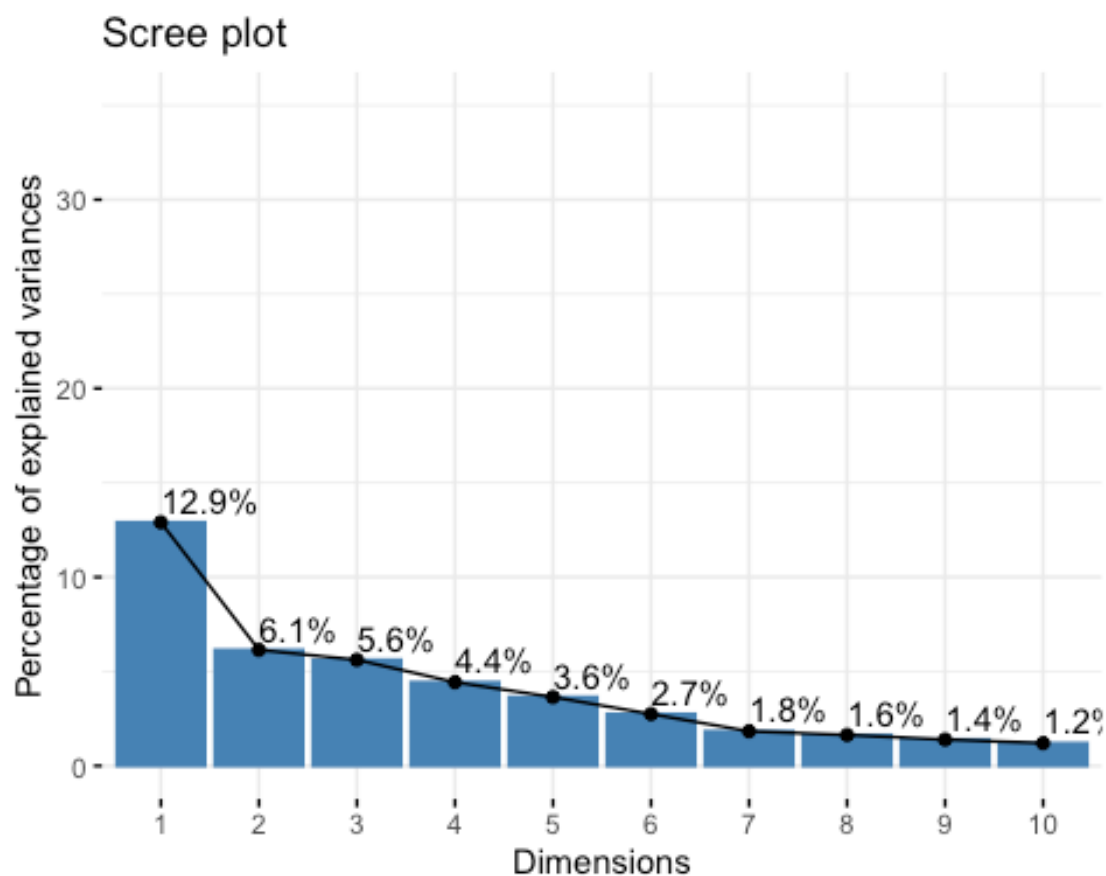
```
pca_res <- PCA(imputed_data, scale.unit = TRUE, graph = FALSE)
```

Create a scree plot using factoextra

```
screePlot <- fviz_screplot(pca_res, addlabels = TRUE, ylim = c(0,35))
```

Print the scree plot

```
print(screePlot)
```



Using the eigenvalue method, we will choose 24 components because each of these has an eigenvalue greater than 1.

Examining the scree plot, the “knee” or “elbow” appears after the first few components, typically where the slope of the line changes most dramatically; based on the visual, this might occur around the 4th or 5th component, suggesting that it will retain 4 or 5 components using the knee of the scree plot method.

QUESTION 3(I)

Load necessary libraries

```
library(tidyverse)
```

Extract the loadings (correlations between original variables and the component)

```
loadings <- as.data.frame(pca_result$rotation)
```

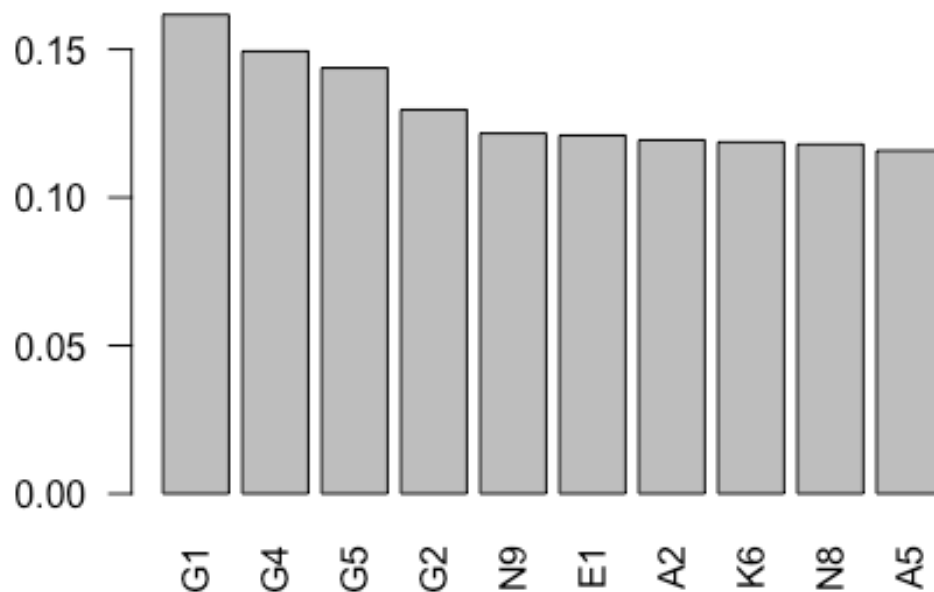
Visualize the top 10 variables of the first principal component

```
top_vars <- loadings %>%  
  arrange(desc(PC1)) %>%  
  head(10)
```

Plot

```
barplot(top_vars$PC1, names.arg = row.names(top_vars), las = 2, main = "Top 10 Variables of the First Principal Component")
```

Top 10 Variables of the First Principal Component



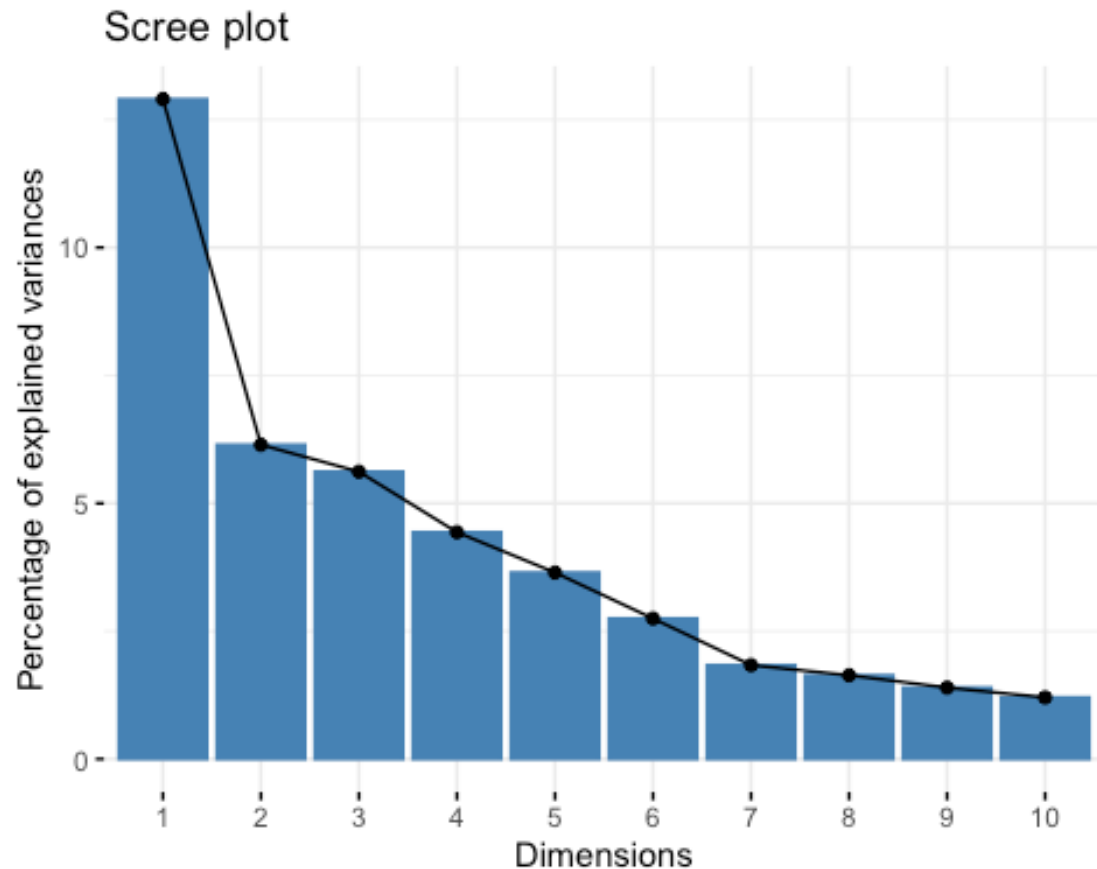
The bar chart displays the loadings of the top 10 variables on the first principal component from a PCA, where variables G1, G4, and G5 have the highest loadings, indicating they contribute most to this component.

The bars represent the magnitude of the loadings, with the height corresponding to the importance of each variable in defining the first principal component.

These variables are the most influential in terms of the variance captured by the first component in the PCA model.

Scree plot

```
scree_plot <- fviz_screepplot(pca_result)
print(scree_plot)
```



The scree plot illustrates the percentage of variance explained by each principal component in a PCA, with a sharp decline after the first component, indicating it captures the most variance.

The curve begins to flatten after the fourth dimension, suggesting that additional components beyond this point contribute incrementally less to explaining the data's variability.

This plot could guide the choice to retain around four components, considering the “elbow” method, where the slope of the plot changes most notably.

Eigenvalues

```
eigenvalues <- pca_result$sdev^2
print(eigenvalues)
```

```
## [1] 20.7731306 9.8925731 9.0444396 7.1388799 5.8698999 4.4227507
## [7] 2.9554945 2.6364142 2.2474933 1.9386841 1.8688879 1.7414111
## [13] 1.6382740 1.5795070 1.4775910 1.4018792 1.2942024 1.2619641
## [19] 1.2258592 1.1494430 1.0954344 1.0677632 1.0537659 1.0231974
## [25] 0.9968069 0.9823894 0.9585848 0.9442862 0.9288614 0.9032875
## [31] 0.8846997 0.8683616 0.8539531 0.8461026 0.8357765 0.8159867
## [37] 0.8124461 0.7988876 0.7899404 0.7815977 0.7787223 0.7636469
```

```
## [43] 0.7564680 0.7453756 0.7391524 0.7320288 0.7243391 0.7125656
## [49] 0.7062698 0.6978654 0.6919977 0.6884708 0.6827570 0.6800557
## [55] 0.6767289 0.6687179 0.6669473 0.6603401 0.6580724 0.6467347
## [61] 0.6461758 0.6406465 0.6401078 0.6382983 0.6323545 0.6257760
## [67] 0.6226126 0.6174816 0.6134684 0.6083693 0.6048399 0.6038194
## [73] 0.6003931 0.5978269 0.5947133 0.5939780 0.5879450 0.5838439
## [79] 0.5789869 0.5755368 0.5748794 0.5688912 0.5627146 0.5585490
## [85] 0.5560689 0.5545955 0.5481788 0.5459370 0.5399063 0.5385976
## [91] 0.5305320 0.5278281 0.5238327 0.5234587 0.5209685 0.5156157
## [97] 0.5135176 0.5093142 0.5056640 0.4991777 0.4982326 0.4961898
## [103] 0.4926544 0.4900202 0.4874171 0.4858253 0.4792671 0.4748814
## [109] 0.4704659 0.4694090 0.4679313 0.4656139 0.4620917 0.4601501
## [115] 0.4568218 0.4551246 0.4481044 0.4440381 0.4389041 0.4370581
## [121] 0.4362156 0.4318763 0.4311008 0.4268544 0.4246035 0.4211097
## [127] 0.4153961 0.4118971 0.4094555 0.4083039 0.4077431 0.4063957
## [133] 0.4026761 0.3957654 0.3893268 0.3888513 0.3881580 0.3856874
## [139] 0.3852083 0.3770966 0.3728339 0.3691096 0.3680131 0.3589251
## [145] 0.3564118 0.3488417 0.3453188 0.3395019 0.3366409 0.3336789
## [151] 0.3216916 0.3177482 0.2995647 0.2975002 0.2944308 0.2917608
## [157] 0.2792995 0.2742877 0.2722410 0.2641644 0.2305475
```

Determine number of components with eigenvalues > 1

```
num_components <- sum(eigenvalues > 1)
print(num_components)

## [1] 24
```

Based on the eigenvalues from a PCA, 24 components were chosen because they have eigenvalues greater than 1, which suggests that each of these components explains more variance than a single variable.

This criterion, known as the Kaiser criterion, is a common rule of thumb for selecting the number of components in PCA. The choice is made to retain only those components that contribute significantly to the variance in the dataset.

QUESTION 3(II)

Load necessary library

```
library(tidyverse)
```

Get the loadings (also called eigenvectors) of the first principal component

```
loadings <- pca_result$rotation[, 1]
```

Print the loadings of the first component

```
print(loadings)
```

```
##           A1           A2           A3           A4           A5
##  1.076968e-01  1.193647e-01  5.567373e-02  7.933802e-02  1.157858e-01
##           A6           A7           A8           A9           A10
##  1.151016e-01  8.353805e-02 -5.479441e-02 -9.912975e-02 -5.528656e-02
##           B1           B2           B3           B4           B5
##  2.310018e-02  4.587063e-03 -3.512017e-04  3.893428e-02  4.601346e-02
##           B6           B7           B8           B9           B10
## -1.419484e-03  1.264079e-02  3.838638e-02 -2.121543e-02 -8.020471e-02
##           B11          B12          B13           C1           C2
## -7.432825e-02 -2.752066e-02 -1.692427e-02  7.389672e-02  1.155199e-01
##           C3           C4           C5           C6           C7
##  9.433505e-02  8.512024e-02  8.412130e-02 -9.967379e-02 -1.275783e-01
##           C8           C9          C10          D1           D2
## -1.303918e-01 -1.118177e-01 -1.154676e-01  9.999642e-02  5.377944e-02
##           D3           D4           D5           D6           D7
##  6.854828e-02  4.130400e-02  8.017446e-02  5.263692e-02 -9.854824e-02
##           D8           D9          D10          E1           E2
## -5.691920e-02 -6.511046e-02 -7.936946e-02  1.209497e-01  1.092938e-01
##           E3           E4           E5           E6           E7
##  4.947434e-02  9.243309e-02  7.114615e-02  3.017627e-02 -3.570522e-02
##           E8           E9          E10          F1           F2
## -1.090806e-01 -5.496566e-02 -2.740387e-02  2.483609e-02  3.327546e-02
##           F3           F4           F5           F6           F7
##  2.733931e-02  6.192250e-02  7.334672e-02 -5.307307e-02 -2.221932e-02
##           F8           F9          F10          G1           G2
## -3.373039e-02 -5.380116e-02 -1.318506e-02  1.616875e-01  1.296174e-01
##           G3           G4           G5           G6           G7
##  1.047490e-01  1.493471e-01  1.437128e-01 -1.508467e-01 -1.417570e-01
##           G8           G9          G10          H2           H3
## -1.055049e-01 -1.283277e-01 -1.263001e-01  2.289127e-02 -1.145275e-02
##           H4           H5           H6           H7           H8
## -3.816794e-02  9.710007e-03  4.229346e-02 -3.918074e-02 -1.468912e-02
##           H9          H10          I1           I2           I3
##  1.065336e-02 -2.834743e-02 -9.357112e-02 -8.397495e-02 -9.345978e-02
##           I4           I5           I6           I7           I8
## -1.204297e-01 -6.917816e-02 -9.030692e-02  8.308076e-02  1.109384e-01
##           I9          I10          J1           J2           J3
##  1.003835e-01  7.510297e-02 -5.688594e-02 -6.726210e-02 -3.758735e-02
##           J4           J5           J6           J7           J8
## -5.480822e-02 -9.296578e-03 -4.710525e-02  2.092116e-02  5.827092e-03
##           J10          K1           K2           K3           K4
##  4.272056e-02 -1.084693e-01 -1.230838e-01 -1.172784e-01 -1.177718e-01
##           K5           K6           K7           K8           K9
## -1.087080e-01  1.186874e-01  1.141474e-01  4.051683e-02  8.682521e-02
##           K10          L1           L2           L3           L4
##  9.753317e-02 -9.689140e-02 -1.130090e-01 -8.213118e-02 -8.793013e-02
##           L5           L6           L7           L8           L9
```

```
## -8.813148e-02 -3.824644e-02 -8.894680e-02 6.208267e-02 7.899861e-02
##          L10          M1          M2          M3          M4
##  9.349789e-02  8.255904e-03  6.456284e-02  4.920668e-02  5.861934e-02
##          M5          M6          M7          M8          M9
##  4.829439e-02 -1.648400e-02 -4.538671e-03 -8.183895e-03 -8.270908e-03
##          M10         N1          N2          N3          N4
## -3.738425e-02 -1.381793e-01 -8.773667e-02 -7.098060e-02 -1.072238e-01
##          N5          N6          N7          N8          N9
## -4.112086e-02 -5.970872e-02 -7.428071e-02 1.179103e-01 1.216242e-01
##          N10         O1          O2          O3          O4
##  3.644911e-02 -2.252037e-02  6.567016e-02  2.011257e-02  1.256172e-02
##          O5          O6          O7          O8          O9
##  4.532869e-02 -3.817121e-03 -5.114938e-05 -5.269991e-02 -2.636245e-02
##          O10         P1          P2          P3          P4
## -7.256038e-02 -1.062205e-01 -8.666075e-02 -6.233608e-02 -6.825434e-02
##          P5          P6          P7          P8          P9
## -9.668191e-02 -5.877854e-02 -3.982269e-02 8.106026e-02 7.413901e-02
##          P10
##  8.031815e-02
```

Construct the equation for the first principal component as a string

```
equation <- paste("PC1 = ", paste0(loadings, "*x", 1:length(loadings), collapse = " + "), sep = " ")
```

Print the equation for the first principal component

```
print(equation)
```

```
## [1] "PC1 = 0.10769676517954*x1 + 0.119364669043886*x2 + 0.0556737307280377
*x3 + 0.0793380248754808*x4 + 0.115785788919937*x5 + 0.115101582949653*x6 + 0.
.0835380463635063*x7 + -0.0547944091957275*x8 + -0.0991297544354453*x9 + -0.0
552865550144107*x10 + 0.0231001756860651*x11 + 0.00458706274689637*x12 + -0.0
00351201652053501*x13 + 0.0389342832757614*x14 + 0.046013458013556*x15 + -0.0
0141948383265612*x16 + 0.0126407927400163*x17 + 0.0383863766034284*x18 + -0.0
212154266707533*x19 + -0.080204706001431*x20 + -0.0743282502149602*x21 + -0.0
27520663746799*x22 + -0.0169242664721286*x23 + 0.0738967162668336*x24 + 0.115
519868132091*x25 + 0.0943350507721503*x26 + 0.0851202412987988*x27 + 0.084121
2953205705*x28 + -0.0996737860967384*x29 + -0.127578309176443*x30 + -0.130391
764964882*x31 + -0.111817709199045*x32 + -0.115467575988171*x33 + 0.099996423
8678303*x34 + 0.0537794373865306*x35 + 0.068548277443337*x36 + 0.041303996247
7243*x37 + 0.0801744620543038*x38 + 0.0526369221676791*x39 + -0.0985482447943
725*x40 + -0.0569192045092007*x41 + -0.0651104563713546*x42 + -0.079369461499
1908*x43 + 0.120949741067595*x44 + 0.109293770664983*x45 + 0.049474341725365*
x46 + 0.0924330944912305*x47 + 0.0711461544168916*x48 + 0.0301762746672921*x4
9 + -0.0357052172349009*x50 + -0.109080562102081*x51 + -0.0549656595484347*x5
2 + -0.0274038669560141*x53 + 0.0248360922145136*x54 + 0.0332754645990715*x55
+ 0.0273393116822547*x56 + 0.0619224988480266*x57 + 0.0733467211779691*x58 +
-0.0530730664939163*x59 + -0.0222193222241554*x60 + -0.0337303945503304*x61 +
```

$-0.0538011640282941 \times x_{62} + -0.0131850616298436 \times x_{63} + 0.161687534724466 \times x_{64} + 0.129617380669824 \times x_{65} + 0.104749033862198 \times x_{66} + 0.149347128477129 \times x_{67} + 0.143712763973536 \times x_{68} + -0.150846749223437 \times x_{69} + -0.141757025808784 \times x_{70} + -0.105504935126467 \times x_{71} + -0.128327688634243 \times x_{72} + -0.126300085911192 \times x_{73} + 0.022891272539384 \times x_{74} + -0.0114527520423522 \times x_{75} + -0.0381679445881762 \times x_{76} + 0.00971000721877481 \times x_{77} + 0.042293457261871 \times x_{78} + -0.0391807433595187 \times x_{79} + -0.0146891240524678 \times x_{80} + 0.0106533632122419 \times x_{81} + -0.0283474299797577 \times x_{82} + -0.093571124390741 \times x_{83} + -0.0839749453470841 \times x_{84} + -0.0934597806082914 \times x_{85} + -0.120429651476777 \times x_{86} + -0.0691781613652704 \times x_{87} + -0.0903069202220367 \times x_{88} + 0.0830807574530391 \times x_{89} + 0.110938356756437 \times x_{90} + 0.100383527830159 \times x_{91} + 0.0751029664573097 \times x_{92} + -0.0568859355449997 \times x_{93} + -0.0672620961090386 \times x_{94} + -0.0375873478500297 \times x_{95} + -0.0548082217934934 \times x_{96} + -0.00929657834344465 \times x_{97} + -0.0471052501751023 \times x_{98} + 0.0209211604476965 \times x_{99} + 0.00582709248615015 \times x_{100} + 0.042720563458218 \times x_{101} + -0.108469298872716 \times x_{102} + -0.123083785034304 \times x_{103} + -0.117278408213206 \times x_{104} + -0.117771765923043 \times x_{105} + -0.108707979006803 \times x_{106} + 0.118687425884527 \times x_{107} + 0.114147379206718 \times x_{108} + 0.0405168306229751 \times x_{109} + 0.0868252109770417 \times x_{110} + 0.0975331687219128 \times x_{111} + -0.0968914010704701 \times x_{112} + -0.113009004785669 \times x_{113} + -0.0821311751191762 \times x_{114} + -0.0879301280189712 \times x_{115} + -0.0881314797343172 \times x_{116} + -0.0382464400024922 \times x_{117} + -0.0889467950923519 \times x_{118} + 0.0620826699950573 \times x_{119} + 0.0789986069141265 \times x_{120} + 0.0934978906208914 \times x_{121} + 0.00825590435231281 \times x_{122} + 0.0645628365473102 \times x_{123} + 0.0492066833867297 \times x_{124} + 0.0586193438407804 \times x_{125} + 0.0482943864763896 \times x_{126} + -0.0164840045404703 \times x_{127} + -0.00453867063183092 \times x_{128} + -0.00818389475679192 \times x_{129} + -0.00827090776368217 \times x_{130} + -0.0373842499768471 \times x_{131} + -0.138179342800253 \times x_{132} + -0.0877366683251409 \times x_{133} + -0.0709806039790091 \times x_{134} + -0.107223792751675 \times x_{135} + -0.0411208590457524 \times x_{136} + -0.0597087162880416 \times x_{137} + -0.0742807125081411 \times x_{138} + 0.117910290447822 \times x_{139} + 0.121624206195682 \times x_{140} + 0.03644910502906 \times x_{141} + -0.0225203670807466 \times x_{142} + 0.0656701570819524 \times x_{143} + 0.0201125692050463 \times x_{144} + 0.0125617239265296 \times x_{145} + 0.0453286923695985 \times x_{146} + -0.00381712072880992 \times x_{147} + -5.11493776968036e-05 \times x_{148} + -0.052699913935179 \times x_{149} + -0.0263624538366952 \times x_{150} + -0.0725603825258931 \times x_{151} + -0.10622054124566 \times x_{152} + -0.0866607512927252 \times x_{153} + -0.0623360773488486 \times x_{154} + -0.0682543449309424 \times x_{155} + -0.0966819136118558 \times x_{156} + -0.0587785446564874 \times x_{157} + -0.0398226862778885 \times x_{158} + 0.0810602599320182 \times x_{159} + 0.0741390111625445 \times x_{160} + 0.0803181506472815 \times x_{161}"$

$\# \text{PC1} = 0.10769676517954x_1 + 0.119364669043886x_2 + 0.0556737307280377x_3 + 0.0793380248754808x_4 + 0.115785788919937x_5 + 0.115101582949653x_6 + 0.0835380463635063x_7 + -0.0547944091957275x_8 + -0.0991297544354453x_9 + -0.0552865550144107x_{10} + 0.0231001756860651x_{11} + 0.00458706274689637x_{12} + -0.000351201652053501x_{13} + 0.0389342832757614x_{14} + 0.046013458013556x_{15} + -0.00141948383265612x_{16} + 0.0126407927400163x_{17} + 0.0383863766034284x_{18} + -0.0212154266707533x_{19} + -0.080204706001431x_{20} + -0.0743282502149602x_{21} + -0.027520663746799x_{22} + -0.0169242664721286x_{23} + 0.0738967162668336x_{24} + 0.115519868132091x_{25} + 0.0943350507721503x_{26} + 0.0851202412987988x_{27} + 0.0841212953205705x_{28} + -0.0996737860967384x_{29} + -0.127578309176443x_{30} + -0.130391764964882x_{31} + -0.111817709199045x_{32} + -0.115467575988171x_{33} + 0.0999964238678303x_{34} + 0.0537794373865306x_{35} + 0.068548277443337x_{36} + 0.0413039962477243x_{37} + 0.0801744620543038x_{38} + 0.0526369221676791x_{39} + -0.0985482447943725x_{40} + -0.0569192045092007x_{41} + -0.0651104563713546x_{42} + -$

$0.0793694614991908x43 + 0.120949741067595x44 + 0.109293770664983x45 +$
 $0.049474341725365x46 + 0.0924330944912305x47 + 0.0711461544168916x48 +$
 $0.0301762746672921x49 + -0.0357052172349009x50 + -0.109080562102081x51 + -$
 $0.0549656595484347x52 + -0.0274038669560141x53 + 0.0248360922145136x54 +$
 $0.0332754645990715x55 + 0.0273393116822547x56 + 0.0619224988480266x57 +$
 $0.0733467211779691x58 + -0.0530730664939163x59 + -0.0222193222241554x60 + -$
 $0.0337303945503304x61 + -0.0538011640282941x62 + -0.0131850616298436x63 +$
 $0.161687534724466x64 + 0.129617380669824x65 + 0.104749033862198x66 +$
 $0.149347128477129x67 + 0.143712763973536x68 + -0.150846749223437x69 + -$
 $0.141757025808784x70 + -0.105504935126467x71 + -0.128327688634243x72 + -$
 $0.126300085911192x73 + 0.022891272539384x74 + -0.0114527520423522x75 + -$
 $0.0381679445881762x76 + 0.00971000721877481x77 + 0.042293457261871x78 + -$
 $0.0391807433595187x79 + -0.0146891240524678x80 + 0.0106533632122419x81 + -$
 $0.0283474299797577x82 + -0.093571124390741x83 + -0.0839749453470841x84 + -$
 $0.0934597806082914x85 + -0.120429651476777x86 + -0.0691781613652704x87 + -$
 $0.0903069202220367x88 + 0.0830807574530391x89 + 0.110938356756437x90 +$
 $0.100383527830159x91 + 0.0751029664573097x92 + -0.0568859355449997x93 + -$
 $0.0672620961090386x94 + -0.0375873478500297x95 + -0.0548082217934934x96 + -$
 $0.00929657834344465x97 + -0.0471052501751023x98 + 0.0209211604476965x99 +$
 $0.00582709248615015x100 + 0.042720563458218x101 + -0.108469298872716x102 + -$
 $0.123083785034304x103 + -0.117278408213206x104 + -0.117771765923043x105 + -$
 $0.108707979006803x106 + 0.118687425884527x107 + 0.114147379206718x108 +$
 $0.0405168306229751x109 + 0.0868252109770417x110 + 0.0975331687219128x111 + -$
 $0.0968914010704701x112 + -0.113009004785669x113 + -0.0821311751191762x114 + -$
 $0.0879301280189712x115 + -0.0881314797343172x116 + -0.0382464400024922x117 + -$
 $0.0889467950923519x118 + 0.0620826699950573x119 + 0.0789986069141265x120 +$
 $0.0934978906208914x121 + 0.00825590435231281x122 + 0.0645628365473102x123 +$
 $0.0492066833867297x124 + 0.0586193438407804x125 + 0.0482943864763896x126 + -$
 $0.0164840045404703x127 + -0.00453867063183092x128 + -0.00818389475679192x129$
 $+ -0.00827090776368217x130 + -0.0373842499768471x131 + -0.138179342800253x132 +$
 $-0.0877366683251409x133 + -0.0709806039790091x134 + -0.107223792751675x135 + -$
 $0.0411208590457524x136 + -0.0597087162880416x137 + -0.0742807125081411x138 +$
 $0.117910290447822x139 + 0.121624206195682x140 + 0.03644910502906x141 + -$
 $0.0225203670807466x142 + 0.0656701570819524x143 + 0.0201125692050463x144 +$
 $0.0125617239265296x145 + 0.0453286923695985x146 + -0.00381712072880992x147 + -$
 $5.11493776968036e-05x148 + -0.052699913935179x149 + -0.0263624538366952x150 + -$
 $0.0725603825258931x151 + -0.10622054124566x152 + -0.0866607512927252x153 + -$
 $0.0623360773488486x154 + -0.0682543449309424x155 + -0.0966819136118558x156 + -$
 $0.0587785446564874x157 + -0.0398226862778885x158 + 0.0810602599320182x159 +$
 $0.0741390111625445x160 + 0.0803181506472815x161$

Interpret the components based on loadings

```

interpret_components <- function(pca_result, num_components) {
  for (i in 1:num_components) {
    cat("Component", i, ":\n")
  }
}

```

```

    comp_loadings <- sort(pca_result$rotation[, i], decreasing = TRUE)
    top_variables <- names(comp_loadings)[1:10] # adjust based on how many t
op variables you want to look at
    cat("Top variables:", toString(top_variables), "\n")
    cat("\n")
  }
}

```

Interpret the first few components

```
interpret_components(pca_result, 24)
```

```

## Component 1 :
## Top variables: G1, G4, G5, G2, N9, E1, A2, K6, N8, A5
##
## Component 2 :
## Top variables: B6, J5, M4, F7, B1, B2, F9, F6, F10, J1
##
## Component 3 :
## Top variables: E6, L3, C6, E4, B10, N10, P2, P1, E2, C10
##
## Component 4 :
## Top variables: H8, A9, M7, H9, F7, A10, M9, I6, F9, M6
##
## Component 5 :
## Top variables: O3, O4, D5, O1, D1, D2, O5, F1, P4, O2
##
## Component 6 :
## Top variables: K9, K7, P1, H3, L4, K10, P2, H5, K6, H4
##
## Component 7 :
## Top variables: K4, K1, E3, K5, E5, E2, E4, N8, F4, I4
##
## Component 8 :
## Top variables: O10, P7, O8, H8, P3, E3, O9, I7, A10, H7
##
## Component 9 :
## Top variables: N3, N7, N6, N4, H5, N5, H6, J4, P1, E5
##
## Component 10 :
## Top variables: I10, I7, I9, I8, D2, P2, M1, J3, P1, E4
##
## Component 11 :
## Top variables: I9, A7, H8, A1, D5, I10, A4, F7, D1, F9
##
## Component 12 :
## Top variables: O3, J4, O1, J2, E3, P7, O4, D7, J3, D9
##
## Component 13 :
## Top variables: H4, E10, P7, O10, P3, O9, F3, D8, O8, B1

```

```

##
## Component 14 :
## Top variables: F3, G9, B13, O8, B12, I2, G6, J1, O9, J7
##
## Component 15 :
## Top variables: P7, H6, P3, I1, N8, B9, H2, H5, N9, M1
##
## Component 16 :
## Top variables: N5, N3, B12, L5, E9, N7, B13, N6, B8, E7
##
## Component 17 :
## Top variables: O4, O5, B4, A8, B5, B3, O6, O7, B7, O8
##
## Component 18 :
## Top variables: D5, D1, O4, H5, P8, O5, C5, H6, M6, D2
##
## Component 19 :
## Top variables: K3, D2, B2, P7, K8, E4, E2, N2, P8, C5
##
## Component 20 :
## Top variables: M5, M3, M2, A8, P7, P9, F5, M10, I5, E3
##
## Component 21 :
## Top variables: H4, E5, E3, I1, B13, J10, B12, M2, P3, A10
##
## Component 22 :
## Top variables: J7, M6, H4, D9, M8, P3, P7, D6, C3, O5
##
## Component 23 :
## Top variables: P9, L8, D9, H9, H4, J10, O10, M1, H6, D3
##
## Component 24 :
## Top variables: L8, P9, A10, I5, P5, J4, M5, J2, N10, P10

```

Component 1 (12.9%): Likely reflects the most dominant pattern in the dataset, explaining a significant portion of the variance.

Component 2 (19.0%): Captures additional variance, likely representing a secondary pattern that is distinct from the first.

Component 3 (24.7%): Builds on the variance explained, possibly indicating another underlying factor or pattern.

Component 4 (29.1%): Continues to add nuanced variance, which could be a less dominant but still relevant pattern.

Component 5 (32.7%): Represents further subtle patterns in the data not captured by previous components.

Component 6 (35.5%): Contributes to a more complete picture of data structure, explaining additional complexity.

Component 7 (37.3%): Adds a slight increase to the explained variance, indicating diminishing returns on new patterns.

Component 8 (39.0%): Reflects a finer pattern in the data structure, adding small incremental variance.

Component 9 (40.4%): Suggests a continuing trend of components explaining smaller portions of variance.

Component 10 (41.6%): Points to further nuanced variance that may be linked to more obscure patterns.

Component 11 (42.7%): Continues the trend of progressively smaller contributions to the total variance.

Component 12 (43.8%): Indicates an even finer aspect of the data's structure being captured.

Component 13 (44.8%): Adds a small but potentially meaningful pattern to the overall variance explained.

Component 14 (45.8%): Suggests continuing to capture finer details within the dataset's structure.

Component 15 (46.7%): Shows the components are starting to explain smaller and smaller slices of variance.

Component 16 (47.6%): Reflects increasingly subtle patterns within the data.

Component 17 (48.4%): Indicates that additional unique variance is being identified by the model.

Component 18 (49.2%): Demonstrates a slight increase in cumulative variance explained by the PCA.

Component 19 (49.9%): Approaches 50% of the total variance explained by the dataset.

Component 20 (50.7%): Crosses the halfway mark of total variance explained, with diminishing increments.

Component 21 (51.3%): Continues to add to the explained variance, albeit at a reduced rate.

Component 22 (52.0%): Contributes to a more nuanced understanding of the data's variance.

Component 23 (52.7%): Highlights the pattern of diminishing marginal returns in variance explanation.

Component 24 (53.3%): Final component in this series, adding a small increment to the total explained variance.

QUESTION 4

Extract scores for the first 24 observations.

```
component_scores <- pca_result$x[,1:24]
```

Five Number Summary

```
Summary <- apply(component_scores,2,summary)
print(Summary)
```

##	PC1	PC2	PC3	PC4	
PC5					
## Min.	-1.780669e+01	-1.420064e+01	-1.508590e+01	-1.115662e+01	-1.104487e+01
## 1st Qu.	-2.951717e+00	-2.191567e+00	-1.957384e+00	-1.717616e+00	-1.519221e+00
## Median	1.245435e-01	-2.151332e-01	2.299843e-03	-4.389942e-02	1.466709e-02
## Mean	3.064177e-14	-2.268426e-14	3.314491e-14	7.376064e-15	-3.685231e-14
## 3rd Qu.	3.161699e+00	1.990250e+00	1.963092e+00	1.591782e+00	1.529951e+00
## Max.	1.620390e+01	1.382057e+01	1.301498e+01	1.732996e+01	1.203836e+01
##	PC6	PC7	PC8	PC9	P
C10					
## Min.	-1.624091e+01	-8.680438e+00	-8.215789e+00	-7.662324e+00	-7.883111e+00
## 1st Qu.	-1.208301e+00	-1.126494e+00	-1.079348e+00	-9.591025e-01	-8.646895e-01
## Median	5.622701e-02	-3.834137e-02	-3.173126e-02	-1.877827e-03	1.657820e-02
## Mean	-3.120376e-14	1.482876e-14	-1.673562e-14	2.706751e-14	-1.489919e-15
## 3rd Qu.	1.282374e+00	1.074480e+00	1.037643e+00	9.693335e-01	8.904326e-01
## Max.	1.799326e+01	8.716574e+00	7.930124e+00	6.937151e+00	6.935781e+00
##	PC11	PC12	PC13	PC14	P
C15					
## Min.	-7.137170e+00	-5.871243e+00	-6.964785e+00	-9.480928e+00	-6.071959e+00
## 1st Qu.	-8.804685e-01	-8.703041e-01	-8.390063e-01	-8.148673e-01	-7.610609e-01
## Median	-2.155496e-02	-1.782019e-02	-4.688206e-14	-1.075936e-02	2.092650e-02
## Mean	2.681150e-14	1.704333e-14	-4.950467e-14	4.349249e-15	-2.405615e-15

-14					
## 3rd Qu.	8.497695e-01	8.492564e-01	8.271623e-01	7.919836e-01	7.953443e-01
## Max.	7.897526e+00	6.280855e+00	6.498877e+00	6.610967e+00	6.451816e+00
##	PC16	PC17	PC18	PC19	P
C20					
## Min.	-6.320504e+00	-6.076141e+00	-5.571066e+00	-5.043682e+00	-4.918983e+00
## 1st Qu.	-7.643246e-01	-7.338674e-01	-7.240610e-01	-7.096110e-01	-6.877536e-01
## Median	-5.680075e-04	-1.272925e-02	3.862415e-03	1.367928e-14	1.025486e-14
## Mean	-7.306652e-15	5.561276e-15	-3.260594e-14	1.872589e-14	1.275742e-14
## 3rd Qu.	7.589272e-01	7.246393e-01	7.232586e-01	7.109999e-01	6.872246e-01
## Max.	5.766502e+00	5.289025e+00	5.509233e+00	5.937037e+00	5.134520e+00
##	PC21	PC22	PC23	PC24	
## Min.	-6.423059e+00	-6.285683e+00	-5.121086e+00	-4.842667e+00	
## 1st Qu.	-6.755120e-01	-6.526112e-01	-6.601196e-01	-6.563451e-01	
## Median	4.119191e-14	-4.658864e-14	-5.973983e-04	-1.618281e-14	
## Mean	4.811662e-14	-4.620914e-14	-3.099791e-14	-1.655490e-14	
## 3rd Qu.	6.795493e-01	6.577314e-01	6.512582e-01	6.463769e-01	
## Max.	4.666523e+00	6.745779e+00	6.748712e+00	4.930584e+00	

The five-number summary for the component scores reveals the spread and central tendency of respondent scores across the 24 principal components.

Interpretation related to the respondents' personalities and the distribution of component scores:

Wide Score Distribution: Components with a large range between the minimum and maximum values, such as PC1, PC2, PC3, and PC18, indicate a wide variation in respondent scores. This suggests that the corresponding personality traits or factors captured by these components vary significantly among the respondents.

For instance, PC1, with a range from approximately -17.81 to 16.20, shows the most substantial variation, indicating that the personality trait or factor it represents is expressed very differently across individuals.

Narrow Score Distribution: Components with a smaller range between the minimum and maximum values, like PC24, suggest a narrower variation in how the corresponding traits are expressed among respondents. The relatively tighter distribution implies that respondents are more similar in terms of the personality traits or factors these components represent.

Central Tendency: The median values, mostly hovering around zero, indicate that for most components, the central tendency of the scores is near the average of the distribution. This is typical in PCA, where scores are often centered around zero.

Components where the median significantly deviates from zero might indicate a skew in the distribution of traits among the respondents.

Components with Wide Scores: Specifically, PC1, PC2, PC3, and PC18 have wider score distributions, suggesting significant variability among respondents regarding the traits these components represent.

Components with Relatively Narrow Score Distribution: Components like PC24, with its maximum value being notably lower than those of the components with wider distributions, exhibit a relatively narrow score distribution, indicating less variability among respondents for the traits these components capture.

The five-number summary provides a quantitative basis to understand the variability and central tendencies of component scores, which, when mapped back to original variables, can offer insights into the diversity and commonality of personality traits among the respondents.

QUESTION 5

Load necessary library

```
library(stats)
```

Conduct Factor Analysis

```
fa_result <- factanal(imputed_data, factors = 24)
fa_result_1 <- fa(imputed_data, nfactors = 24, rotate = "varimax")
```

Print the summary of Factor Analysis

```
print(summary(fa_result))
```

##	Length	Class	Mode
## converged	1	-none-	logical
## loadings	3864	loadings	numeric
## uniquenesses	161	-none-	numeric
## correlation	25921	-none-	numeric
## criteria	3	-none-	numeric
## factors	1	-none-	numeric
## dof	1	-none-	numeric
## method	1	-none-	character
## rotmat	576	-none-	numeric
## STATISTIC	1	-none-	numeric
## PVAL	1	-none-	numeric
## n.obs	1	-none-	numeric
## call	3	-none-	call

```
print(summary(fa_result_1))

##
## Factor analysis with Call: fa(r = imputed_data, nfactors = 24, rotate = "v
arimax")
##
## Test of the hypothesis that 24 factors are sufficient.
## The degrees of freedom for the model is 9292 and the objective function w
as 3.8
## The number of observations was 49159 with Chi Square = 186292.9 with p
rob < 0
##
## The root mean square of the residuals (RMSA) is 0.01
## The df corrected root mean square of the residuals is 0.01
##
## Tucker Lewis Index of factoring reliability = 0.927
## RMSEA index = 0.02 and the 10 % confidence intervals are 0.02 0.02
## BIC = 85913.12NULL
```

KMO Test

```
kmo <- KMO(imputed_data)
```

Bartlett's Test

```
bartlett <- cortest.bartlett(cor(imputed_data))

## Warning in cortest.bartlett(cor(imputed_data)): n not specified, 100 used
```

When comparing the variance explained by the first 24 components of PCA with the variance explained by Factor Analysis (FA) for the same number of factors, several key differences emerge:

PCA Variance:

In PCA, the cumulative variance explained by the first 24 components reaches approximately 53.3%. This indicates that these components together account for a little over half of the total variance in the dataset. The variance explained by individual PCA components decreases with each subsequent component, but each contributes significantly to the total variance, with the first few components explaining a large portion of the variance.

FA Variance:

In FA, the cumulative variance explained by 24 factors is 45.2%. This suggests that these factors together account for less than half of the total variance in the data, which is lower than the variance explained by the first 24 PCA components. The sum of squared loadings (SS loadings) for each factor in FA represents the amount of common variance explained by

that factor. The decrease in SS loadings from one factor to the next is generally smoother than in PCA, indicating a more even distribution of explained variance across factors.

Key Differences:

Focus on Variance: PCA components are constructed to explain the maximum possible variance, with the first component explaining the most variance and each subsequent component explaining less.

FA factors, however, aim to model the underlying structure by capturing common variance shared among variables, which may not necessarily be the largest variances. **Common vs. Total Variance:** PCA explains total variance (common + unique variance of variables), while FA focuses on common variance only, excluding unique variances and error terms. This is why PCA tends to explain more total variance with the same number of components/factors than FA.

Interpretation:

PCA components are orthogonal and do not allow for the interpretation of correlations between components. FA factors can be correlated, especially when using oblique rotation, which can provide more nuanced insights into the relationships among underlying constructs.

Utility:

PCA is more suited for dimensionality reduction and data summarization, whereas FA is better for identifying and interpreting latent constructs underlying the data.

In summary, the first 24 PCA components explain more total variance than the 24 factors in FA, reflecting their different objectives and approaches to handling variance in the data. PCA is more efficient in data reduction by maximizing explained variance, while FA provides deeper insights into the latent structure of the data by focusing on shared variance among variables.