Project Report

Contents

Team members

Name	ID	Department
Ahmed Ashraf Mohamed	2022446758	Business
Abdelrhman Mohamed Abdelhady	2022513643	Intelligent Systems
Antonuose Gerges Nageh	20221903971	Intelligent Systems

Roles

- Ahmed Ashraf Mohamed
    - ☒ Data Processing and cleaning
        - ☒ preparing the data for k-means and Apriori algorithms
    - ☒ Data visualization
        - ☒ Part 1
        - ☒ Part 2
    - ☒ Implementation of Apriori's algorithm
- Abelrhman Mohamed Adelhady

    - ☒ Data visualization
        - ☒ Cleaning the data in order to be used in data visualization part 3 & 4
        - ☒ Part 3
        - ☒ Part 4
- Antonuose Gerges Nageh
    - ☐ Project description
        - ☒ Describe Role of members
        - ☒ Full Description of data set
        - ☐ What will the program do?
        - ☐ What the output from the program will be ?

Project Description

Description of the data

When we skim through our data, we find that it's a grocery store's data. The data contains

- items sold
- count (Number of items in a single entry)
- Total price of the item entry
- Rnd (Customer number)
- Customer's name
- Customer's age
- City
- Type of Payment (Cash or Credit)

Items are types of food, vegetables, fruits, . . . , etc. Customers are from different cities and they have different ages. There are two ways to pay cash or credit. The number of transactions in this data is 9863.

Getting and preparing the data set

The first thing we are going to do is importing the libraries we are going to use, and then import the data into a data frame. The second thing is that we will prepare data for k-means and apriori algorithms by

- Creating a data frame containing the customers, their ages, and total spending, and then grouping it by the customers
- We make the data frame suitable for k-means.
    - By sub-setting the data frame created using only the age and total columns
- We make the data suitable for apriori's algorithm by
    - Splitting the transactions'e entries into separate items and transforming it into a transactions object

importing the library we are going to use

```
library(dplyr)
library(ggplot2)
library(forcats)
library(arules)
library(hrbrthemes)
```

Reading the data and checking the frist 10 rows of it

```
dataPath <- readline("Enter the path to the data set : ")
grc <- as_tibble(read.csv(dataPath,stringsAsFactors = FALSE))
```

```r
# displaying first 10 rows of our data
print(grc,n = 10, width = 80)
```

```
## # A tibble: 9,835 x 8
##    items                     count total   rnd customer   age city   paymentType
##    <chr>                     <int> <int> <int> <chr>    <int> <chr>  <chr>
##  1 citrus fruit,semi-finish~     4  1612     9 Maged       60 Hurgh~ Cash
##  2 tropical fruit,yogurt,co~     3   509    12 Eman        23 Aswan  Cash
##  3 whole milk                    1  2084     8 Rania       37 Dakah~ Cash
##  4 pip fruit,yogurt,cream c~     4   788     8 Rania       37 Dakah~ Cash
##  5 other vegetables,whole m~     4  1182    14 Magdy       36 Sohag  Cash
##  6 whole milk,butter,yogurt~     5  1771     3 Ahmed       30 Giza   Credit
##  7 rolls/buns                    1  2196     7 Huda        39 Gharb~ Cash
##  8 other vegetables,UHT-mil~     5  1657     6 Walaa       29 Cairo  Cash
##  9 pot plants                    1  2373     2 Mohamed     25 Alexa~ Credit
## 10 whole milk,cereals            2   343     5 Shimaa      55 Port ~ Cash
## # ... with 9,825 more rows
```

Preparing the data for k-means and apriori algorithms

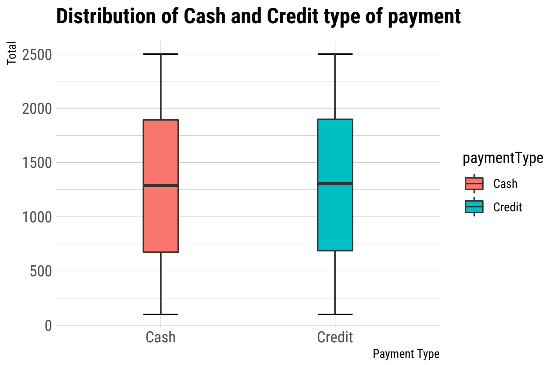
```r
# creating a data framing containing customers, their ages and total spending
grc_customers <- grc %>%
  select(customer,age,total) %>%
  group_by(customer)%>%
  mutate(total = sum(total))%>%
  unique()
# making the data frame suitable for k-means
grc_k <- data.frame(grc_customers[,2:3],row.names = grc_customers$customer)
# splitting the items to be suitable for apriori algorithms
tdata <- strsplit(as.vector(grc$items), ',')
tdata <- transactions(tdata)
```

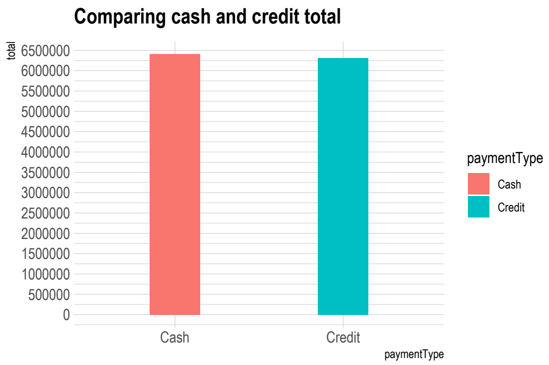
Visualizing our Data

Comparison between cash and credit total spending

```r
boxplot_cashCredit <- ggplot(
  grc,
  aes(x = paymentType, y = total, fill = paymentType)) +
  stat_boxplot(geom = "errorbar", width = .2) +
  geom_boxplot(width = .2,
               outlier.color = "orange",
               outlier.size = 2)+
  theme_ipsum_rc() +
  xlab("Payment Type") +
  ylab("Total") +
  theme(
    plot.title = element_text(size=16)) +
  ggtitle("Distribution of Cash and Credit type of payment")
barplot_cashCredit <- ggplot(grc,
                             aes(x = paymentType,
                                 y = total,
                                 fill = paymentType)) +
  geom_col(width = .3) +
  scale_y_continuous(n.breaks = 12) +
  theme_ipsum() +
  theme(
    plot.title = element_text(size = 16)) +
  ggtitle("Comparing cash and credit total")

print(boxplot_cashCredit)
print(barplot_cashCredit)
```





Observations

After brief moments of seeing the figures, we find that the distribution of cash and credit types of payment is nearly identical, but the amount payed in cash is greater than the amount payed in credit.

Compare each age and sum of total spending.

Before visualizing

let's create a contingency table using the data prepared and look at number of individuals in each age group.

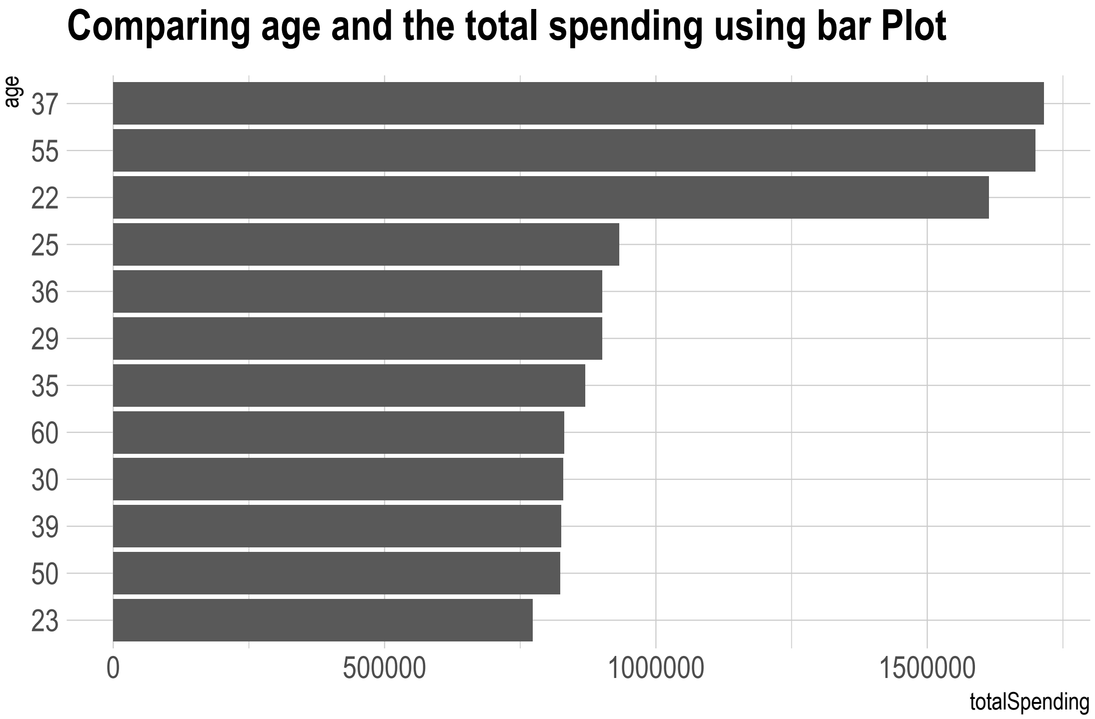
```
table(grc_customers$age)
```

```
##
## 22 23 25 29 30 35 36 37 39 50 55 60
##  2  1  1  1  1  1  1  2  1  1  2  1
```

It becomes clear that there are more people aged (22,37 and 55) than the other age groups

```
grc_age <- select(grc,age,total)
grc_age <- grc_age %>%
  group_by(age) %>%
  summarise(totalSpending = sum(total))
grc_age <- mutate(grc_age,age = fct_reorder(as.factor(age),totalSpending))
barPlotAgeSum<-ggplot(
  grc_age,
  aes(x = age, y = totalSpending)) +
  geom_col() +
  coord_flip() +
  theme_ipsum() +
  theme(
    plot.title = element_text(size=16),
    legend.position = "none")+
  ggtitle("Comparing age and the total spending using bar Plot")

print(barPlotAgeSum)
```



Observations

Since there are more customers aged (22, 37 and 55), it makes sense that the total spending of these age groups is higher than the rest.

Comparing the cities' total spending

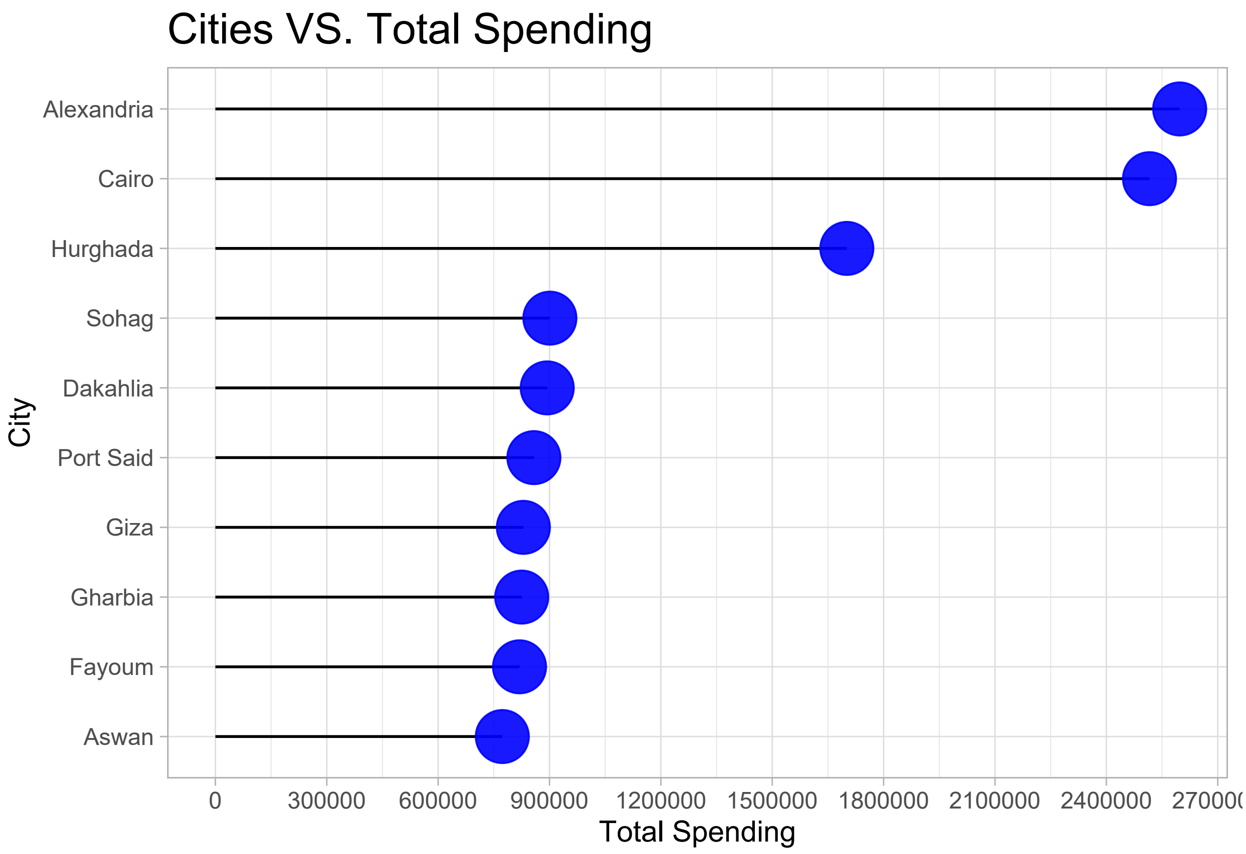
```
###Cleaning the data in order to be prepared for data visualization
C_Vs_To <-grc %>%
  select(city,total) %>%
  group_by(city) %>%
  summarise(totalspending = sum(total))
C_Vs_To <- mutate(C_Vs_To, city = fct_reorder(as.factor(city),totalspending))
print(C_Vs_To)
```

```
## # A tibble: 10 x 2
##    city       totalspending
##    <fct>              <int>
##  1 Alexandria       2597481
##  2 Aswan             772871
##  3 Cairo             2516267
##  4 Dakahlia           893789
##  5 Fayoum             819231
##  6 Gharbia            825147
##  7 Giza               829587
##  8 Hurghada          1700940
##  9 Port Said          857901
## 10 Sohag              901010
```

```
###Visualizing

CityandTotalspending<- ggplot(C_Vs_To,aes(city,totalspending)) +
  geom_segment( aes(xend=city,y = 0,yend=totalspending)) +
  scale_y_continuous(n.breaks = 10) +
  geom_point( color="blue", size=9, alpha=.9) +
  theme_light() +
  xlab("City") +
  ylab("Total Spending") +
  coord_flip() +
  theme(
    plot.title = element_text(size=16))+
  ggtitle("Cities VS. Total Spending")

print(CityandTotalspending)
```



Observations

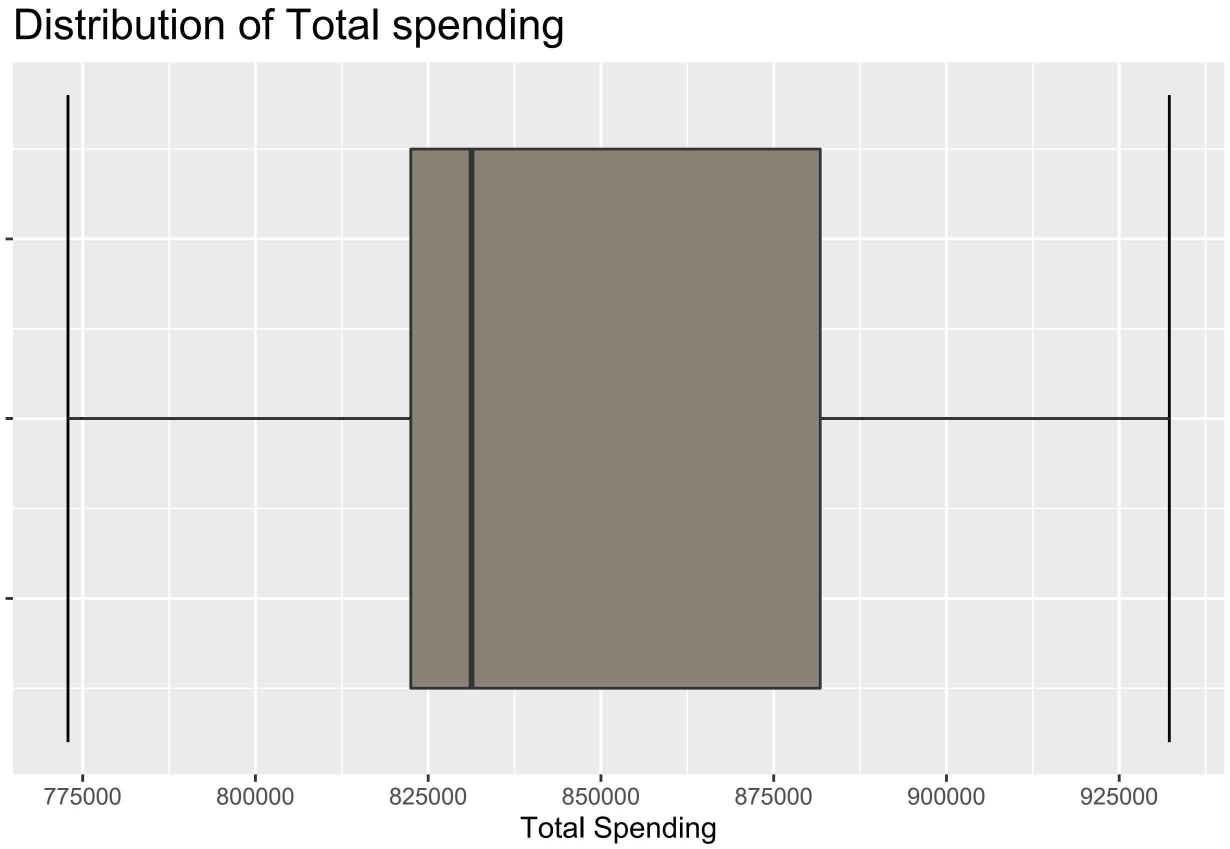
After we doing the data visualization on cities and their total spending, We will find that Alexandria and Cairo are the highest cities in spending, Aswan and Fayoum are the smallest in spending.

Distribution of Total spending

```
## Display the distribution of total spending.
Distribution_of_total_spending<-ggplot(grc_customers,aes(total)) +
  stat_boxplot(geom = "errorbar",width = .9) +
  geom_boxplot(fill = "antiquewhite4",) +
  scale_x_continuous(n.breaks = 8) +
  theme_grey() +
  theme(
    plot.title = element_text(size=16),
    axis.text.y = element_blank()) +
  xlab("Total Spending")  +
  ggtitle("Distribution of Total spending")
summary(grc_customers)
```

```
##    customer              age           total
##  Length:15          Min.   :22.0   Min.   :772871
##  Class :character   1st Qu.:27.0   1st Qu.:822482
##  Mode  :character   Median :36.0   Median :831272
##                     Mean   :37.0   Mean   :847615
##                     3rd Qu.:44.5   3rd Qu.:881729
##                     Max.   :60.0   Max.   :932250
```

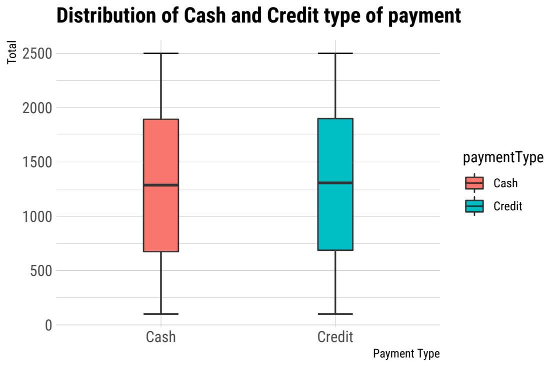
```
print(Distribution_of_total_spending)
```

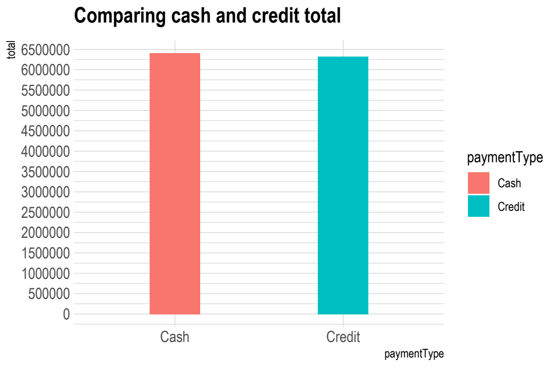


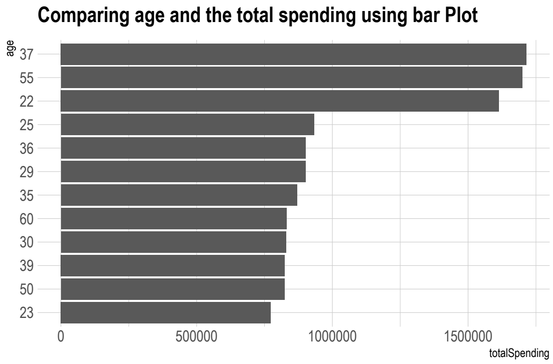
Observations

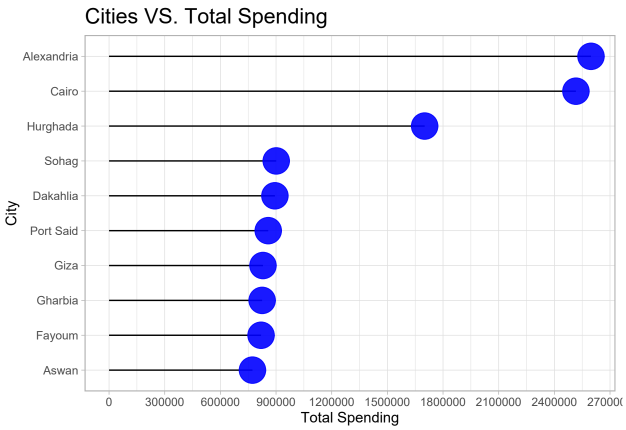
We will find that the total spending are between 772871 and 932250 and most of customers spend between 822482 and 881729 and their mean is 847615.

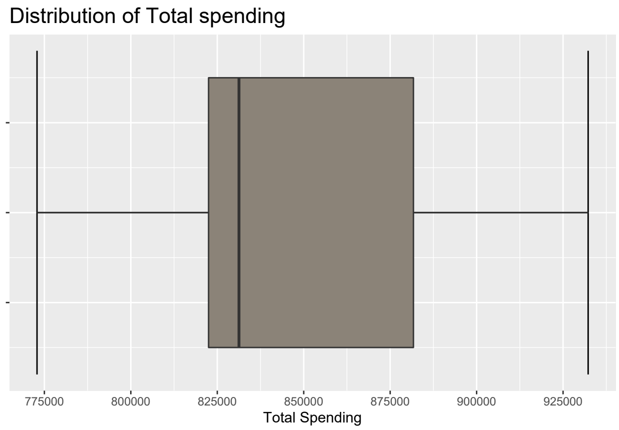
Dashboard











K-means

In k-means we will group the customers into (n) of clusters according to their age and their total spending and then we will put it in a table, The number of clusters will be specified by the user.

Getting the number of clusters from the user

```
No_of_clusters<-as.numeric(readline("Enter the number of clusters: "))
```

Implementation of the algorithm using the bult-in function

```
Kmeans_Algorithm<-kmeans(grc_kmeans,centers = No_of_clusters)
grc_kmeans<-mutate(grc_kmeans,cluster=Kmeans_Algorithm$cluster)
print(grc_kmeans)
```

```
##         age  total cluster
## Maged    60 831272       2
## Eman     23 772871       2
## Rania    37 893789       1
## Magdy    36 901010       1
## Ahmed    30 829587       2
## Huda     39 825147       2
## Walaa    29 900797       1
## Mohamed  25 932250       1
## Shimaa   55 857901       1
## Farida   22 794570       2
## Hanan    22 819231       2
## Sayed    37 820900       2
## Adel     50 824064       2
## Sameh    35 869668       1
## Samy     55 841167       2
```

Generating the association rules

Brief explanition of Apriori algorithm for generating the rules

Apriori algorithm is an iterative approach for discovering the most frequent item sets. The frequent item sets generated by the algorithm can be used to determine association rules that highlight general trends in the data-set, it is especially useful in the analysis of super-market items in our data set

Implementing the alogrithm

Reading both minimum support and minimum confidance from the user

```
min_support <- as.numeric(readline("Enter the minimum Support : "))
min_conf <- as.numeric(readline("Enter the minimum Confidance : "))
```

implementing the algorithm using the built-in function

```
apriori_rules <- apriori(
  tdata,
  parameter = list(supp = min_support, conf = min_conf, minlen = 2))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
##         0.5    0.1    1 none FALSE            TRUE       5    0.01      2     10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
```

```
## Absolute minimum support count: 98
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [88 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [15 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

Printing the result Taking into consideration that there might be no rules generated.

```
# Displaying at most 100 rows of the rules
if(length(size(apriori_rules)) == 0){
  print("No rules were generated")
}else{
  as_tibble(DATAFRAME(apriori_rules,separate = TRUE, setStart = "", setEnd = "")) %>%
    print(n = 100, width = 90)
}
```

```
## # A tibble: 15 x 7
##    LHS                                 RHS         support confidence coverage  lift count
##    <fct>                               <fct>         <dbl>      <dbl>    <dbl> <dbl> <int>
##  1 curd,yogurt                         whole milk   0.0101      0.582   0.0173  2.28    99
##  2 butter,other vegetables             whole milk   0.0115      0.574   0.0200  2.24   113
##  3 domestic eggs,other vegetables      whole milk   0.0123      0.553   0.0223  2.16   121
##  4 whipped/sour cream,yogurt           whole milk   0.0109      0.525   0.0207  2.05   107
##  5 other vegetables,whipped/sour cream whole milk   0.0146      0.507   0.0289  1.98   144
##  6 other vegetables,pip fruit          whole milk   0.0135      0.518   0.0261  2.03   133
##  7 citrus fruit,root vegetables        other vege~  0.0104      0.586   0.0177  3.03   102
##  8 root vegetables,tropical fruit      other vege~  0.0123      0.585   0.0210  3.02   121
##  9 root vegetables,tropical fruit      whole milk   0.0120      0.570   0.0210  2.23   118
## 10 tropical fruit,yogurt               whole milk   0.0151      0.517   0.0293  2.02   149
## 11 root vegetables,yogurt              other vege~  0.0129      0.5     0.0258  2.58   127
## 12 root vegetables,yogurt              whole milk   0.0145      0.563   0.0258  2.20   143
## 13 rolls/buns,root vegetables          other vege~  0.0122      0.502   0.0243  2.59   120
## 14 rolls/buns,root vegetables          whole milk   0.0127      0.523   0.0243  2.05   125
## 15 other vegetables,yogurt             whole milk   0.0223      0.513   0.0434  2.01   219
```