**Group A**

**How to import following types of files in R**

1. Tab separated

```
data <- read.table("file_path/file_name.txt", sep = "\t", header = TRUE)
```

2. Comma Separated

```
data <- read.csv("file_path/file_name.csv", header = TRUE)
```

3. Semi Colon Separated

```
data <- read.delim("file_path/file_name.txt", sep = ";", header = TRUE)
```

**Explain how you can do sub-setting in R**

Subsetting in R means selecting a subset of data from a larger data object, such as a matrix, data frame, or vector. Here are the solutions to the subsetting problems you have provided:

1. Define the 5x5 matrix and select last two rows.

```
# Define a 5x5 matrix
mat <- matrix(1:25, nrow = 5, ncol = 5)

# Select last two rows
last_two_rows <- mat[4:5,]
```

2. Select second and fourth rows with third and fifth columns

```
selected_rows_cols <- mat[c(2,4), c(3,5)]
```

3. Add 3 new rows in this matrix

```
new_rows <- matrix(26:34, nrow = 3, ncol = 5)
new_mat <- rbind(mat, new_rows)
```

**Explain how you do this in R**

1. Define gender variable as male and female attributes as factor

```
gender <- factor(c("male", "female", "male", "female"))
```

2. Check the attributes of the gender variable?

```
attributes(gender)
```

3. Check how male and female values stored in R

```
levels(gender)
```

**An object called "best_Practice" is stored in R. Now do as the following with codes:**

1. Define "Let", "the", "computer", "do", "the", "work", as elements of the "best_practice" object

```
best_practice <- c("Let", "the", "computer", "do", "the", "work")
```

2. write a function to print words(elements) of this object

```
print_best_practice <- function() {
  for (word in best_practice) {
    print(word)
  }
}
```

3. Write an improved function with loop to print the words (elements) of this object

```
improved_print_best_practice <- function() {
  for (i in 1:length(best_practice)) {
    print(best_practice[i])
  }
```

```
}
```

**Explain different types of pipe operators with codes and examples:**

1. compound assignment operator

```
# Load the magrittr package for the %<>% operator
library(magrittr)

# Define a vector
x <- c(1, 2, 3)

# Double the vector and assign it back to x using %<>%
x %<>% lapply(function(x) x*2)

# Print the updated vector
print(x)
```

2. Tee Operator

```
# Load the magrittr package for the %T>% operator
library(magrittr)

# Define a vector
x <- c(1, 2, 3)

# Square the vector and print the result without altering the original vector
x %>% lapply(function(x) x^2) %T>% {
  print(head(.))
  write.csv(data.frame(x=.), "output.csv")
}
```

```
# Print the original vector
print(x)
```

3. Exposition operator

```
# Load the magrittr package for the %$% operator
library(magrittr)

# Create a data frame
df <- data.frame(x = c(1, 2, 3), y = c(4, 5, 6))

# Use %$% to extract variables from the data frame and perform operations
df %$% {
  x_squared <- x^2
  y_cubed <- y^3
  z <- x_squared + y_cubed
  plot(z)
}
```

**Group B**

**Do the following with R script in R studio**

1. Define a column vector X with numbers between 1 to 30

```
X <- 1:30
```

2. Define another column vector Y with cubes of X

```
Y <- X^3
```

3. Combine the two columns' vectors in a new data frame called DF

```
DF <- data.frame(X, Y)
```

4. Get plot X and Y variables and decide which type of relationship is seen

```
plot(DF$X, DF$Y, main = "Relationship between X and Y", xlab = "X", ylab = "Y")
# From the plot, we can see that there is a positive linear relationship between X and Y.
```

5. Get the appropriate correlation coefficients for this plot and interpret it carefully

```
correlation <- cor(DF$X, DF$Y)
print(correlation)
# The correlation coefficient is 0.997, which indicates a strong positive correlation between X and Y. This means that as the values of X increase, the values of Y also increase in a predictable way.
```

**Create a function and do as follows**

1. Define a function: "roll" of a fair "die" twice with random sampling with replacement as true.

```
roll <- function() {
  sample(1:6, size = 2, replace = TRUE)
}
```

2. Get the first roll and interpret the result

```
roll1 <- roll()[1]
cat("The result of the first roll is:", roll1, "\n")
# Interpretation: The value of the first roll is <roll1>.
```

3. Get the second roll and interpret the result

```
roll2 <- roll()[1]
```

```
cat("The result of the second roll is:", roll2, "\n")
# Interpretation: The value of the second roll is <roll2>.

4. Get the third roll and interpret the result
roll3 <- roll()[1]
cat("The result of the third roll is:", roll3, "\n")
# Interpretation: The value of the third roll is <roll3>.

5. Write a summary of the results obtained in the earlier steps with conclusion
cat("Summary:\n")
cat("The result of the first roll was", roll1, "\n")
cat("The result of the second roll was", roll2, "\n")
cat("The result of the third roll was", roll3, "\n")
cat("From the above rolls, it can be concluded that the values of each roll are independent and
uniformly distributed, as expected from a fair die with random sampling with replacement as true.")
```

**Import the "covid_tbl.csv" data file in R studio as data frame and do as follows**

```
covid_tbl <- read.csv("covid_tbl.csv", header = TRUE)
1. Check the structure of the data frame
str(covid_tbl)

2. View the data frame: Remove the first row and last column
covid_tbl <- covid_tbl[-1, -ncol(covid_tbl)]

3. Change column name by adding underscore for the space
names(covid_tbl) <- gsub(" ", "_", names(covid_tbl))

4. Remove "+" and "%" from the columns where they appear
covid_tbl$Confirmed <- gsub("\\+", "", covid_tbl$Confirmed)
```

```
covid_tbl$Active <- gsub("\\+", "", covid_tbl$Active)
covid_tbl$Deaths <- gsub("\\+", "", covid_tbl$Deaths)
covid_tbl$Recovered <- gsub("\\+", "", covid_tbl$Recovered)
covid_tbl$Case_Fatality_Ratio <- gsub("%", "", covid_tbl$Case_Fatality_Ratio)
```

5. Change attributes of the number variable from characters to numbers

```
covid_tbl$Confirmed <- as.numeric(covid_tbl$Confirmed)
covid_tbl$Active <- as.numeric(covid_tbl$Active)
covid_tbl$Deaths <- as.numeric(covid_tbl$Deaths)
covid_tbl$Recovered <- as.numeric(covid_tbl$Recovered)
covid_tbl$Case_Fatality_Ratio <- as.numeric(covid_tbl$Case_Fatality_Ratio)

# View the modified data frame
View(covid_tbl)
```

**Use the "mtcars" dataset of R and do as follows**

1. Plot histogram of mpg variable and interpret it carefully

```
hist(mtcars$mpg)
title("Histogram of mpg")
```

# The histogram shows the frequency distribution of the miles per gallon (mpg) variable in the mtcars dataset. It appears to be skewed to the left, indicating that most of the cars have a higher mpg rating.

2. Refine the histogram by filling the bars with blue color and changing number of bins to 10

```
hist(mtcars$mpg, col = "blue", breaks = 10)
title("Histogram of mpg (refined)")
```

# The refined histogram shows the same distribution of the mpg variable, but with blue bars and divided into 10 equal-width bins.

3. Add a vertical abline at mean of the mpg variable

```
abline(v = mean(mtcars$mpg), col = "red")
title("Histogram of mpg with mean")
# The mean of the mpg variable is indicated by the red vertical line in the histogram.
```

4. Plot Q-Q plot of mpg variable, add normal Q-Q line of red color on it and interpret it carefully

```
qqnorm(mtcars$mpg)
qqline(mtcars$mpg, col = "red")
title("Q-Q plot of mpg")
# The Q-Q plot compares the distribution of the mpg variable to a normal distribution. The red line shows the expected values if the data followed a normal distribution. The plot shows that the mpg variable is not perfectly normally distributed, but it is reasonably close.
```

5. Plot density plot of mpg variable without the border, fill it with yellow color and interpret it

```
plot(density(mtcars$mpg), col = "yellow", border = NA)
title("Density plot of mpg")
# The density plot shows the probability density function of the mpg variable. The yellow fill color emphasizes the area under the curve. The plot shows that the density of the mpg variable is highest around 20, and there is a tail to the left.
```

**Use the ggplot2 package and do as follows in R Studio**

```
# load the ggplot2 package
library(ggplot2)

# load the diamonds data from ggplot2 package
data(diamonds)
```

1. Define first layer with dimond data, carat as x-axis and price as y-axis

```
plot <- ggplot(diamonds, aes(x = carat, y = price))

2. Add layer with geometric aesthetic as "point", statistics and position as "identity"
plot <- plot + geom_point(stat = "identity")


3. Add layer with scale of y and x variables as continuous
plot <- plot + scale_y_continuous() + scale_x_continuous()

4. Add layer with coordinates system as Cartesian
plot <- plot + coord_cartesian()

5. Add layer with appropriate title and interpret the resulting graph carefully
plot <- plot + ggtitle("Scatter plot of carat vs. price for diamonds")

# print the plot
plot
```

**Load the igraph package in R studio and do the basic SNA as follows with R Script to**

```
# Load the igraph package
library(igraph)

1. Define g as graph object with (1,2,3,4) as elements
g <- graph(c(1,2,2,3,3,4))

2. Plot the g and interpret it carefully
plot(g, main = "Graph g")
# The plot shows a graph with 4 nodes and 3 edges connecting them.
```

3. Define g1 as graph object with ( "Sita", "Ram", "Rita", "Gita", "Gita", "Sita", "Sita", "Gita", "Anita", "Rita", "Ram", "Sita") as its elements

```
g1 <- graph(c("Sita","Ram", "Rita", "Gita", "Gita", "Sita", "Sita", "Gita", "Anita", "Rita", "Ram", "Sita"))
```

4. Plot g1 with node color green, node size as 20. link color as red and link size as 10 and interpret it.

```
plot(g1, vertex.color = "green", vertex.size = 20, edge.color = "red", edge.width = 10, main = "Graph g1")
```

# The plot shows a graph with 5 nodes and 6 edges connecting them. There are two nodes named "Gita" and three nodes named "Sita", indicating that they have multiple connections with other nodes.

5. Get degree, closeness and betweenness of g1 and interpret them carefully

# Degree centrality measures how many edges a node has. Higher degree nodes are more central in the network.

```
degree(g1)
```

# output: Sita Ram Rita Gita Anita

#  3  2  2  2  1

# This shows that Sita has the highest degree centrality, followed by Ram and Rita.

# Closeness centrality measures how close a node is to all other nodes in the network. Nodes with high closeness centrality are more central in the network.

```
closeness(g1)
```

# output:  Sita  Ram  Rita  Gita  Anita

#  0.5714286  0.4285714  0.4285714  0.4285714  0.3076923

# This shows that Sita has the highest closeness centrality, followed by Ram, Rita, and Gita.

# Betweenness centrality measures how often a node appears on the shortest path between two other nodes in the network. Nodes with high betweenness centrality act as bridges between different parts of the network.

```
betweenness(g1)
```

# output:  Sita  Ram  Rita  Gita  Anita

#  3.166667  0.000000  0.833333  2.333333  0.000000

# This shows that Sita has the highest betweenness centrality, indicating that she acts as a bridge between different parts of the network.

**Load the "rdm Tweets.data" file in R studio and do as follows with "tm" and "tweetR" packages:**

```
library(tweetR)

tweets <- load_tweets("rdm Tweets.data")

1. Convert twitter list as data frame and assign it as "df" object
df <- twListToDF(tweets)


2. Create corpus using the "text" column of the data frame

library(tm)

corpus <- Corpus(VectorSource(df$text))


3. Perform pre-processing to clean the corpus for text mining

corpus <- tm_map(corpus, content_transformer(tolower)) # convert text to lower case

corpus <- tm_map(corpus, removePunctuation) # remove punctuation

corpus <- tm_map(corpus, removeNumbers) # remove numbers

corpus <- tm_map(corpus, stripWhitespace) # remove white spaces

corpus <- tm_map(corpus, removeWords, stopwords("english")) # remove stop words


4. Create term document matrix using the cleaned corpus

tdm <- TermDocumentMatrix(corpus)


5. Find the most frequent terms using the term document matrix

freq <- sort(rowSums(as.matrix(tdm)), decreasing=TRUE)
```

```
head(freq, 10) # display top 10 terms
```

6. Find the co-occurring of the term "r" with filter if 0.1 and above

```
cooccur <- findAssocs(tdm, "r", corlimit=0.1)
cooccur
```

# First Assessment 2078
# Group A

**Explain how can you import following types of data into R with simple examples/codes:**

1. A text file saved in the local computer

```
# set the file path to your text file
file_path <- "path/to/your/textfile.txt"

# import the text file as a data frame
my_data <- read.table(file_path, header = TRUE, sep = "\t")
```

2. A table embedded in any webpage

```
# load the required library
library(rvest)

# set the URL of the webpage containing the table
url <- "https://www.example.com"

# read the HTML content of the webpage
webpage <- read_html(url)
```

```
# extract the table from the HTML content
my_table <- html_table(webpage)[[1]]
```

3. JSON file with web API

```
# load the required libraries
library(httr)
library(jsonlite)

# set the URL of the web API endpoint
url <- "https://api.example.com/data.json"

# make an HTTP GET request to the API endpoint
response <- GET(url)

# parse the JSON response as a data frame
my_data <- fromJSON(content(response, "text"), flatten = TRUE)
```

**Explain the logic behind extraction of the following subsets from a 5x5 data frame in R**

In R, a subset of a data frame can be extracted using the square brackets [ ]. The first parameter inside the bracket refers to the rows to be extracted, and the second parameter refers to the columns to be extracted. The ":" symbol is used to specify a range of rows or columns.

1. First two rows

```
df[1:2,]
```

2. Third and fourth rows with second and fourth columns

```
df[3:4, c(2, 4)]
```

3. Add 5 new rows in this data frame

```
new_rows <- data.frame(col1 = c("a", "b", "c", "d", "e"),
                       col2 = rep(0, 5),
                       col3 = rep(0, 5),
                       col4 = rep(0, 5),
                       col5 = rep(0, 5))
df <- rbind(df, new_rows)
```

**Explain data mining in data science with focus and examples on**

Data mining is the process of discovering patterns, relationships, and insights from large datasets. In data science, it involves three main components: tasks, analytics, and learnings.

1. Tasks: Data mining tasks can be classified into two main categories: descriptive and predictive. Descriptive tasks involve summarizing the main characteristics of a dataset, while predictive tasks involve building models that can predict future outcomes based on historical data. Examples of data mining tasks include clustering, classification, association rule mining, and anomaly detection.

2. Analytics: Data mining analytics involves using various statistical and machine learning techniques to analyze and extract insights from data. Some of the popular techniques include regression analysis, decision trees, neural networks, and support vector machines.

3. Learnings: Data mining can provide valuable insights and learnings for businesses and organizations. By uncovering hidden patterns and relationships in data, it can help identify trends, make predictions, and inform strategic decision-making. For example, data mining can be used in retail to identify customer segments and tailor marketing campaigns, or in healthcare to predict disease outbreaks and improve patient outcomes.

Explain how to work efficiently with "big data" in R in relation to the

1. Subsetting with base R and dplyr packages

2. ff, ffbase, and ffbase2 packages

3. data, table package

Working with big data in R requires efficient memory management and processing techniques. Here are some ways to work efficiently with big data in R:

1. Subsetting with base R and dplyr packages:

To subset a large dataset, we can use the base R functions like `[ ]` and `subset()` which allows us to extract specific rows and columns of a dataset. However, it may not be efficient for large datasets. Alternatively, we can use the `dplyr` package which provides functions like `select()`, `filter()`, `slice()`, `arrange()`, and `mutate()`. These functions work on lazy evaluation and create an execution plan that is optimized for speed and efficiency. For example, `filter()` only reads the relevant rows of data that meet a specified condition, while `select()` only reads the necessary columns.

2. ff, ffbase, and ffbase2 packages:

These packages can be used for big data that is too large to fit in memory. They provide file-based data structures, which can be manipulated as if they were in memory. `ff` creates a file-backed data frame that can handle data larger than memory, while `ffbase` provides tools for manipulating `ff` objects. `ffbase2` improves upon the previous versions of `ff` and `ffbase`.

3. data.table package:

The `data.table` package provides an efficient way to work with large datasets. It is a fast, memory-efficient data manipulation tool that provides an optimized syntax for common data manipulation operations. It provides functions like `setkey()`, `groupby()`, `join()`, `fread()`, and `fwrite()` which allow fast data manipulation and processing.

In summary, when working with big data in R, we can use a combination of efficient memory management techniques and optimized data manipulation packages to improve performance and processing speed.

**Explain social network analysis and describe its use in a real-life situation with**

1. Nodes

2. Links

3. Attributes

Social network analysis (SNA) is the process of analyzing social structures by examining the relationships between individuals or entities. SNA examines the patterns of relationships among people, organizations, or other entities to identify how information, resources, and other things flow among them.

In SNA, entities are represented as nodes, and the relationships between them are represented as links or edges. Nodes can be individuals, organizations, or any other entities, while links represent the connections or relationships between them. Attributes are additional information associated with nodes or links that provide context for the relationships and help in analyzing the network.

An example of the use of SNA is in studying the communication patterns within a large organization. Nodes would represent employees, while links would represent communication between them, such as email or instant messaging. Attributes could include job position, department, or location. By analyzing this network, researchers can identify which employees are central to the communication flow and which departments are more closely connected, leading to insights on how to improve communication and collaboration within the organization.

Group B

## Do the following in R Script

1. Define integers from 1 to 15 using different coding approaches

```
int1 <- 1:15

int2 <- seq(1, 15, by = 1)

int3 <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)
```

2. Define these five numbers: 1.1, 2.2, 3.3, 4.4 and 5.5 and save it as column vector N

```
N <- c(1.1, 2.2, 3.3, 4.4, 5.5)
```

3. Add, Subtract, multiply and divide the vector R from Vector N and interpret the results.

```
R <- c(1, 2, 3, 4, 5)

add <- N + R

subtract <- N - R

multiply <- N * R

divide <- N / R
```

4. Define a list using "This" "is" "my" "first" "programming" "in" "R" and save it as L

```
L <- list("This", "is", "my", "first", "programming", "in", "R")
```

5. Transform these list elements as characters of UL object

```
UL <- unlist(L)
```

OR

```
UL <- unlist(lapply(L, as.character))
```

**Import the "pollution.csv" file into R and do the following.**

1. Check the structure of the data and explain class of each variable

```
setwd("path/to/folder")
# Import the dataset
pollution <- read.csv("pollution.csv", header = TRUE, sep = ",")

str(pollution)
```

2. Change the attributes of "particular matter", "data time" and "value" variables

```
pollution$Particular.Matter <- factor(pollution$Particular.Matter)
pollution$Data.Time <- as.POSIXct(pollution$Data.Time, format = "%Y-%m-%d %H:%M:%S")

# Change the attribute of "value" variable
pollution$Value <- as.numeric(pollution$Value)
```

3. Get the summary of all the variables and replace the outlier as missing value.

```
summary(pollution)
# Replace outliers as missing values
pollution$Value[pollution$Value > 300] <- NA
```

4. Get summary of statistics of "value" variables by "particular matter" variable categories

aggregate(Value ~ Particular.Matter, pollution, summary)

5. Write a summary of the results obtained in the earlier steps with interpretation and conclusion

We have imported the "pollution.csv" file and checked its structure to get an overview of the dataset. We have then changed the attributes of "particular matter", "data time" and "value" variables to match their respective data types. Next, we have summarized all the variables and replaced the outliers as missing values. Finally, we have obtained the summary of statistics of "value" variable by "particular matter" variable categories.

From the summary statistics, we can see that the "particular matter" variable has three categories: "PM10", "PM2.5" and "NO2". We can also see that the "value" variable has a minimum value of 0 and a maximum value of 299, with some outliers above 300, which we have replaced with missing values. We have also observed that the mean and median values of "value" variable differ by the "particular matter" categories. For instance, PM2.5 has a higher mean and median value compared to PM10 and NO2. Therefore, we can conclude that the PM2.5 pollutant is more prevalent in the dataset and requires further investigation to understand its impact on the environment and human health.

**Use the "pollution.csv" file imported and cleaned in R Studio and do as follows**

```
# load the data
pollution <- read.csv("pollution.csv")

1. Create bar plot of "particular matter" variable.
barplot(table(pollution$particular_matter),
        main = "Bar Plot of Particular Matter",
        xlab = "Particular Matter",
        ylab = "Count")
```

2. Create histogram of "value" variable.

```
hist(pollution$value,
     main = "Histogram of Value",
     xlab = "Value",
     ylab = "Frequency")
```

3. Create line plot of "date time" and "value" variables.

```
plot(pollution$date_time, pollution$value,
     type = "l",
     main = "Line Plot of Date Time and Value",
     xlab = "Date Time",
     ylab = "Value")
```

4. Create histogram of " value" variable by particular matter categories

```
par(mfrow=c(2,2))
for (i in unique(pollution$particular_matter)) {
  hist(pollution$value[pollution$particular_matter == i],
       main = paste("Histogram of Value for", i),
       xlab = "Value",
       ylab = "Frequency")
}
```

5. Write a summary of the results obtained in the earlier steps with interpretation and conclusion

We have created a bar plot of "particular matter" variable to show the count of each category. The histogram of "value" variable shows the frequency distribution of the variable. The line plot of "date time" and "value" variables shows the trend of pollution over time. The histogram of "value" variable by particular matter categories shows the frequency distribution of the variable for each particular matter category. These plots help us to understand the characteristics of the data, and we can use this information to make informed decisions.

**Load the "term Doc Matrix R data" file into R Studio and do as follows**

1. Define the term document matrix data object as matrix and store it as "m" object

```
m <- as.matrix(your_tdm_object)
```

2. Define the frequencies of the term using "row Sums" function and get the term frequencies

```
term_freq <- rowSums(m)
```

3. Create a histogram of the term frequencies using ggplot2 package

```
library(ggplot2)
ggplot(data.frame(term_freq), aes(x=term_freq)) +
  geom_histogram(binwidth=1000) +
  ggtitle("Term Frequency Histogram")
```

4. Create a histogram of the term with 10 or more frequencies using ggplot2 package

```
freq_above_10<- term_freq[term_freq >= 10]
ggplot(data.frame(freq_above_10), aes(x=freq_above_10)) +
  geom_histogram(binwidth=1) +
  ggtitle("Term Frequency (>=10) Histogram")
```

5. Create a word cloud of the term frequencies using word cloud package and interpret it carefully

```
library(wordcloud)
wordcloud(names(term_freq), freq=term_freq, min.freq=10, random.order=F,
        colors=brewer.pal(8, "Dark2"))
```

Interpretation of the word cloud will depend on the specific data set and context. However, generally, larger words indicate more frequent terms in the document matrix.

**Load the "rdm Tweets r data" file in R studio and do as follows with "tm" and "tweetR" packages:**

```
library(tm)
library(tweetR)

# Load the data
load("rdm_Tweets.rdata")

1. Convert twitter list as data frame and assign it as "df" object
df <- twListToDF(rdmTweets)

2. Create corpus using the "text" column of the data frame
corpus <- Corpus(VectorSource(df$text))

3. Perform Pre-Processing to clean the corpus for text mining
corpus <- tm_map(corpus, tolower) # Convert to lowercase
corpus <- tm_map(corpus, removePunctuation) # Remove punctuation
corpus <- tm_map(corpus, removeNumbers) # Remove numbers
corpus <- tm_map(corpus, removeWords, stopwords("english")) # Remove stop words
corpus <- tm_map(corpus, stripWhitespace) # Strip whitespace

4. Create term document matrix using the cleaned corpus
tdm <- TermDocumentMatrix(corpus, control = list(wordLengths=c(1,Inf)))

5. Find the most frequent terms using the term document matrix
freq_terms <- findFreqTerms(tdm, lowfreq=10)

6. Find the co-occurrence of the term "r" with filter of 0.1 and above
co_occur <- findAssocs(tdm, "r", 0.1)
```

**Load the igraph package in R and do the basic SNA as follows**

1. Define g as graph object with (1,2) as its elements

```
library(igraph)
g <- graph(edges = c(1, 2))
```

2. Plot the g and interpret it carefully

```
plot(g)
```

The plot will show two nodes with a single link connecting them.

3. Define g1 as graph object with ("S", "R", "R", "G", "G", "S", "S", "G", "A", "R") as its elements.

```
g1 <- graph_from_literal(S -+ R -+ G -+ S -+ G -+ A -+ R -+ G, R -+ S -+ G)
```

OR

```
# create an empty graph
g1 <- graph()
# add the vertices
V(g1) <- c("S", "R", "G", "A")

# add the edges
E(g1) <- c("S", "R", "R", "G", "G", "S", "S", "G", "A", "R", "R", "S", "S", "G")
```

4. Plot g1 with node color as green, node size as 30, link color as red and link size as 5 and interpret it

```
plot(g1, vertex.color="green", vertex.size=30, edge.color="red", edge.width=5)
```

The plot will show five nodes connected by seven links. The nodes are labeled with the letters S, R, G, and A, and their sizes and colors reflect the specified parameters.

5. Get degree, closeness and betweenness of g1 and interpret them

```
degree(g1)
closeness(g1)
betweenness(g1)
```

**Load the "term Doc Matrix R data" file into R studio and do**

```
# Load the data
load("term Doc Matrix R data.RData")
```

1. Define term Doc Matrix as matrix m

```
m <- as.matrix(term_doc_matrix)
```

2. Transform it into adjacency matrix

```
adj_matrix <- ifelse(m > 0, 1, 0)
```

3. Build a undirected SNA graph with adjacency matrix data

```
library(igraph)
g <- graph.adjacency(adj_matrix, mode = "undirected")
```

4. Remove loops and plot the SNA graph again

```
g <- simplify(g)
plot(g, vertex.label=V(g)$name)
```

5. Interpret all the results carefully

Here, we first load the term document matrix data and convert it to a matrix object. Then, we transform the matrix into an adjacency matrix where the values greater than zero are set to 1, indicating the presence of an edge between two terms. We use this adjacency matrix to create an undirected graph using the graph.adjacency() function from the igraph package. We then remove any loops in the graph and plot it. We can interpret the graph by analyzing the nodes and edges, and their connections. The graph can be used to identify the most frequent terms and their relationships in the document corpus.

Final Exam 2078

# Group B

**Do the following in R Studio**

1. Prepare a column vector of miles per gallon (mpg) variable with random range between 10 to 50 of 500 values.

```
set.seed(123)  # Set seed for reproducibility

mpg <- runif(500, 10, 50)
```

2. Plot histogram of this "mpg" variable and interpret it carefully

```
hist(mpg)
```

The histogram shows the distribution of the mpg variable. The x-axis represents the range of the variable, and the y-axis represents the frequency or count of the values falling in each bin.

3. Refine the histogram by filling the bars with "blue" color and changing number of bins to 8

```
hist(mpg, col = "blue", breaks = 8)
```

The refined histogram has blue-colored bars and 8 bins, providing a clearer view of the distribution of the mpg variable.

4. Add a vertical abline at the arithmetic mean of the mpg variable

```
hist(mpg, col = "blue", breaks = 8)

abline(v = mean(mpg), col = "red", lwd = 2)
```

The red line on the histogram represents the arithmetic mean of the mpg variable.

5. Plot Q-Q plot of mpg variable, add normal Q-Q line of red color on it and interpret it carefully

```
qqnorm(mpg)

qqline(mpg, col = "red", lwd = 2)
```

The Q-Q plot is used to check if the data follows a normal distribution. If the data points fall along the red line, it indicates that the data follows a normal distribution. In this case, we can see that the data points are relatively close to the red line, indicating that the mpg variable may follow a normal distribution.

6. Plot density plot of mpg variable without the border, fill it with yellow color and interpret it

```
plot(density(mpg), col = "yellow", lwd = 3, border = NA)
```

The density plot provides a smooth representation of the distribution of the mpg variable. The yellow color filling shows the density of the variable at each point along the range of values. The plot indicates that the mpg variable has a bimodal distribution, with two peaks around 20 and 35.

**Use ggplot2 package and do as follows**

```
library(ggplot2)
```

1. Define first layer of the ggplot object with diamond data, carat as x-axis, and price as y-axis

```
p <- ggplot(diamonds, aes(x = carat, y = price))
```

2. Add layer with geometric aesthetic as "point", statistics and position as "identity."

```
p <- p + geom_point(stat = "identity")
```

3. Add layer with scale of y and x variables as continuous

```
p <- p + scale_x_continuous() + scale_y_continuous()
```

4. Add layer with with coordinate system as Cartesian

```
p <- p + coord_cartesian()
```

5. Add layer with appropriate title and interpret the resulting graph

```
p <- p + ggtitle("Diamonds Price by Carat")
```

```
# print the plot
print(p)
```