Market Basket Analysis

R Markdown

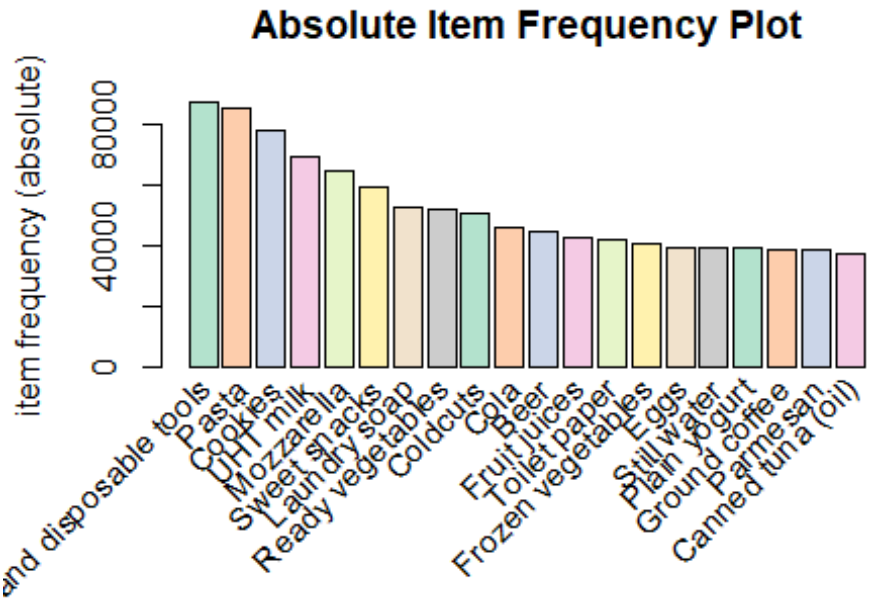
```
#install libraries
library(data.table)
library(tidyverse)
library(arules)
library(arulesViz)
library(dplyr)
library(readxl)
library(RColorBrewer)
library(writexl)


#Import dataset
catalogue <- read_excel("C:/Users/aberg/OneDrive - Politecnico di Milano/Università/3.Magistrale/Analytics Lab/Project/MBA/Dataset/products_DB.xlsx")
transactions <- read_csv("C:/Users/aberg/OneDrive - Politecnico di Milano/Università/3.Magistrale/Analytics Lab/Project/MBA/Dataset/ticket_description.txt")

transactions$prod_id <- as.numeric(transactions$prod_id)

mydata <- dplyr::full_join(transactions, catalogue, by="prod_id")
```

#Data Cleaning: NAs management

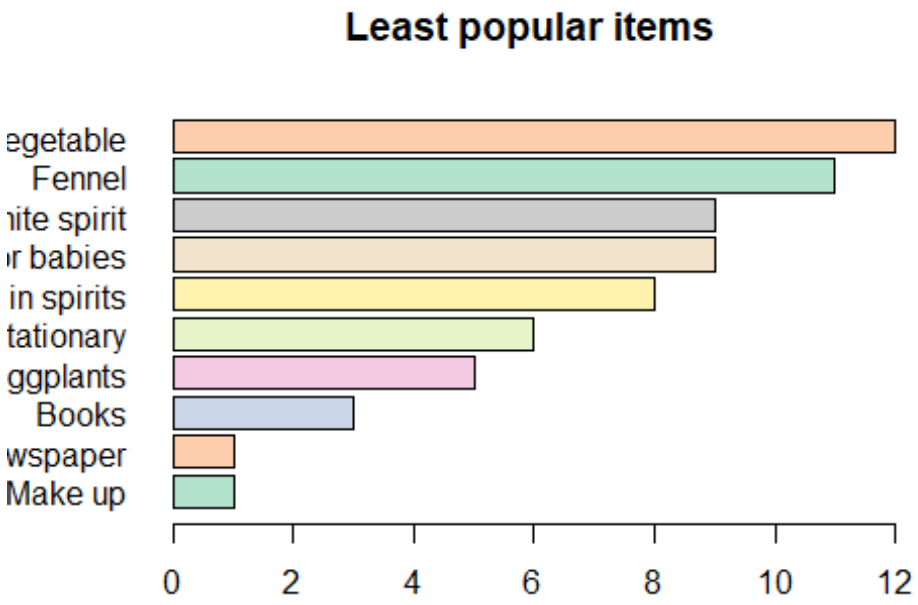
```
#Identify products sold but not present in the catalog
table(is.na(mydata$Description)==T)                    #16302 NAs lines (no description)

##
##   FALSE    TRUE
## 3791709   16302

NAs_index = which(is.na(mydata$Description)==T)
WO_Descr = unique(mydata[NAs_index, "prod_id"])         #17 prod_ID without description


#idenitfy products present in the catalog but not sold
table(is.na(mydata$ticket_id)==T)                      #424 NAs lines (no ticket_ID)

##
##   FALSE    TRUE
## 3807587     424

NAs_index2 = which(is.na(mydata$ticket_id)==T)
wo_ticket = unique(mydata[NAs_index2, "Description"])   #126 ticket with different description

#Keep the lines without NAs, that correspond to an inner join
mydata2 = na.omit(mydata)
mydata2
```

#Data Exploration

```
unique_descr <- unique(catalogue$Description)
#442 different categories in catalog
unique_prod <- unique(catalogue$prod_id)
#874 different products in catalog
unique_prod_sold <- unique(transactions$prod_id)
#467 different products sold

`%!in%` <- Negate(`%in%`)
unique_prod_unsold <- mydata[mydata$prod_id %!in% unique_prod_sold,3]
#424 product unsold
unique_prod_unsold <- unique(unique_prod_unsold)
#126 unique product unsold

categories_sold <- as.data.frame(unique(mydata2$Description))
#338 different categories sold
colnames(categories_sold) <- c('Description')
categories_unsold <- catalogue[catalogue$Description%!in%categories_sold$Description,2]
categories_unsold <- unique(categories_unsold)
#104 unique categories unsold

#Average Basket Size
mydata2$Description <- as.character(mydata2$Description)
MBA_list = split(mydata2$Description, mydata2$ticket_id)

# removing duplicated items in transactions
for (i in 1:length(MBA_list)) {
  MBA_list[[i]] = unique(MBA_list[[i]])
}

#most sold items
MBA_list = as(MBA_list, 'transactions')

best_seller <- data.frame(itemFrequency(MBA_list, type = "absolute"))
itemFrequencyPlot(MBA_list, type = "absolute", topN = 20, col=brewer.pal(8,'Pastel2'), ma
in="Absolute Item Frequency Plot")
```



```
#unpopular items
leastseller <- sort(table(unlist(LIST(MBA_list))))
barplot(sort(table(unlist(LIST(MBA_list))))[1:10],horiz = TRUE,las = 1, main = 'Least pop
ular items', col=brewer.pal(8,'Pastel2'))
```




```
#Extract the set of most frequent itemsets
itemsets = apriori(MBA_list, parameter = list(support = 0.2, target = 'frequent'))

inspect(sort(itemsets, by='support', decreasing = T)[1:5,])

##     items                          support   count
## [1] {Paper and disposable tools} 0.2981461 87135
## [2] {Pasta}                      0.2918708 85301
## [3] {Cookies}                    0.2676934 78235
## [4] {UHT milk}                   0.2359746 68965
## [5] {Mozzarella}                 0.2210288 64597

# Extract the set of most frequent itemsets
itemsets_minlen2 = apriori(MBA_list, parameter = list(support = 0.01, minlen = 2, target
= 'frequent'))

inspect(sort(itemsets_minlen2, by='support', decreasing = T)[1:5])

##     items                                    support    count
## [1] {Paper and disposable tools, Pasta}      0.10930828 31946
## [2] {Cookies, Pasta}                         0.10057963 29395
## [3] {Cookies, Paper and disposable tools}    0.09385949 27431
## [4] {Pasta, UHT milk}                        0.09082790 26545
## [5] {Paper and disposable tools, UHT milk} 0.08711541 25460
```

#Rules Definition

```
rules <- apriori(MBA_list, parameter = list(supp = 0.0001, conf = 0.0), target = 'rules',
minlen = 2, maxlen = 3)

rules <- sort(rules, by = 'lift', decreasing = T)
```

```
#delete duplicates in rules

gi <- generatingItemsets(rules)
d <- which(duplicated(gi))
rules <- rules[-d]

MBA_all <- DATAFRAME(rules)

mean(MBA_all$support)

## [1] 0.0005259385

mean(MBA_all$confidence)

## [1] 0.2187186

mean(MBA_all$lift)

## [1] 1.6104282

basket <- filter(MBA_all, MBA_all$lift > 1.6 & MBA_all$support > 0.0005 &MBA_all$confiden
ce < 0.2)
```

#Rules with one item in the itemset on the left

```
rules2 <- apriori(MBA_list, parameter = list(supp=0.0001, conf=0), target = 'rules', minl
en = 2, maxlen = 3)

gi <- generatingItemsets(rules2)
d <- which(duplicated(gi))
rules2 <- rules2[-d]

rules2 <- DATAFRAME(rules2)
MBA_only2 <- filter(rules2, rules2$lift > 1.6 & rules2$support > 0.0005)
```

#Rules with two item in the itemset on the left

```
rules3 <- apriori(MBA_list, parameter = list(supp=0.0001, conf=0), target = 'rules', minl
en = 3, maxlen = 3)

gi <- generatingItemsets(rules3)
d <- which(duplicated(gi))
rules3 <- rules3[-d]

rules3 <- DATAFRAME(rules3)
MBA_only3 <- filter(rules3, rules3$lift > 1.6 & rules3$support > 0.0005)
```

#Rules with three item in the itemset on the left

```
rules4 <- apriori(MBA_list, parameter = list(supp=0.0001, conf=0), target = 'rules', minl
en = 4, maxlen = 4)

gi <- generatingItemsets(rules4)
d <- which(duplicated(gi))
rules4 <- rules4[-d]

rules4 <- DATAFRAME(rules4)
MBA_only4 <- filter(rules4, rules4$lift > 1.6 & rules4$support > 0.0005)
```

#Panko Rules

```
rules_panko <- apriori(MBA_list, parameter = list(supp = 0.0001, conf = 0.0), target = 'r
ules', minlen = 2, maxlen = 3, appearance = list(default = 'rhs', lhs = c('Panko'))

rules_panko <- DATAFRAME(rules_panko)
rules_panko <- filter(rules_panko, rules_panko$lift > 1.2 & rules_panko$support > 0.0005)
```

#Gluten free Products

```
#import dataset
glutenfree <- read_excel("C:/Users/aberg/OneDrive - Politecnico di Milano/Università/3.Ma
gistrale/Analytics Lab/Project/MBA/Dataset/GlutenFree_DB.xlsx")

glutendata <- dplyr::inner_join(transactions, glutenfree, by="prod_id")

glutendata$Description <- as.character(glutendata$Description)
MBA_gluten = split(glutendata$Description, glutendata$ticket_id)

for (i in 1:length(MBA_gluten)) {
  MBA_gluten[[i]] = unique(MBA_gluten[[i]])
}

MBA_gluten = as(MBA_gluten, 'transactions')

#generate rules
rules_gluten <- apriori(MBA_gluten, parameter = list(supp=0.0001, conf=0), target = 'rule
s', minlen = 2, maxlen = 3, appearance = list(default = 'rhs', lhs = 'Gluten free'))

rules_gluten <- DATAFRAME(rules_gluten)
rules_gluten <- filter(rules_gluten, rules_gluten$lift > 1.4 & rules_gluten$support > 0.0
005)
```

#Vegan Products

```
#import dataset
vegan <- read_excel("C:/Users/aberg/OneDrive - Politecnico di Milano/Università/3.Magistr
ale/Analytics Lab/Project/MBA/Dataset/Vegan_DB.xlsx")

vegandata <- dplyr::inner_join(transactions, vegan, by="prod_id")

vegandata$Description2 <- as.character(vegandata$Description2)
MBA_vegan = split(vegandata$Description2, vegandata$ticket_id)

for (i in 1:length(MBA_vegan)) {
  MBA_vegan[[i]] = unique(MBA_vegan[[i]])
}

MBA_vegan = as(MBA_vegan, 'transactions')

#generate rules
rules_vegan <- apriori(MBA_vegan, parameter = list(supp=0.0001, conf=0), target = 'rules'
, minlen = 2, maxlen = 3, appearance = list(default = 'rhs', lhs = 'Vegan'))
```

```
rules_vegan <- DATAFRAME(rules_vegan)
rules_vegan <- filter(rules_vegan, rules_vegan$lift > 1.2 & rules_vegan$support > 0.0005)
```

#Vegetarian Products

```
#import dataset
vegetarian <- read_xlsx("C:/Users/aberg/OneDrive - Politecnico di Milano/Università/3.Magistrale/Analytics Lab/Project/MBA/Dataset/Vegetarian_DB.xlsx")

vegetariandata <- dplyr::inner_join(transactions, vegetarian, by="prod_id")
vegetariandata$Description <- as.character(vegetariandata$Description)

MBA_vegetarian = split(vegetariandata$Description, vegetariandata$ticket_id)

for (i in 1:length(MBA_vegetarian)) {
MBA_vegetarian[[i]] = unique(MBA_vegetarian[[i]])
}

MBA_vegetarian = as(MBA_vegetarian, 'transactions')

#generate rules
rules_vegetarian <- apriori(MBA_vegetarian, parameter = list(supp=0.0001, conf=0), target = 'rules', minlen = 2, maxlen = 3, appearance = list(default = 'rhs', lhs = 'Vegetarian'))

rules_vegetarian <- DATAFRAME(rules_vegetarian)
rules_vegetarian <- filter(rules_vegetarian, rules_vegetarian$lift > 1.6 & rules_vegetarian$support > 0.0005)

rules_vegetarian2 <- apriori(MBA_vegetarian, parameter = list(supp=0.0001, conf=0), target = 'rules', minlen = 2, maxlen = 3, appearance = list(rhs = 'Vegetarian', default = 'lhs'))

rules_vegetarian2 <- DATAFRAME(rules_vegetarian2)
rules_vegetarian2 <- filter(rules_vegetarian2, rules_vegetarian2$lift > 1.6 & rules_vegetarian2$support > 0.0005)
```

#Baby products

```
#import dataset
baby <- read_excel("C:/Users/aberg/OneDrive - Politecnico di Milano/Università/3.Magistrale/Analytics Lab/Project/MBA/Dataset/Baby_DB.xlsx")

babydata <- dplyr::inner_join(transactions, baby, by="prod_id")
babydata$Description <- as.character(babydata$Description)

MBA_baby = split(babydata$Description, babydata$ticket_id)

for (i in 1:length(MBA_baby)) {
MBA_baby[[i]] = unique(MBA_baby[[i]])
}

MBA_baby = as(MBA_baby, 'transactions')

#generate rules
rules_baby <- apriori(MBA_baby, parameter = list(supp=0.0001, conf=0), target = 'rules',
```

```
minlen = 2, maxlen = 4, appearance = list(default = 'rhs', lhs =c('Baby food', 'Baby prod
ucts')))

rules_baby <- DATAFRAME(rules_baby)
rules_baby <- filter(rules_baby, rules_baby$lift > 1.6 & rules_baby$support > 0.0005)
```

#Animals products

```
#import dataset
animals <- read_excel("C:/Users/aberg/OneDrive - Politecnico di Milano/Università/3.Magis
trale/Analytics Lab/Project/MBA/Dataset/Animals_DB.xlsx")

animalsdata <- dplyr::inner_join(transactions, animals, by="prod_id")
animalsdata$Description <- as.character(animalsdata$Description)

MBA_animals = split(animalsdata$Description, animalsdata$ticket_id)

for (i in 1:length(MBA_animals)) {
MBA_animals[[i]] = unique(MBA_animals[[i]])
}

MBA_animals = as(MBA_animals, 'transactions')

#generate rules
rules_animals <- apriori(MBA_animals, parameter = list(supp=0.0001, conf=0), target = 'ru
les', minlen = 2, maxlen = 3, appearance = list(default = 'lhs', rhs =c('Cat', 'Dog')))

rules_animals <- DATAFRAME(rules_animals)
rules_animals <- filter(rules_animals, rules_animals$lift > 1.6 & rules_animals$support >
0.0005)

#check of the rules on the original database
rules <- apriori(MBA_list, parameter = list(supp = 0.0001, conf = 0.0), target = 'rules',
minlen = 2, maxlen = 3, appearance = list(default = 'rhs', lhs =c('Cat food', 'Dog food')
))

rules <- sort(rules, by = 'lift', decreasing = T)
rules <- DATAFRAME(rules)
```

#Rule with aggregate cathegories

```
#import dataset
aggregate <- read_excel("C:/Users/aberg/OneDrive - Politecnico di Milano/Università/3.Mag
istrale/Analytics Lab/Project/MBA/Dataset/Aggregate_DB.xlsx")

aggregatedata <- dplyr::inner_join(transactions, aggregate, by="prod_id")
aggregatedata$Cathegories <- as.character(aggregatedata$Cathegories)

MBA_aggregate = split(aggregatedata$Cathegories, aggregatedata$ticket_id)

for (i in 1:length(MBA_aggregate)) {
MBA_aggregate[[i]] = unique(MBA_aggregate[[i]])
}

MBA_aggregate = as(MBA_aggregate, 'transactions')
```

```r
#generate rules
rules_aggregate <- apriori(MBA_aggregate, parameter = list(supp=0.0001, conf=0), target =
'rules', minlen = 2, maxlen = 2)

gi <- generatingItemsets(rules_aggregate)
d <- which(duplicated(gi))
rules_aggregate <- rules_aggregate[-d]

rules_aggregate <- DATAFRAME(rules_aggregate)
rules_aggregate <- filter(rules_aggregate, rules_aggregate$lift > 1.6 & rules_aggregate$s
upport > 0.0005  & rules_aggregate$confidence > 0.2)
```