Names with IDs:

Team 79

جوسام هاني فؤاد 23010095
فادي يسري نجيب 23011407
مينا كيرلس منصور 23011577
شعيب وليد محمد 23012121

Team Roles:

[1] Josam -> Visualizations and Associations.
[2] Fady -> UI
[3] Mina -> Cleaning
[4]Shoaib -> Clustering

Data Set Description:

[1]Item and Quantity: Each entry details the items bought in a single transaction, along with the respective quantities. This facilitates the examination of popular products and purchasing trends.

[2]Total Transaction Cost: This represents the overall expenditure for the transaction, encompassing the cumulative cost of all items, factoring in their quantities. It allows for insights into spending habits and revenue evaluation.

[3]Transaction Identifier: An exclusive identification code assigned to each transaction. This identifier is critical for distinguishing between individual purchases and ensuring the integrity of data analysis.

[4]Customer Name: The designation of the purchasing individual. This field enables personalized analysis and outcomes, albeit raising considerations regarding privacy and data security.

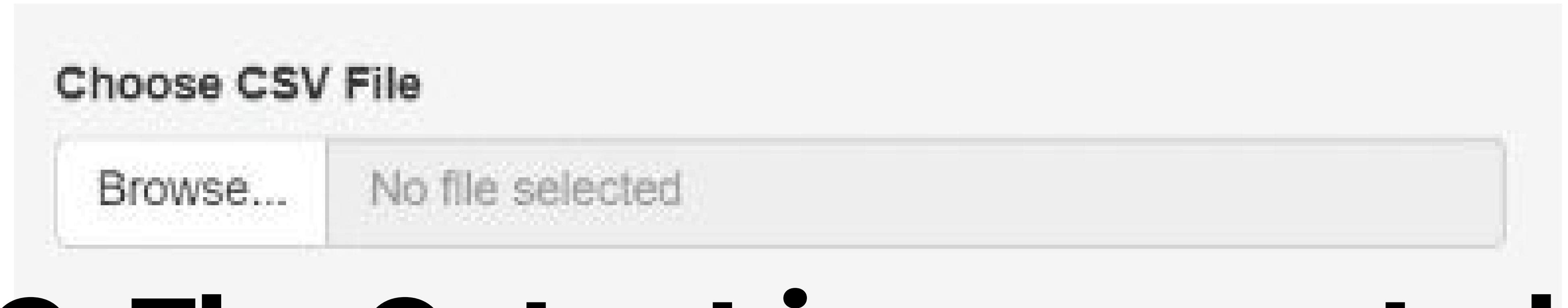
[5]Customer Age: The age of the customer, providing valuable insights for targeted marketing approaches.

[6]Customer Location: The city of residence for the customer. This geographical data serves as a basis for regional analysis and location-centric marketing strategies.

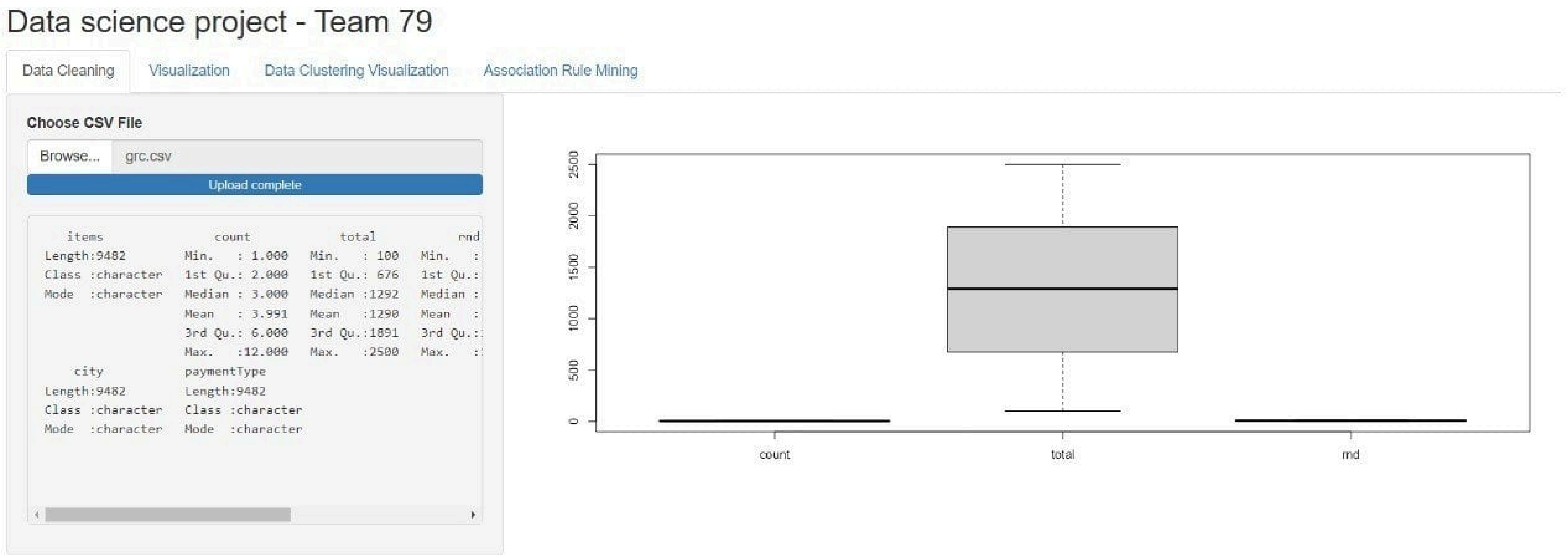
[7]Payment Method: Specifies whether the transaction was settled via cash or credit. This attribute is fundamental for dissecting payment preferences and financial behavior across various customer segments.

A- The Program is going to Clean the provided data

B- The Input would be file.csv



C- The Output is represented as a file of cleaned data



```
   items              count            total          rnd
 Length:9482        Min.   : 1.000   Min.   : 100   Min.   :
 Class :character   1st Qu.: 2.000   1st Qu.: 676   1st Qu.:
 Mode  :character   Median : 3.000   Median :1292   Median :
                    Mean   : 3.991   Mean   :1290   Mean   :
                    3rd Qu.: 6.000   3rd Qu.:1891   3rd Qu.:
                    Max.   :12.000   Max.   :2500   Max.   :
     city           paymentType
 Length:9482        Length:9482
 Class :character   Class :character
 Mode  :character   Mode  :character
```

[0] Libraries

Shiny library : Enables you to build interactive GUI straight from R.

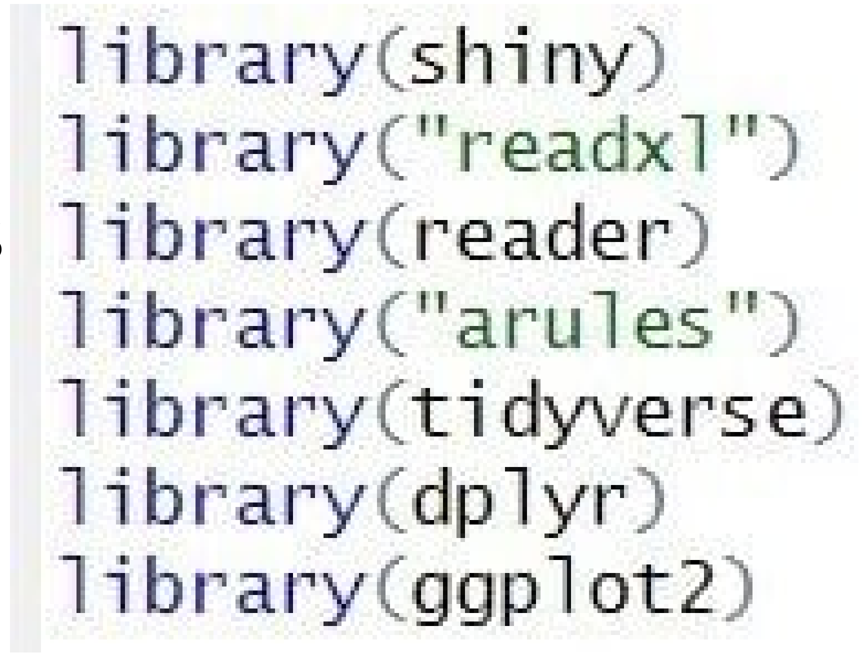
Arules library : Used for mining association rules and frequent item sets. This is useful in market basket analysis.

Dplyr library : This package is essential for data manipulation tasks such as filtering rows, selecting columns, rearranging data, and performing summaries with ease.

Readr library : A package that provides an easy way to read CSV, which is particularly useful for extracting data from spreadsheets directly into R.

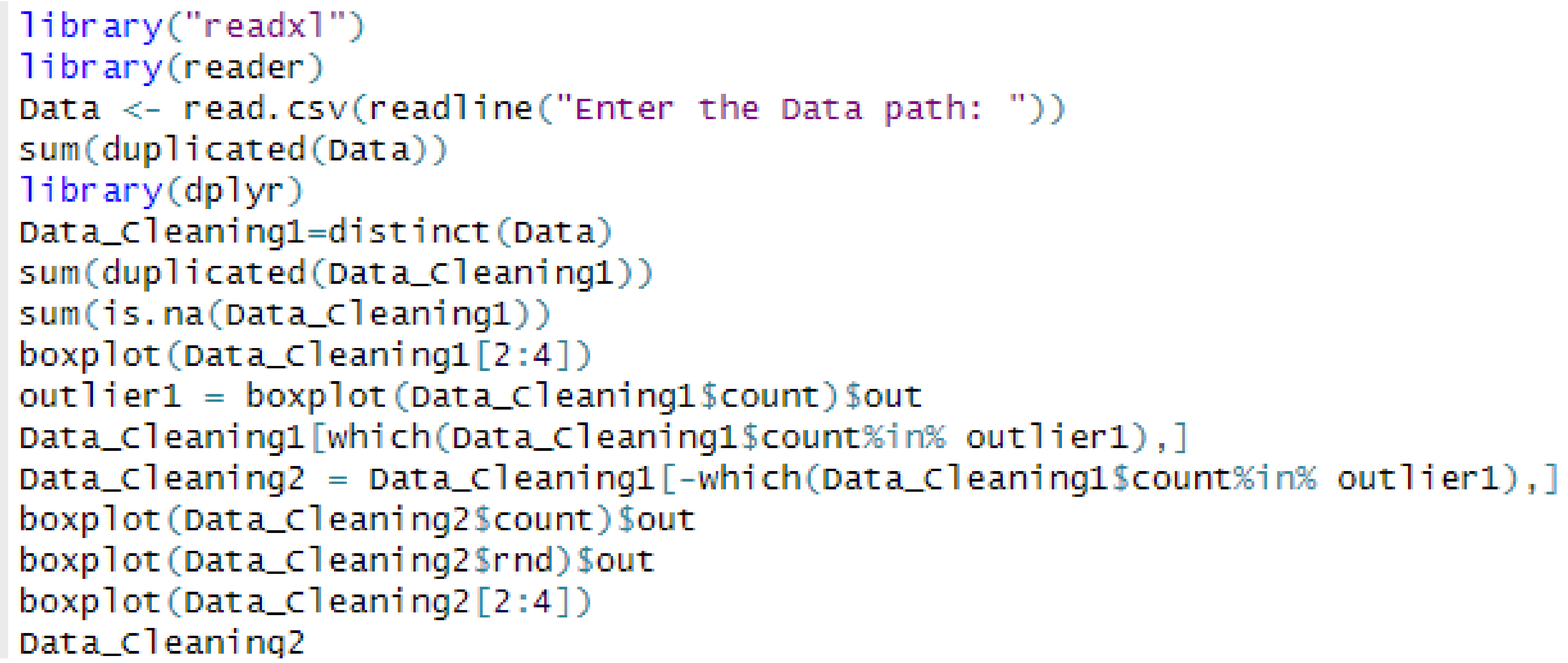
Tidyverse: to group data by age or cities and calculate totals.

Ggplot2 library : Its used for data visualization, it allows for the creation of complex plots from data in a data frame with a powerful and flexible system based on the grammar of graphics.



```
library(shiny)
library("readxl")
library(reader)
library("arules")
library(tidyverse)
library(dplyr)
library(ggplot2)
```

[1] Cleaning



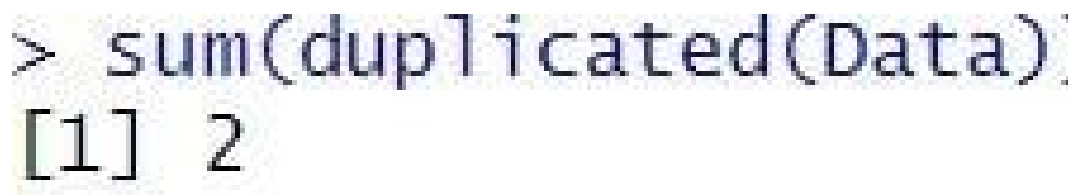
```
library("readxl")
library(reader)
Data <- read.csv(readline("Enter the Data path: "))
sum(duplicated(Data))
library(dplyr)
Data_Cleaning1=distinct(Data)
sum(duplicated(Data_Cleaning1))
sum(is.na(Data_Cleaning1))
boxplot(Data_Cleaning1[2:4])
outlier1 = boxplot(Data_Cleaning1$count)$out
Data_Cleaning1[which(Data_Cleaning1$count%in% outlier1),]
Data_Cleaning2 = Data_Cleaning1[-which(Data_Cleaning1$count%in% outlier1),]
boxplot(Data_Cleaning2$count)$out
boxplot(Data_Cleaning2$rnd)$out
boxplot(Data_Cleaning2[2:4])
Data_Cleaning2
```

[1]Loading Libraries: The code starts by loading the "readxl", "reader", and "dplyr" libraries. These packages provide functions for reading Excel files, handling data frames, and data manipulation tasks.

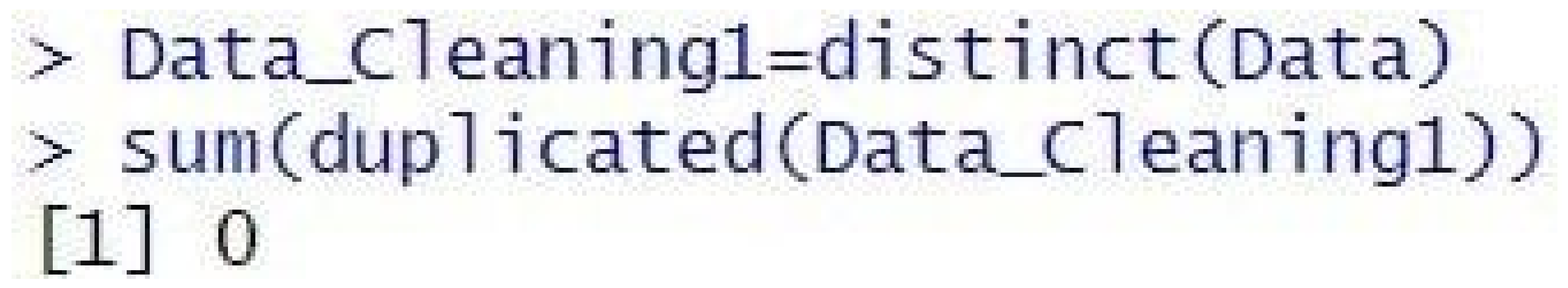
[2]Reading Data: It prompts the user to enter the path for the data file, then reads the data using read.csv() function.

[4]Removing Duplicates: It removes duplicated rows from the dataset using distinct(Data), and assigns the result to Data_Cleaning1.

[3]Identifying Duplicates: It checks for duplicated rows in the dataset using sum(duplicated(Data)), where Data is the name of the dataset.



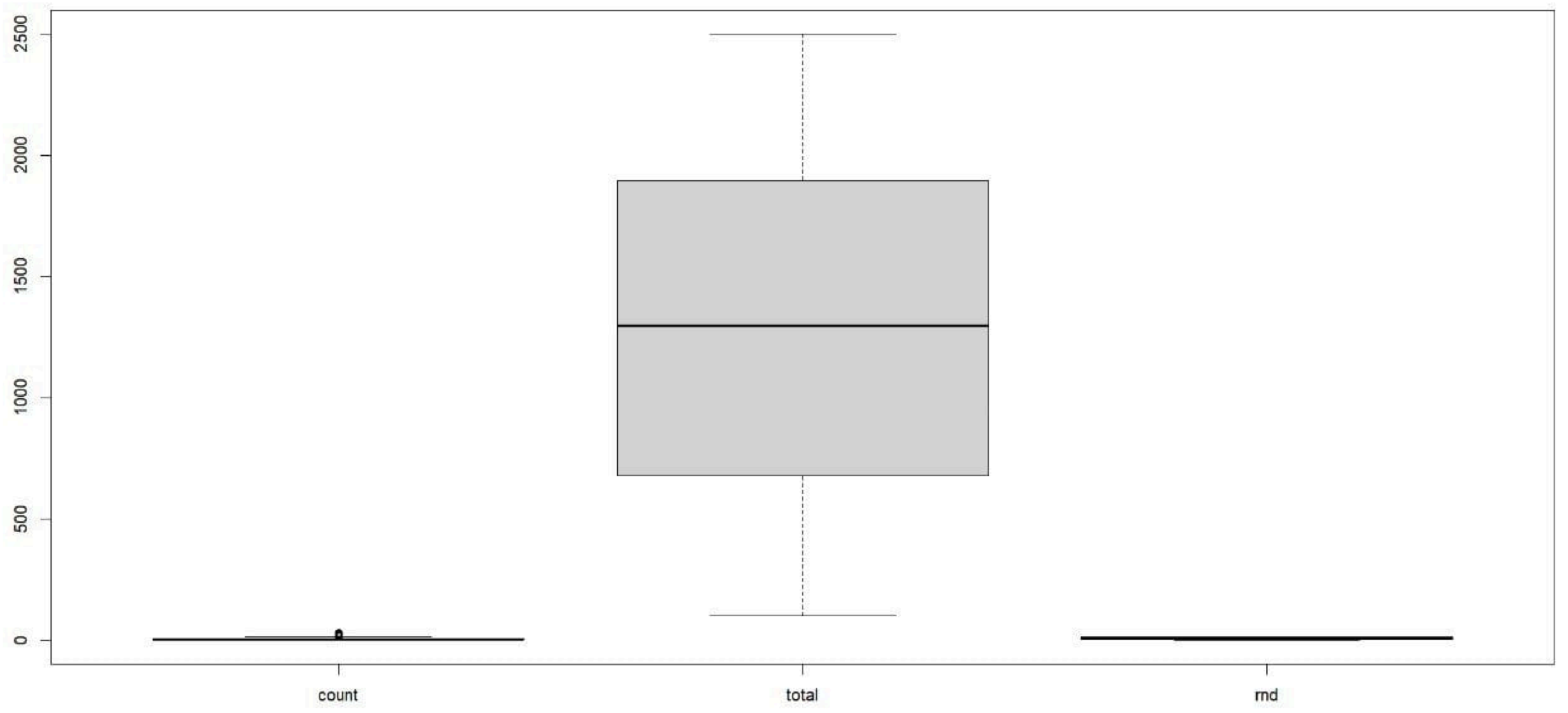
```
> sum(duplicated(Data)
[1] 2
```



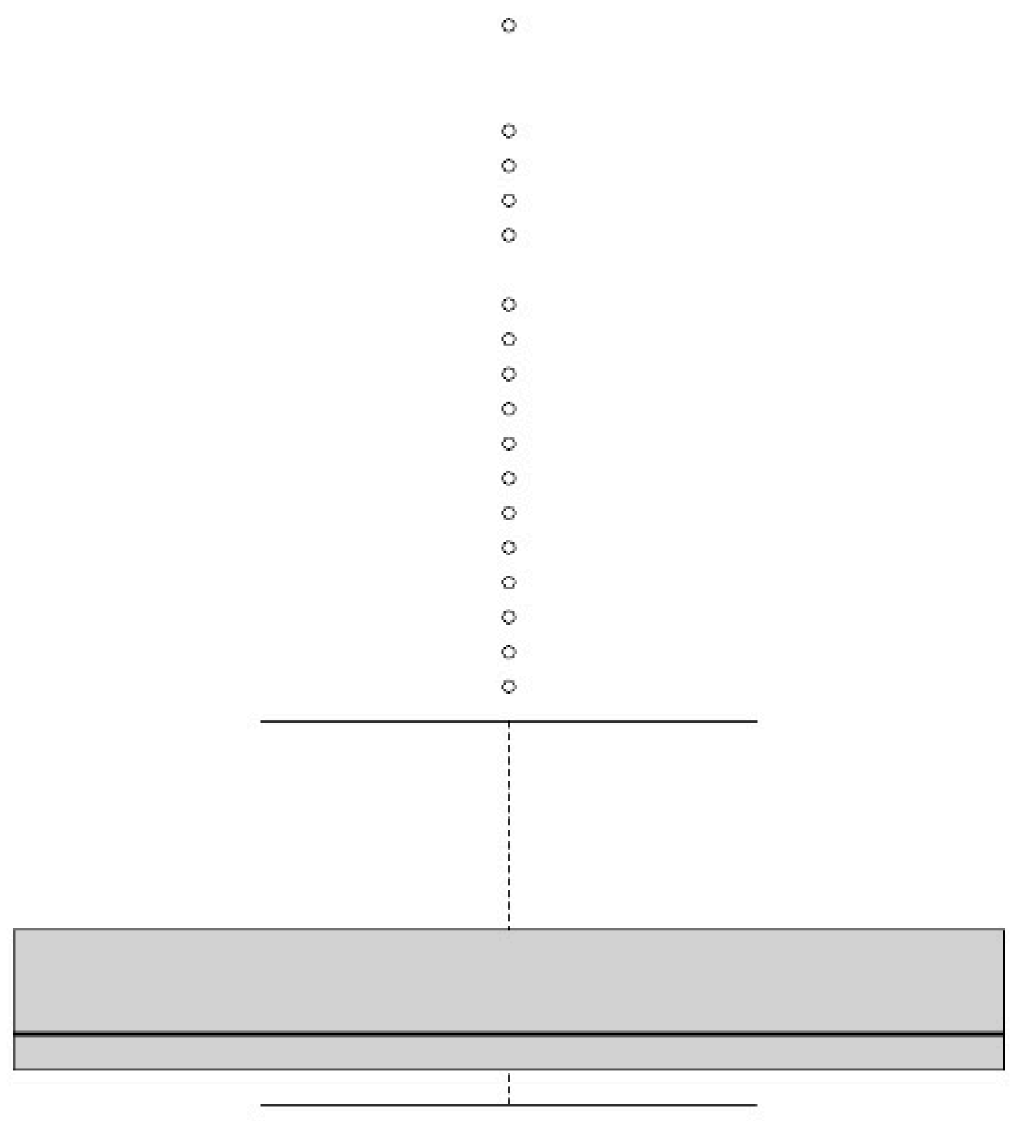
```
> Data_Cleaning1=distinct(Data)
> sum(duplicated(Data_Cleaning1))
[1] 0
```

[5]Checking for Missing Values: It counts the number of missing values in Data_Cleaning1 using sum(is.na(Data_Cleaning1)).

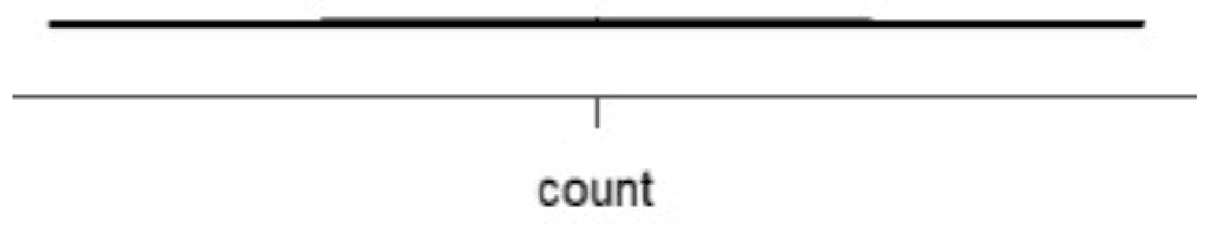
[6]Visualizing Data: It creates boxplots for columns 2 to 4 in the cleaned dataset Data_Cleaning1 to identify outliers.



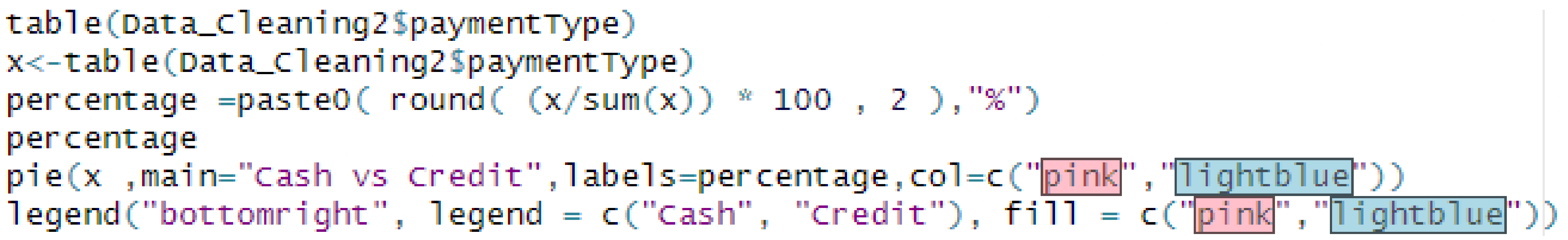
[7]Handling Outliers: It detects outliers for the column named "count" using boxplot(Data_Cleaning1$count)$out, and removes



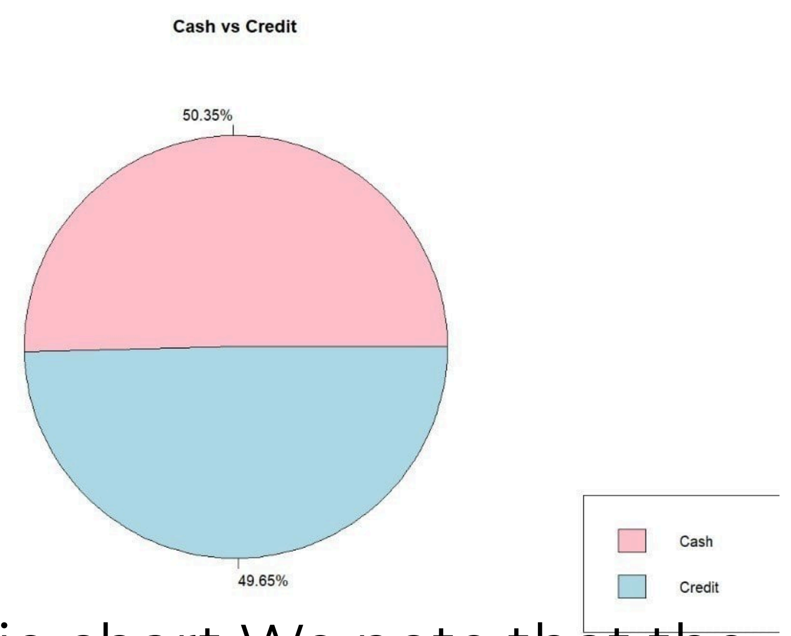
[8]Further Outlier Detection: It creates boxplots for "count" and "rnd" columns separately in the cleaned dataset Data_Cleaning2, then creates a boxplot for columns 2 to 4 collectively.



[2]Visualization



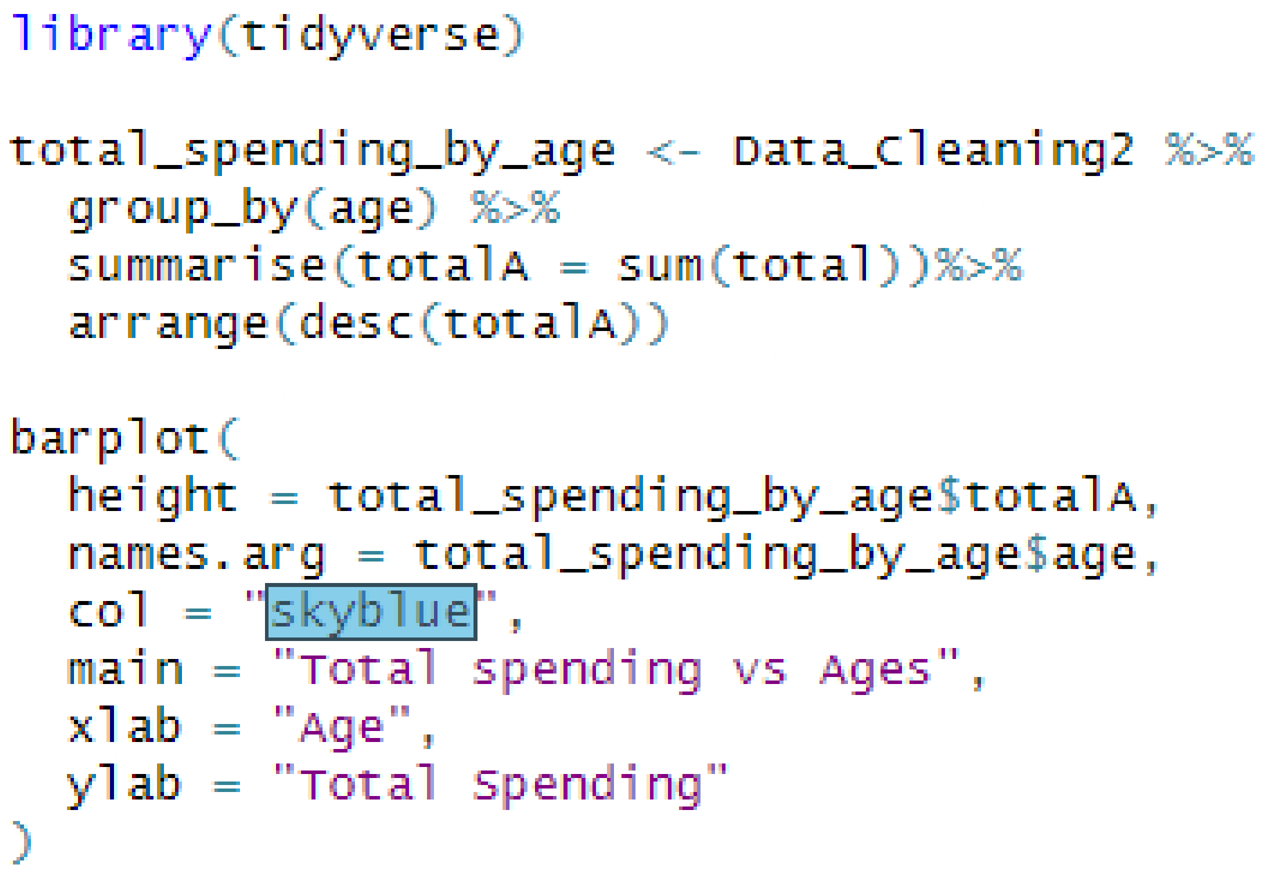
```
table(Data_Cleaning2$paymentType)
x<-table(Data_Cleaning2$paymentType)
percentage =paste0( round( (x/sum(x)) * 100 , 2 ),"%")
percentage
pie(x ,main="Cash vs Credit",labels=percentage,col=c("pink","lightblue"))
legend("bottomright", legend = c("Cash", "Credit"), fill = c("pink","lightblue"))
```



[1]This code generates a pie chart to visualize the distribution of payment types (cash vs. credit) in the cleaned dataset, displaying the percentages of each type.

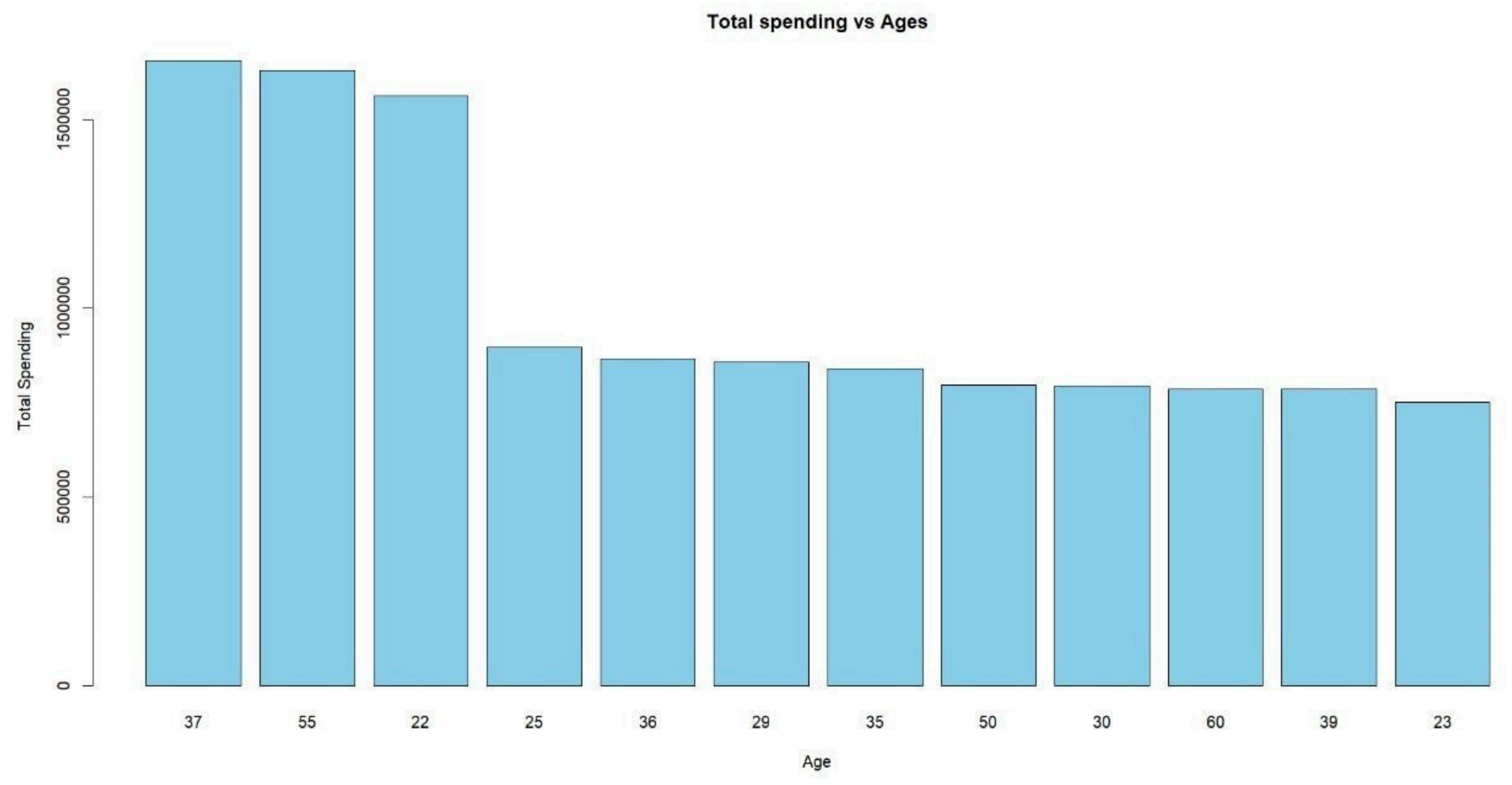
from pie chart We note that the cash payment method has a slightly higher rate than payment via credit

---



```
library(tidyverse)

total_spending_by_age <- Data_Cleaning2 %>%
  group_by(age) %>%
  summarise(totalA = sum(total))%>%
  arrange(desc(totalA))

barplot(
  height = total_spending_by_age$totalA,
  names.arg = total_spending_by_age$age,
  col = "skyblue",
  main = "Total spending vs Ages",
  xlab = "Age",
  ylab = "Total Spending"
)
```

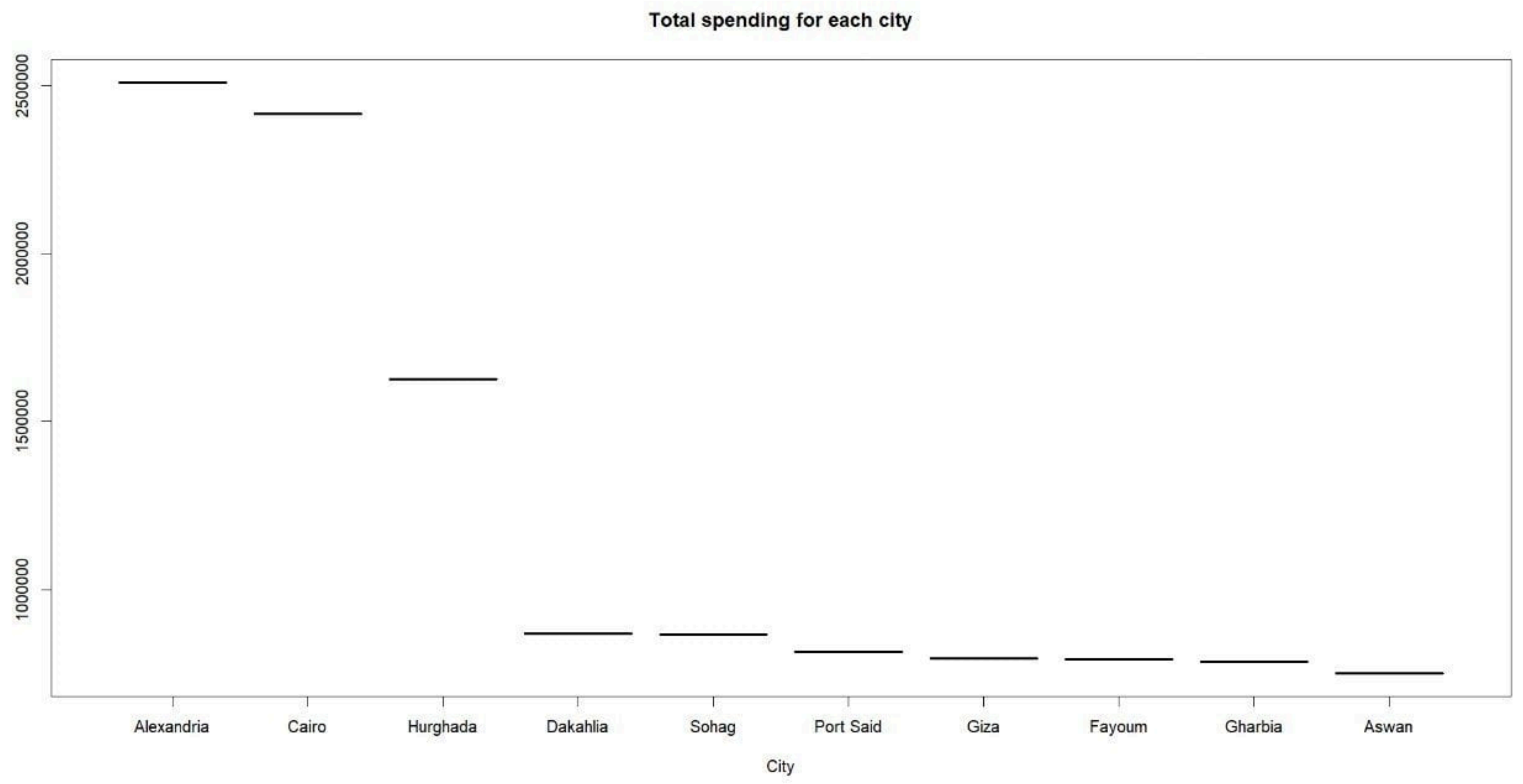
[2]This code uses the tidyverse package to calculate the total spending grouped by age from the cleaned dataset, then creates a bar plot to visualize the total spending versus different age groups.



From the barplot we can note that : the higher of total spending is People who are 37 and the least is 23 years old

```
total_spending_by_city <- Data_Cleaning2 %>%
  group_by(city) %>%
  summarise(totalc = sum(total)) %>%
  arrange(desc(totalc))

plot(reorder(total_spending_by_city$city, -total_spending_by_city$totalc),
     total_spending_by_city$totalc,
     xlab = "City",
     ylab = "Spending",
     main = "Total spending for each city")
```

[3]first calculates the total spending for each city from a dataset called, then arranges the cities in descending order of total spending. Finally, it creates a plot where cities are reordered based on their total spending, with spending values represented on the y-axis and cities on the x-axis, labeled accordingly.



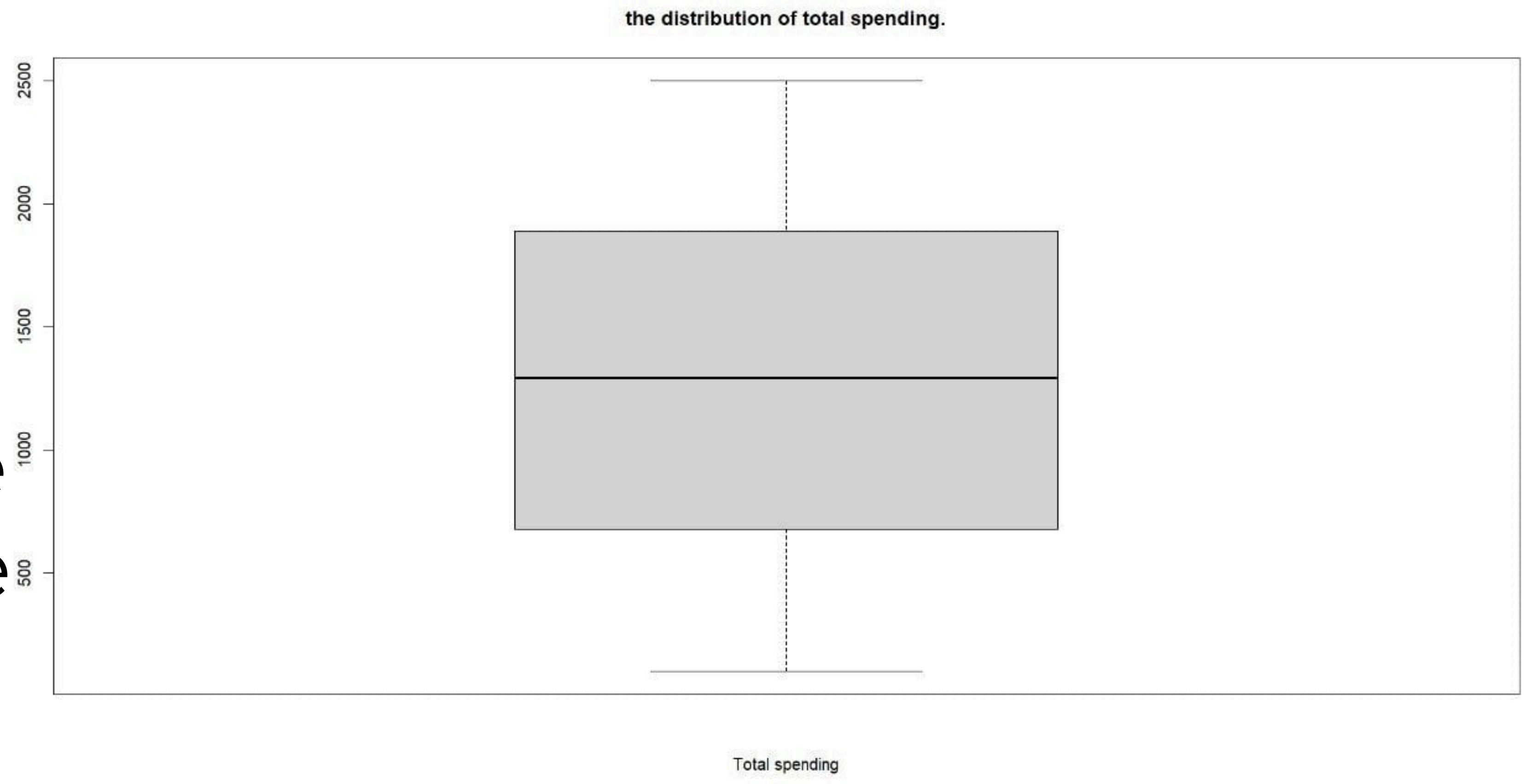
From that plot we can note that: the higher of total spending is Alexandria and the least is Aswan, Arranged in descending order

---

```
boxplot(Data_Cleaning2$total, main=" the distribution of total spending. ",
        xlab="Total spending")
```

[4]This code generates a boxplot to visualize the distribution of total spending from the cleaned dataset Data_Cleaning2, with a title indicating the purpose and a label for the x-axis specifying "Total spending".

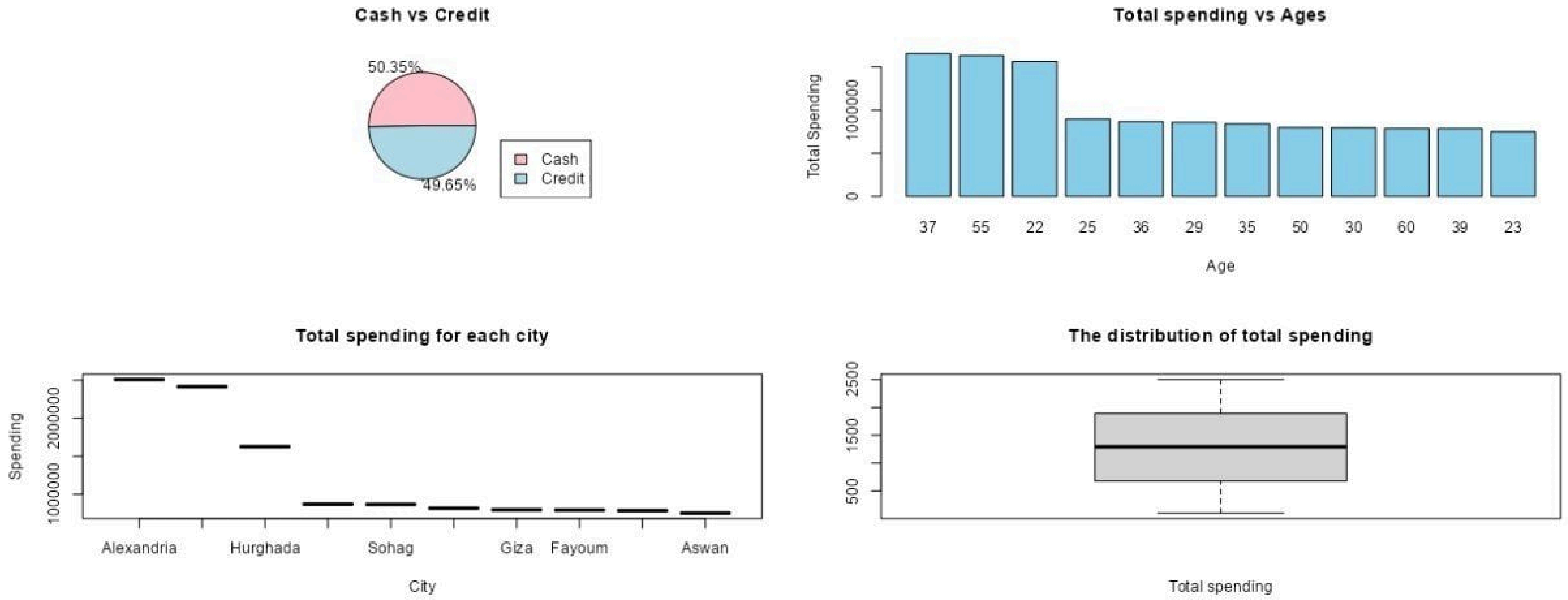
We notice that the peak concentration of the data's value spans from 500 to 2000, with the median falling between 1000 and 1500.



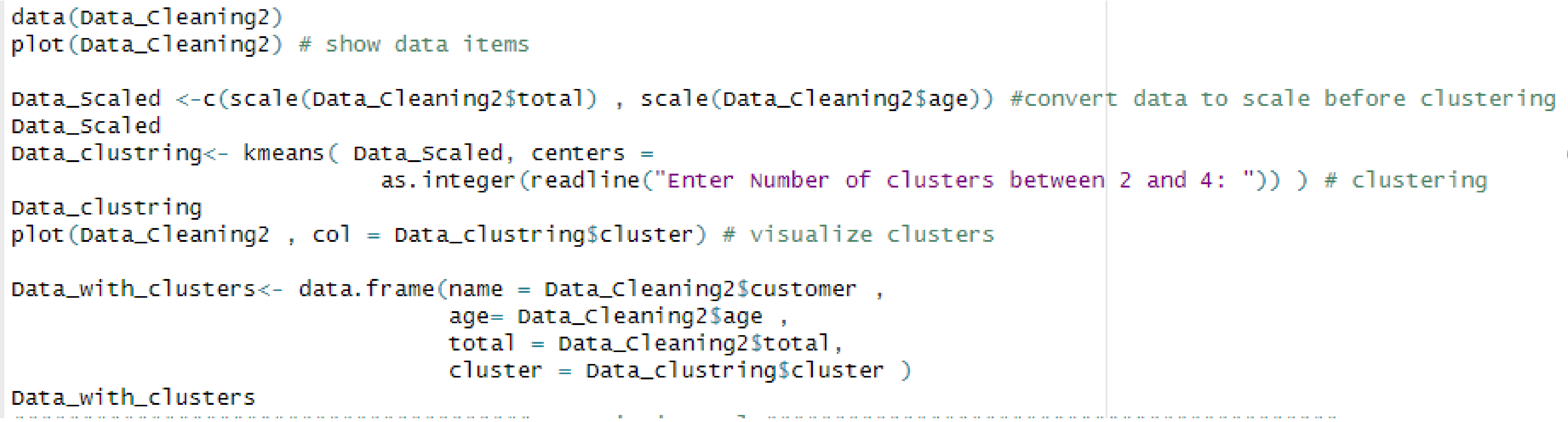
[5] DashBoard

```
>  par(mfrow=c(2,2))
```

[5]This code sets up the plotting layout to display multiple plots in a grid format with 2 rows and 2 columns using the par() function.

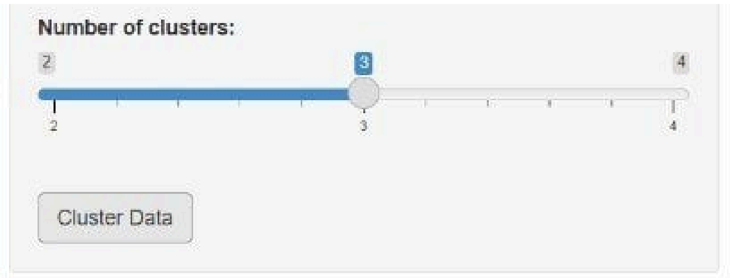


[3]Clustering With UI

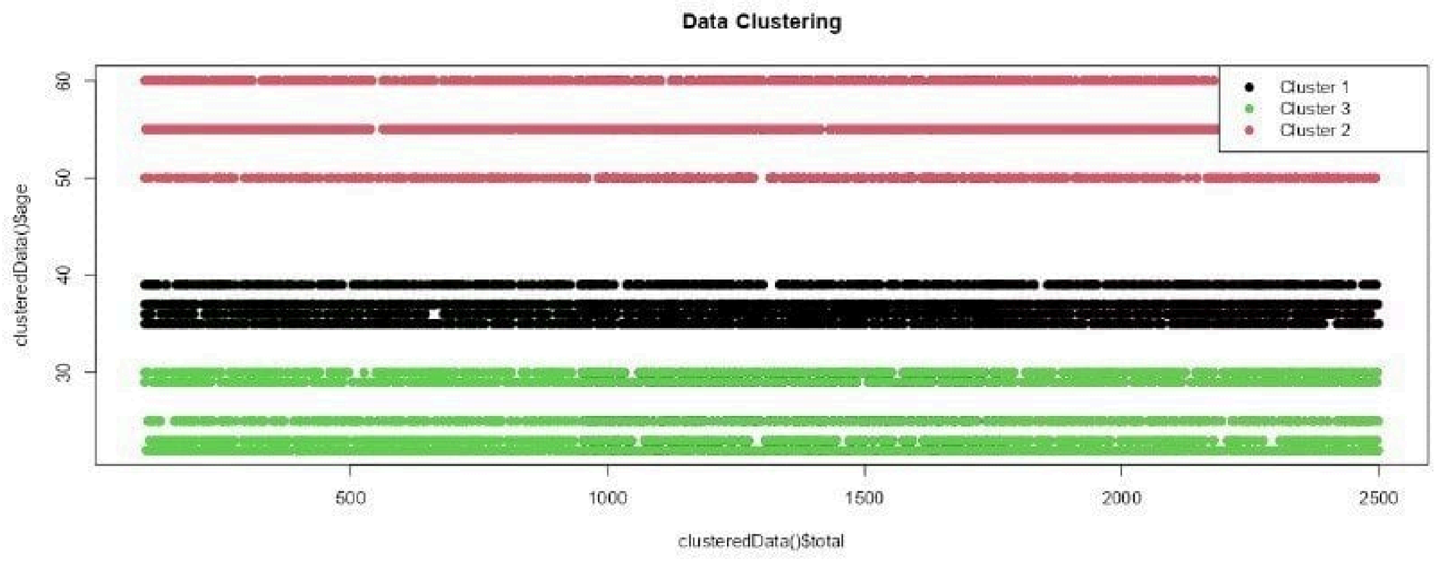


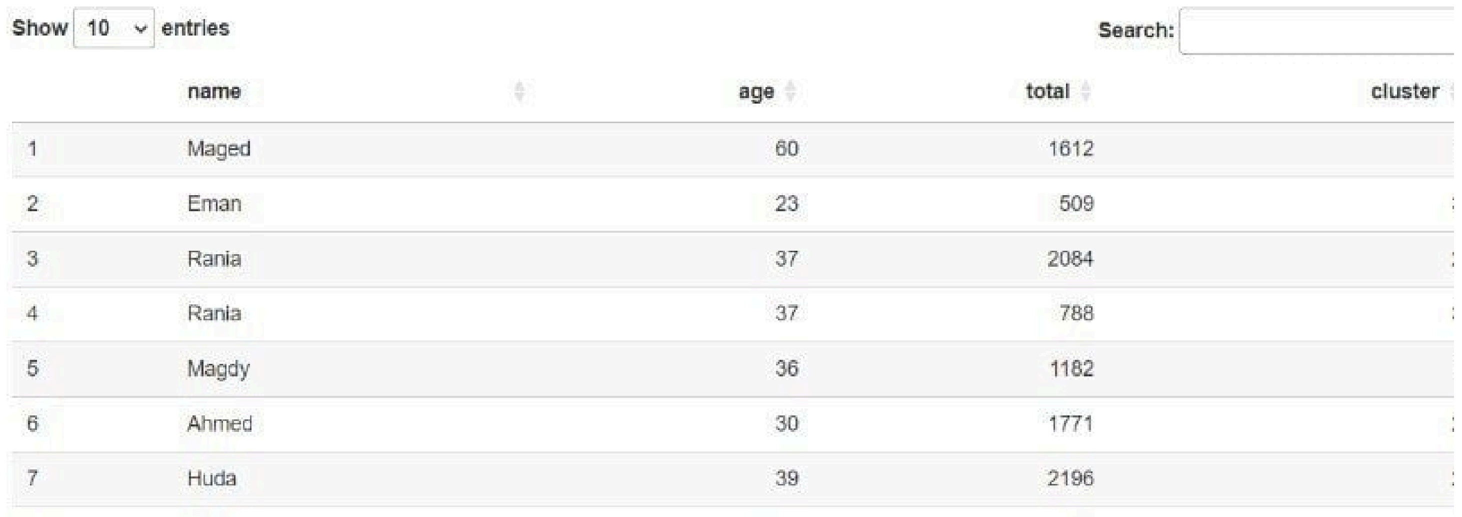
```
data(Data_Cleaning2)
plot(Data_Cleaning2) # show data items

Data_Scaled <-c(scale(Data_Cleaning2$total) , scale(Data_Cleaning2$age)) #convert data to scale before clustering
Data_Scaled
Data_clustring<- kmeans( Data_Scaled, centers =
                         as.integer(readline("Enter Number of clusters between 2 and 4: ")) ) # clustering
Data_clustring
plot(Data_Cleaning2 , col = Data_clustring$cluster) # visualize clusters

Data_with_clusters<- data.frame(name = Data_Cleaning2$customer ,
                                age= Data_Cleaning2$age ,
                                total = Data_Cleaning2$total,
                                cluster = Data_clustring$cluster )
Data_with_clusters
```

First loads and plots the data in Data_Cleaning2. Then, it scales the data for clustering, performs k-means clustering with a user-defined number of clusters, visualizes the clusters on the original data plot, and finally, creates a new dataframe Data_with_clusters including customer names, age, total spending, and assigned cluster labels.



The Input is going to be the number of clusters and the Output is the categorized data

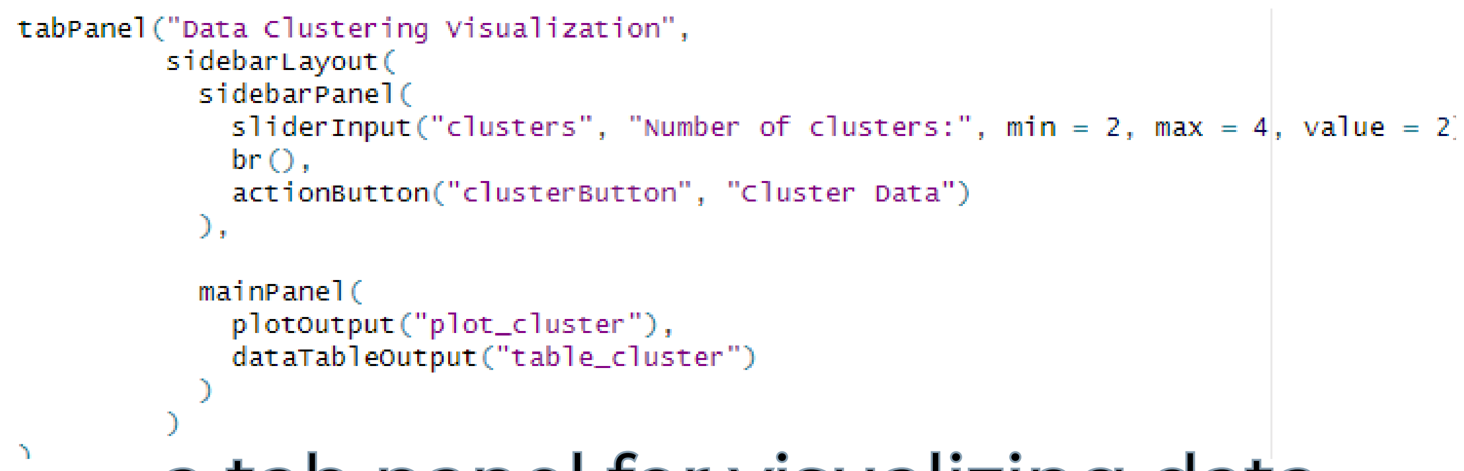




Show 10 entries Search:

	name	age	total	cluster
1	Maged	60	1612	
2	Eman	23	509	
3	Rania	37	2084	
4	Rania	37	788	
5	Magdy	36	1182	
6	Ahmed	30	1771	
7	Huda	39	2196	

UI

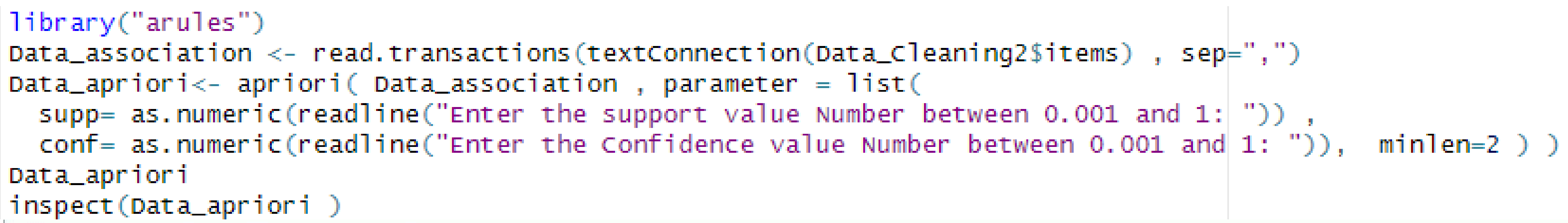


```
tabPanel("Data Clustering Visualization",
         sidebarLayout(
           sidebarPanel(
             sliderInput("clusters", "Number of clusters:", min = 2, max = 4, value = 2
             br(),
             actionButton("clusterButton", "Cluster Data")
           ),

           mainPanel(
             plotOutput("plot_cluster"),
             dataTableOutput("table_cluster")
           )
         )
)
```

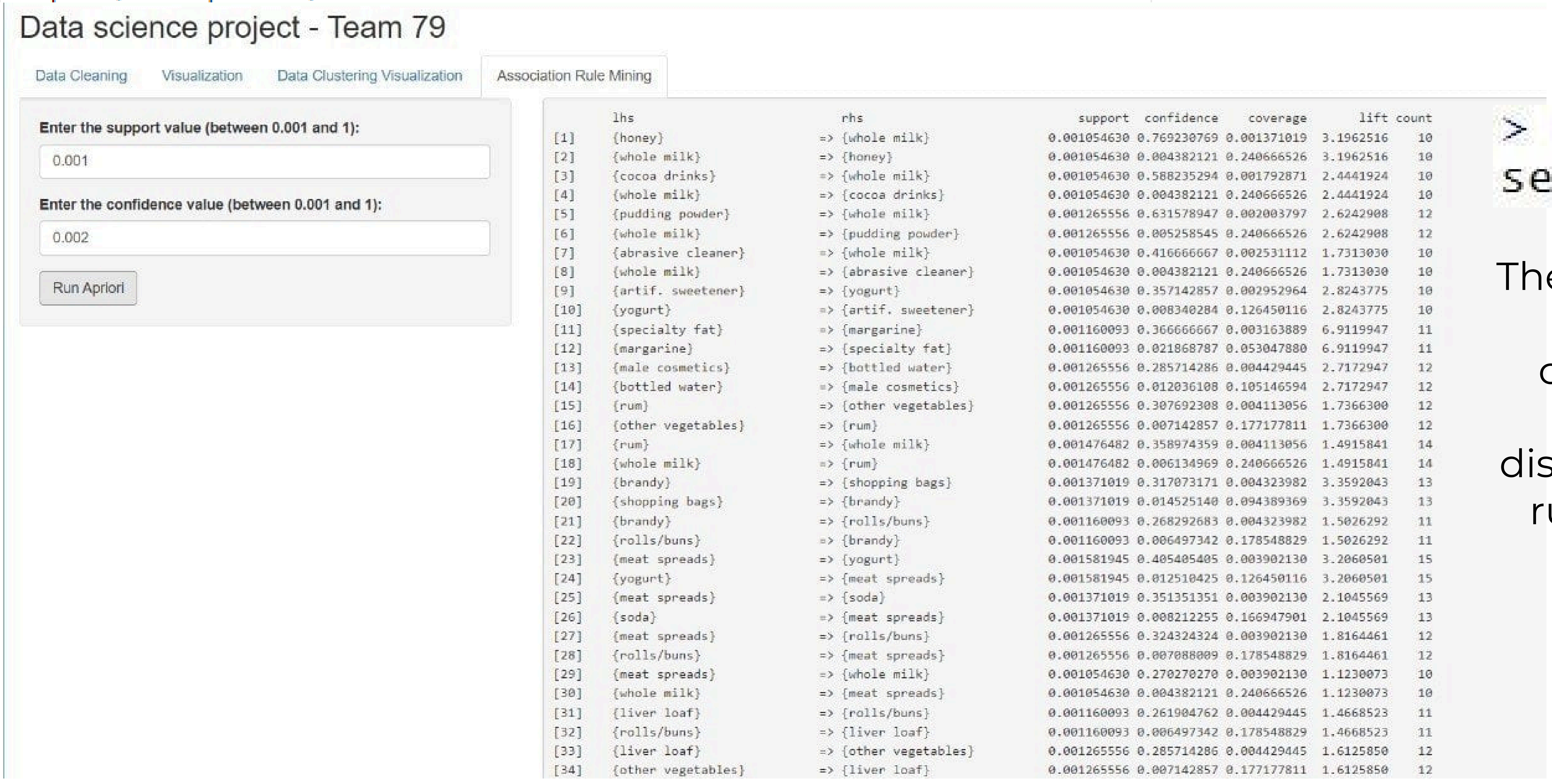
a tab panel for visualizing data clustering, comprising a sidebar with a slider for selecting the number of clusters and a button to initiate clustering. The main panel contains outputs for displaying the clustering plot and a data table.

[4]Association rule

reads transaction data from, converts it into transaction format, and then applies the Apriori algorithm to find association rules based on user-defined support and confidence thresholds. Finally, it inspects and displays the discovered association rules.



```
library("arules")
Data_association <- read.transactions(textConnection(Data_Cleaning2$items) , sep=",")
Data_apriori<- apriori( Data_association , parameter = list(
  supp= as.numeric(readline("Enter the support value Number between 0.001 and 1: ")) ,
  conf= as.numeric(readline("Enter the Confidence value Number between 0.001 and 1: ")),  minlen=2 ) )
Data_apriori
inspect(Data_apriori )
```

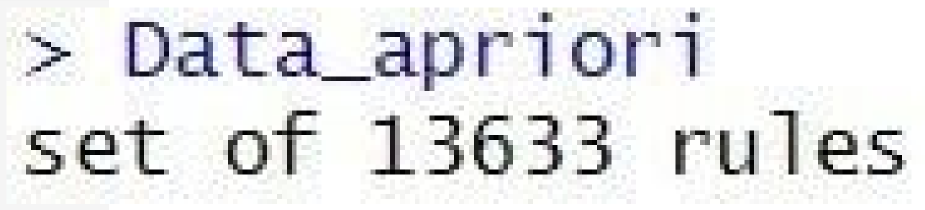


Data science project - Team 79

Enter the support value (between 0.001 and 1): 0.001

Enter the confidence value (between 0.001 and 1): 0.002

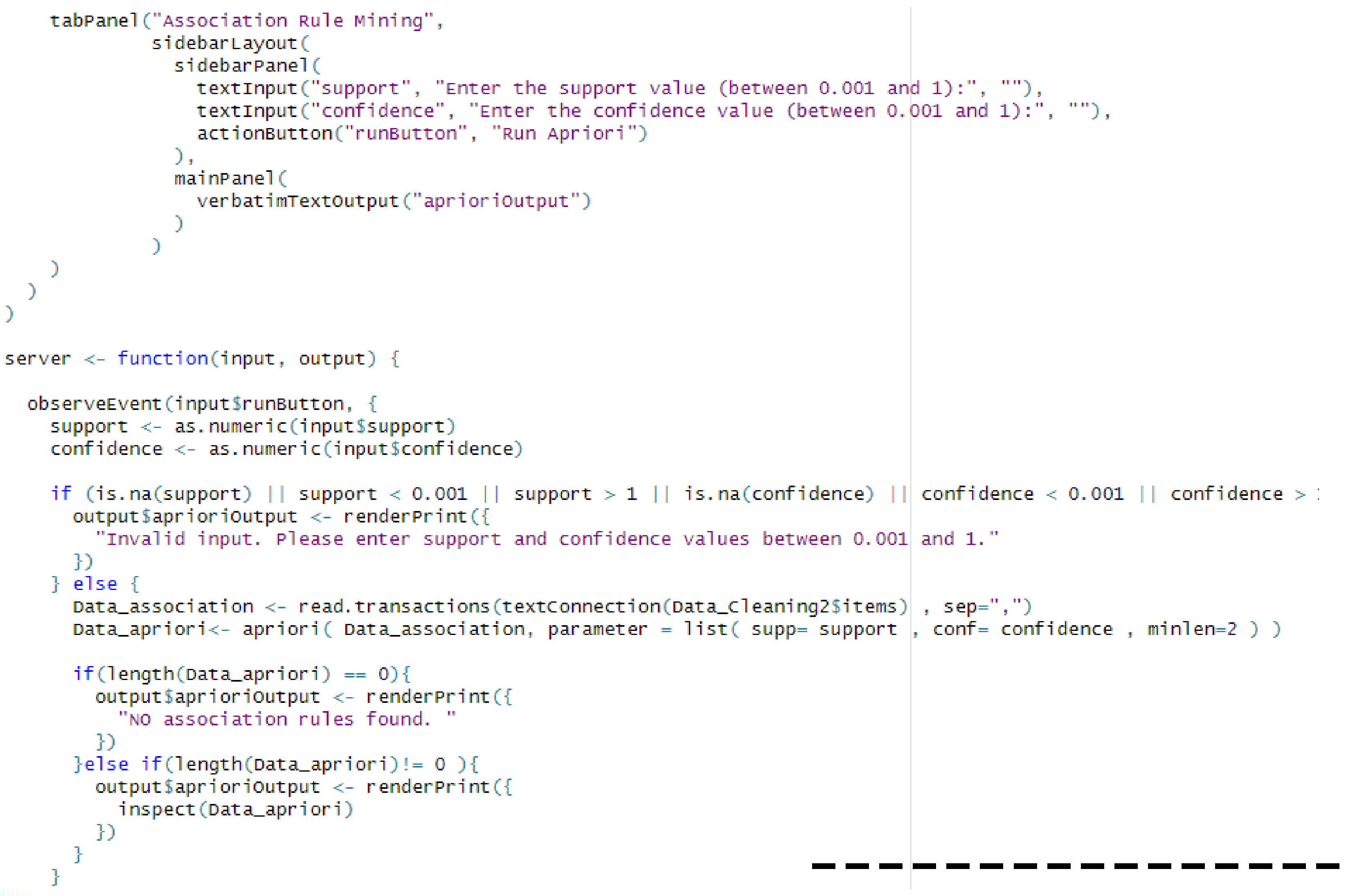
```
        lhs                      rhs                    support confidence    coverage      lift count
[1]     {honey}               => {whole milk}       0.001054630 0.769230769 0.001371019 3.1962516    10
[2]     {whole milk}          => {honey}            0.001054630 0.004382121 0.240666526 3.1962516    10
[3]     {cocoa drinks}        => {whole milk}       0.001054630 0.588235294 0.001792871 2.4441924    10
[4]     {whole milk}          => {cocoa drinks}     0.001054630 0.004382121 0.240666526 2.4441924    10
[5]     {pudding powder}      => {whole milk}       0.001265556 0.631578947 0.002003797 2.6242908    12
[6]     {whole milk}          => {pudding powder}   0.001265556 0.005258545 0.240666526 2.6242908    12
[7]     {abrasive cleaner}    => {whole milk}       0.001054630 0.416666667 0.002531112 1.7313030    10
[8]     {whole milk}          => {abrasive cleaner} 0.001054630 0.004382121 0.240666526 1.7313030    10
[9]     {artif. sweetener}    => {yogurt}           0.001054630 0.357142857 0.002952964 2.8243775    10
[10]    {yogurt}              => {artif. sweetener} 0.001054630 0.008340284 0.126450116 2.8243775    10
[11]    {specialty fat}       => {margarine}        0.001160093 0.366666667 0.003163889 6.9119947    11
[12]    {margarine}           => {specialty fat}    0.001160093 0.021868787 0.053047880 6.9119947    11
[13]    {male cosmetics}      => {bottled water}    0.001265556 0.285714286 0.004429445 2.7172947    12
[14]    {bottled water}       => {male cosmetics}   0.001265556 0.012036108 0.105146594 2.7172947    12
[15]    {rum}                 => {other vegetables} 0.001265556 0.307692308 0.004113056 1.7366300    12
[16]    {other vegetables}    => {rum}              0.001265556 0.007142857 0.177177811 1.7366300    12
[17]    {rum}                 => {whole milk}       0.001476482 0.358974359 0.004113056 1.4915841    14
[18]    {whole milk}          => {rum}              0.001476482 0.006134969 0.240666526 1.4915841    14
[19]    {brandy}              => {shopping bags}    0.001371019 0.317073171 0.004323982 3.3592043    13
[20]    {shopping bags}       => {brandy}           0.001371019 0.014525140 0.094389369 3.3592043    13
[21]    {brandy}              => {rolls/buns}       0.001160093 0.268292683 0.004323982 1.5026292    11
[22]    {rolls/buns}          => {brandy}           0.001160093 0.006497342 0.178548829 1.5026292    11
[23]    {meat spreads}        => {yogurt}           0.001581945 0.405405405 0.003902130 3.2060501    15
[24]    {yogurt}              => {meat spreads}     0.001581945 0.012510425 0.126450116 3.2060501    15
[25]    {meat spreads}        => {soda}             0.001371019 0.351351351 0.003902130 2.1045569    13
[26]    {soda}                => {meat spreads}     0.001371019 0.008212255 0.166947901 2.1045569    13
[27]    {meat spreads}        => {rolls/buns}       0.001265556 0.324324324 0.003902130 1.8164461    12
[28]    {rolls/buns}          => {meat spreads}     0.001265556 0.007088009 0.178548829 1.8164461    12
[29]    {meat spreads}        => {whole milk}       0.001054630 0.270270270 0.003902130 1.1230073    10
[30]    {whole milk}          => {meat spreads}     0.001054630 0.004382121 0.240666526 1.1230073    10
[31]    {liver loaf}          => {rolls/buns}       0.001160093 0.261904762 0.004429445 1.4668523    11
[32]    {rolls/buns}          => {liver loaf}       0.001160093 0.006497342 0.178548829 1.4668523    11
[33]    {liver loaf}          => {other vegetables} 0.001265556 0.285714286 0.004429445 1.6125850    12
[34]    {other vegetables}    => {liver loaf}       0.001265556 0.007142857 0.177177811 1.6125850    12
```



```
> Data_apriori
set of 13633 rules
```

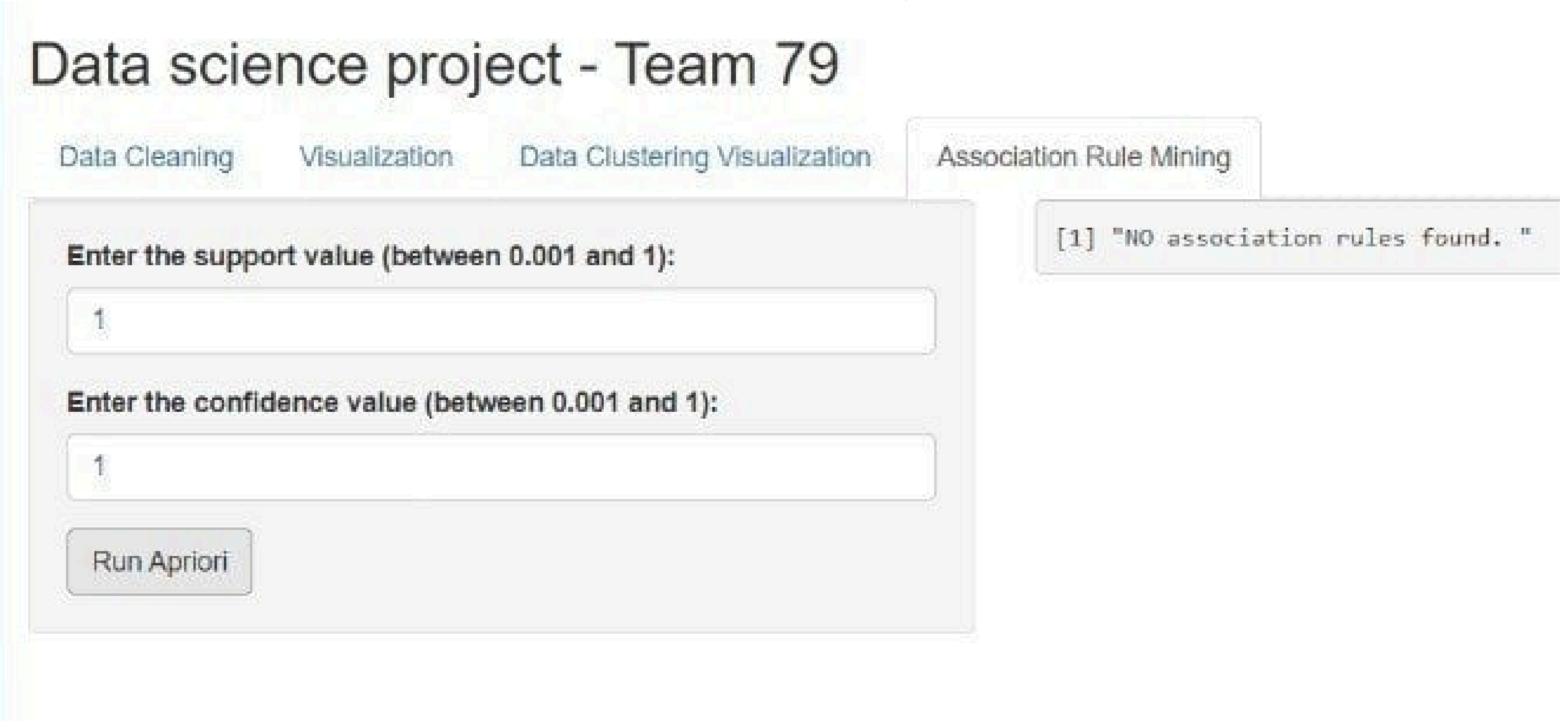
The Input is going to be the support and confidence and the Output will be the discovered associations rules between items

Association UI code



```
    tabPanel("Association Rule Mining",
             sidebarLayout(
               sidebarPanel(
                 textInput("support", "Enter the support value (between 0.001 and 1):", ""),
                 textInput("confidence", "Enter the confidence value (between 0.001 and 1):", ""),
                 actionButton("runButton", "Run Apriori")
               ),
               mainPanel(
                 verbatimTextOutput("aprioriOutput")
               )
             )
    )
  )
)

server <- function(input, output) {

  observeEvent(input$runButton, {
    support <- as.numeric(input$support)
    confidence <- as.numeric(input$confidence)

    if (is.na(support) || support < 0.001 || support > 1 || is.na(confidence) || confidence < 0.001 || confidence > :
      output$aprioriOutput <- renderPrint({
        "Invalid input. Please enter support and confidence values between 0.001 and 1."
      })
    } else {
      Data_association <- read.transactions(textConnection(Data_Cleaning2$items) , sep=",")
      Data_apriori<- apriori( Data_association, parameter = list( supp= support , conf= confidence , minlen=2 ) )

      if(length(Data_apriori) == 0){
        output$aprioriOutput <- renderPrint({
          "NO association rules found. "
        })
      }else if(length(Data_apriori)!= 0 ){
        output$aprioriOutput <- renderPrint({
          inspect(Data_apriori)
        })
      }
    }
```

creating a tab panel for conducting association rule mining. It includes sidebar inputs for users to define support and confidence values, along with a button to run the Apriori algorithm. The main panel displays the discovered association rules or provides feedback if the input values are invalid or if no association rules are found. The server function handles the logic for processing user inputs and displaying results accordingly.



Data science project - Team 79

Data Cleaning | Visualization | Data Clustering Visualization | Association Rule Mining

Enter the support value (between 0.001 and 1):

1

Enter the confidence value (between 0.001 and 1):

1

Run Apriori

```
[1] "NO association rules found. "
```

when support is equal to 1 and confidence is equal to 1 there is no Apriori rules found.

UI functions explanation:

1. fluidPage(): This function creates a fluid layout for the Shiny app, which adjusts its size based on the user's device or browser window.

2. titlePanel(): Creates a title panel at the top of the page with the specified title.

3. tabsetPanel(): Creates a tab set with multiple tabs, where each tab contains different content.

4. tabPanel(): Defines a tab within the tab set. Each tab has a title and content.

5. sidebarLayout(): Divides the layout into a sidebar panel and a main panel. Sidebar panels typically contain input controls, while main panels display outputs or visualizations.

6. sidebarPanel(): Creates a sidebar panel where users can input data or choose options.

7. fileInput(): Creates a file input control for users to upload CSV files.

8. selectInput(): Generates a dropdown menu for users to select different options.

9. sliderInput(): Creates a slider control for users to select a numeric value within a specified range.

10. actionButton(): Creates a button that triggers an action when clicked.

11. textInput(): Generates a text input box for users to enter text or numeric values.

12. mainPanel(): Defines the main content panel where outputs, plots, or visualizations are displayed.

13. plotOutput(): Specifies an area where plots generated by the server are displayed.

14. verbatimTextOutput(): Defines an area where text output generated by the server is displayed as-is.

15. dataTableOutput(): Specifies an area where data tables generated by the server are displayed.

Thank you!