MBA

Suprasanna Pradhan

12 October 2019

```r
#Load the libraries
#library(readr)
library(arules)
```

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

```
## 
## Attaching package: 'arules'
```

```
## The following objects are masked from 'package:base':
## 
##     abbreviate, write
```

```r
library(arulesViz)
```

```
## Loading required package: grid
```

```
## Registered S3 method overwritten by 'seriation':
##   method         from 
##   reorder.hclust gclus
```

```r
library(magrittr)
library(dplyr)
```

```
## 
## Attaching package: 'dplyr'
```

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

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

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```r
library(forcats)
library(ggplot2)
library(arulesViz)
library(datasets)
library(RColorBrewer)
```

```r
library(readxl)
library(tibble)
#Importing Data set
setwd("C:/Users/SuprasannaPradhan/Documents/My Files/Great Lakes Projects/Project-8 M
A")
cafe=read_excel("Cafe Great Transaction Data set.xlsx")
cafe_df = as.data.frame(cafe)
str(cafe_df)
```

```
## 'data.frame':    145830 obs. of  10 variables:
##  $ Date       : POSIXct, format: "2011-01-25" "2011-02-20" ...
##  $ Bill Number: chr  "G0522713" "N0033392" "N0032132" "N0030048" ...
##  $ Item Desc  : chr  "PARTY CHARGES @ 500/-" "KF DRAUGHT PITCHER (2LTR)" "SULA BRUT
(BTL)" "RED SANGRIA (CARAFE) Ã¡Ã¡Ã¡Ã¡Ã¡Ã¡Ã¡Ã¡" ...
##  $ Time       : POSIXct, format: "1899-12-31 21:21:02" "1899-12-31 21:24:40" ...
##  $ Quantity   : num  23 10 3 4 9 13 2 11 8 6 ...
##  $ Rate       : num  500 400 1200 850 350 225 1300 250 330 400 ...
##  $ Tax        : num  2731 1250 936 884 984 ...
##  $ Discount   : num  0 0 0 0 0 0 0 825 0 0 ...
##  $ Total      : num  14231 5250 4536 4284 4134 ...
##  $ Category   : chr  "MISC" "LIQUOR" "WINES" "WINES" ...
```

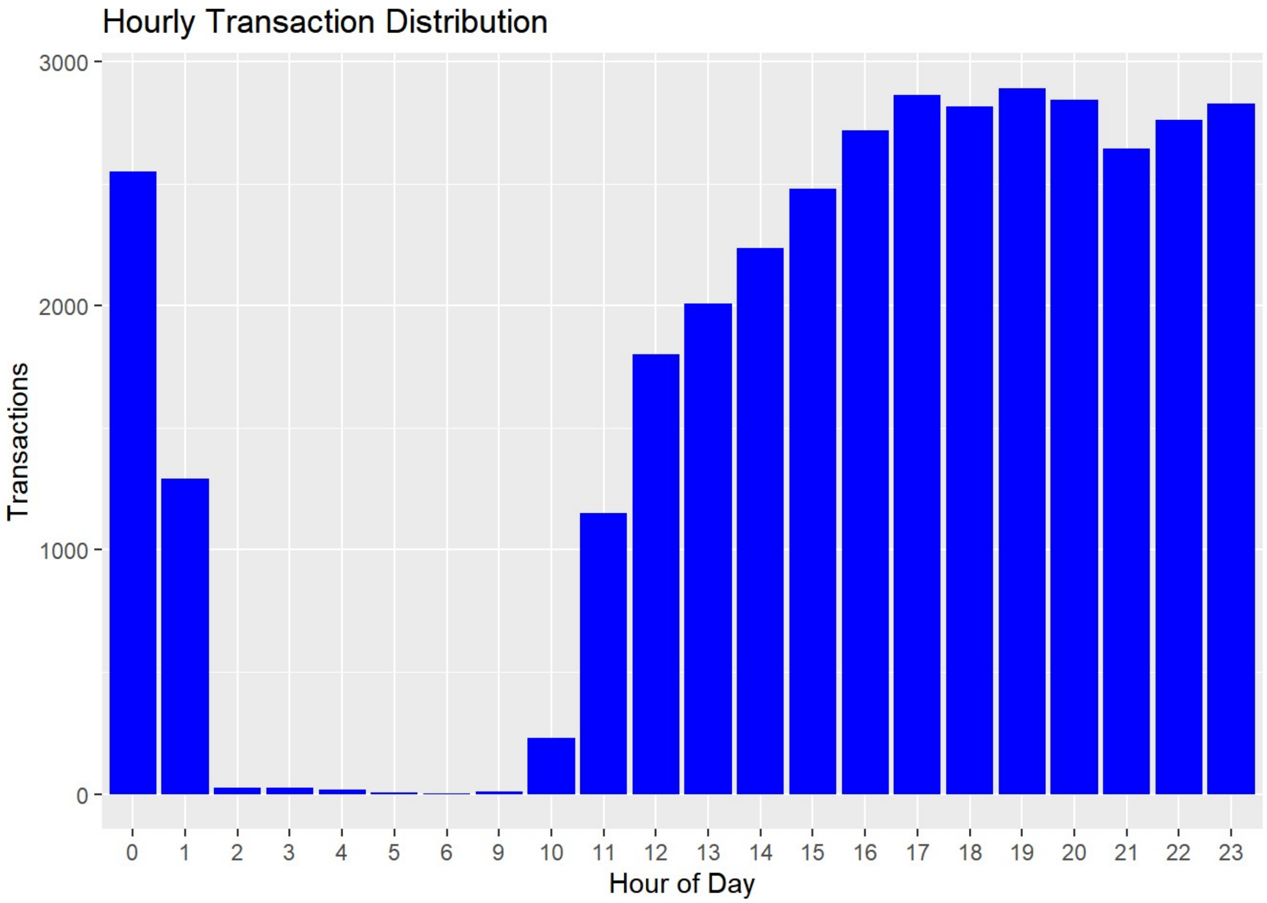
```
colnames(cafe_df)[colnames(cafe_df)=="Item Desc"] <- "Item_Desc"
colnames(cafe_df)[colnames(cafe_df)=="Bill Number"] <-"Bill_Number"
sum(is.na(cafe_df))
```

```
## [1] 0
```

```
str(cafe_df)
```

```
## 'data.frame':    145830 obs. of  10 variables:
##  $ Date       : POSIXct, format: "2011-01-25" "2011-02-20" ...
##  $ Bill_Number: chr  "G0522713" "N0033392" "N0032132" "N0030048" ...
##  $ Item_Desc  : chr  "PARTY CHARGES @ 500/-" "KF DRAUGHT PITCHER (2LTR)" "SULA BRUT
(BTL)" "RED SANGRIA (CARAFE) Ã¡Ã¡Ã¡Ã¡Ã¡Ã¡Ã¡Ã¡" ...
##  $ Time       : POSIXct, format: "1899-12-31 21:21:02" "1899-12-31 21:24:40" ...
##  $ Quantity   : num  23 10 3 4 9 13 2 11 8 6 ...
##  $ Rate       : num  500 400 1200 850 350 225 1300 250 330 400 ...
##  $ Tax        : num  2731 1250 936 884 984 ...
##  $ Discount   : num  0 0 0 0 0 0 0 825 0 0 ...
##  $ Total      : num  14231 5250 4536 4284 4134 ...
##  $ Category   : chr  "MISC" "LIQUOR" "WINES" "WINES" ...
```

```
library(dplyr)
purchase_time <-
  cafe_df %>%
  group_by(Time) %>%
  slice(1) %>%
  mutate(time_of_day = hour(Time)) %>%
  pull(time_of_day) %>%
  as.factor() %>%
  fct_count()
purchase_time %>%
  ggplot() +
  geom_col(aes(x = f, y = n), fill = "blue") +
  xlab("Hour of Day") + ylab("Transactions") +
  ggtitle("Hourly Transaction Distribution")
```



```
library(dplyr)
#How many items are purchased on an average?
items <-
  cafe_df %>%
  group_by(Item_Desc) %>%
  summarize(count = n()) %>%
  pull(count)
mean(items)
```

```
## [1] 251.431
```

```
median(items)
```

```
## [1] 37
```

```
#Most Purchased Items
cafe_df %>%
  group_by(Item_Desc)%>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

```
## # A tibble: 580 x 2
##    Item_Desc                  count
##    <chr>                      <int>
##  1 NIRVANA HOOKAH SINGLE       8553
##  2 MINT FLAVOUR SINGLE         5817
##  3 CAPPUCCINO                  5495
##  4 GREAT LAKES SHAKE           4895
##  5 SAMBUCA                     4425
##  6 POUTINE WITH FRIES          3464
##  7 QUA  MINERAL WATER(1000ML)  3331
##  8 CALCUTTA MINT               3318
##  9 JR.CHL AVALANCHE            3314
## 10 B.M.T. PANINI               2724
## # ... with 570 more rows
```

```
#Average Order Value

total_revenue <-
  cafe_df %>%
  group_by(Bill_Number) %>%
 summarize(order_sum = sum(Total)) %>%
  pull(order_sum) %>%
  sum()

total_transactions <-
  cafe_df%>%
  group_by(Bill_Number) %>%
  summarize(n()) %>%
  nrow()

total_revenue / total_transactions
```

```
## [1] 468.7762
```

```
summary(cafe_df)
```

```
##       Date                     Bill_Number         Item_Desc
##  Min.   :2010-04-01 00:00:00   Length:145830      Length:145830
##  1st Qu.:2010-07-10 00:00:00   Class :character   Class :character
##  Median :2010-10-07 00:00:00   Mode  :character   Mode  :character
##  Mean   :2010-10-04 15:52:45
##  3rd Qu.:2011-01-01 00:00:00
##  Max.   :2011-03-31 00:00:00
##       Time                        Quantity           Rate
##  Min.   :1899-12-31 00:00:01   Min.   : 1.000   Min.   :   0.01
##  1st Qu.:1899-12-31 14:48:53   1st Qu.: 1.000   1st Qu.:  95.00
##  Median :1899-12-31 18:17:24   Median : 1.000   Median : 125.00
##  Mean   :1899-12-31 16:39:16   Mean   : 1.121   Mean   : 161.78
##  3rd Qu.:1899-12-31 21:02:20   3rd Qu.: 1.000   3rd Qu.: 225.00
##  Max.   :1899-12-31 23:59:57   Max.   :30.000   Max.   :2100.00
##       Tax             Discount           Total
##  Min.   :   0.00   Min.   :  0.0000   Min.   :    0.01
##  1st Qu.:  22.56   1st Qu.:  0.0000   1st Qu.:  117.56
##  Median :  32.06   Median :  0.0000   Median :  167.06
##  Mean   :  48.93   Mean   :  0.0951   Mean   :  224.96
##  3rd Qu.:  72.00   3rd Qu.:  0.0000   3rd Qu.:  315.00
##  Max.   :2731.25   Max.   :825.0000   Max.   :14231.25
##    Category
##  Length:145830
##  Class :character
##  Mode  :character
##
##
##
```

```
#cafe_df$ID = as.factor(cafe_df$ID)
#cafe_df$Date = as.factor(cafe_df$Date)
#cafe_df$Bill_Number = as.factor(cafe_df$Bill_Number)
cafe_df$Item_Desc = as.factor(cafe_df$Item_Desc)
#cafe_df$Time = as.factor(cafe_df$Time)
#cafe_df$Quantity = as.factor(cafe_df$Quantity)
#cafe_df$Rate = as.factor(cafe_df$Rate)
#cafe_df$Tax = as.factor(cafe_df$Tax)
#cafe_df$Discount = as.factor(cafe_df$Discount)
#cafe_df$Total = as.factor(cafe_df$Total)
cafe_df$Category = as.factor(cafe_df$Category)
str(cafe_df)
```

```
## 'data.frame':    145830 obs. of  10 variables:
##  $ Date       : POSIXct, format: "2011-01-25" "2011-02-20" ...
##  $ Bill_Number: chr  "G0522713" "N0033392" "N0032132" "N0030048" ...
##  $ Item_Desc  : Factor w/ 580 levels "1 AXE TWIST",..: 411 322 504 448 494 321 490
492 69 322 ...
##  $ Time       : POSIXct, format: "1899-12-31 21:21:02" "1899-12-31 21:24:40" ...
##  $ Quantity   : num  23 10 3 4 9 13 2 11 8 6 ...
##  $ Rate       : num  500 400 1200 850 350 225 1300 250 330 400 ...
##  $ Tax        : num  2731 1250 936 884 984 ...
##  $ Discount   : num  0 0 0 0 0 0 0 825 0 0 ...
##  $ Total      : num  14231 5250 4536 4284 4134 ...
##  $ Category   : Factor w/ 8 levels "BEVERAGE","FOOD",..: 6 3 8 8 3 3 3 3 2 3 ...
```

convert to transaction

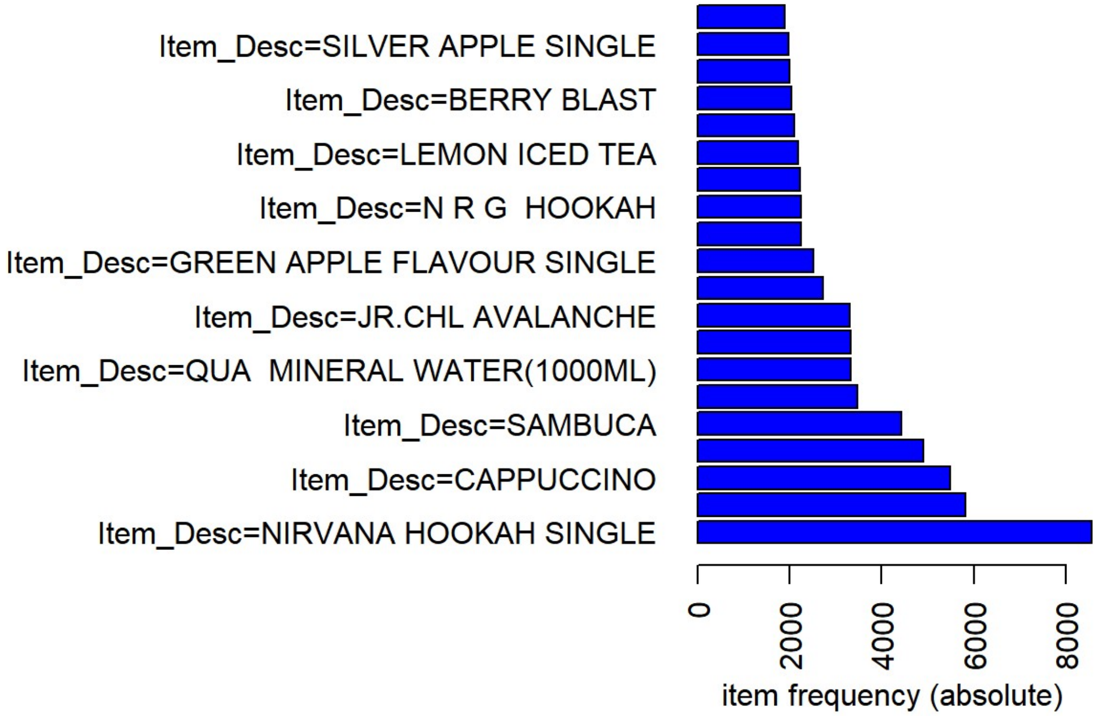
```
cafe_trasn<- subset(cafe_df ,select= c(3))
trans1 <- as(cafe_trasn, "transactions")
summary(trans1)
```

```
## transactions as itemMatrix in sparse format with
##  145830 rows (elements/itemsets/transactions) and
##  580 columns (items) and a density of 0.001724138
##
## most frequent items:
## Item_Desc=NIRVANA HOOKAH SINGLE   Item_Desc=MINT FLAVOUR SINGLE
##                            8553                            5817
##            Item_Desc=CAPPUCCINO     Item_Desc=GREAT LAKES SHAKE
##                            5495                            4895
##               Item_Desc=SAMBUCA                         (Other)
##                            4425                          116645
##
## element (itemset/transaction) length distribution:
## sizes
##      1
## 145830
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1       1       1       1       1       1
##
## includes extended item information - examples:
##                            labels variables                levels
## 1           Item_Desc=1 AXE TWIST Item_Desc           1 AXE TWIST
## 2 Item_Desc=1+1 BTL4 SEASON WHITE Item_Desc 1+1 BTL4 SEASON WHITE
## 3   Item_Desc=1+1 GLS 4SEASON RED Item_Desc   1+1 GLS 4SEASON RED
##
## includes extended transaction information - examples:
##   transactionID
## 1             1
## 2             2
## 3             3
```

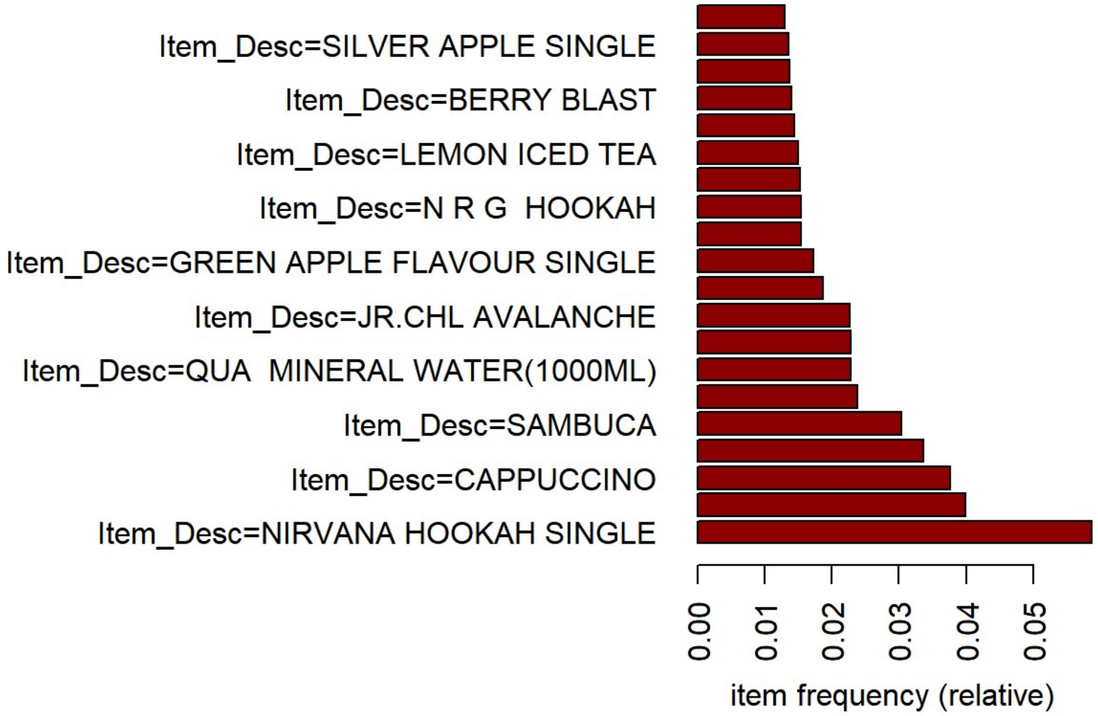
```
inspect(trans1[20])
```

```
##     items                                transactionID
## [1] {Item_Desc=HOEGAARDEN MUG (1 LITRE)} 20
```

```
itemFrequencyPlot(trans1, topN = 20,horiz = TRUE, type = 'absolute', col = "blue")
```



```
itemFrequencyPlot(trans1, topN = 20, type = 'relative',horiz = TRUE, col = "dark red")
```



```
cafe_df$bill_number <- gsub("(\\D+)", "",cafe_df$Bill_Number)
cafe_df$bill_number = as.factor(cafe_df$bill_number)
cafe_sorted <- cafe_df[order(cafe_df$bill_number),]
cafe_mba <- subset(cafe_sorted,select= c(3,11))
str(cafe_mba)
```

```
## 'data.frame':    145830 obs. of  2 variables:
##  $ Item_Desc  : Factor w/ 580 levels "1 AXE TWIST",..: 132 132 539 539 3 132 539 53
9 132 132 ...
##  $ bill_number: Factor w/ 69982 levels "0027835","0027836",..: 1 2 3 4 5 6 7 8 9 10
...
```

```
cafe_mba$bill_number <- as.numeric(cafe_mba$bill_number)
head(cafe_mba)
```

```
##                  Item_Desc bill_number
## 80310           CARLSBERG           1
## 80311           CARLSBERG           2
## 48201              TUBORG           3
## 99550              TUBORG           4
## 53538 1+1 GLS 4SEASON RED           5
## 80312           CARLSBERG           6
```

```
cafe_mba$bill_number <- as.numeric(cafe_mba$bill_number)
tail(cafe_mba)
```

```
##                     Item_Desc bill_number
## 21159  NIRVANA HOOKAH SINGLE       69982
## 66534  RED BULL ENERGY DRINK       69982
## 78061          KIT KAT SHAKE       69982
## 78062      GREAT LAKES SHAKE       69982
## 78063         SUNSHINE SHAKE       69982
## 139211     MOROCCAN MINT TEA       69982
```

```
library(plyr)
```

```
## -------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## -------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following object is masked from 'package:lubridate':
##
##     here
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
df_itemList <- ddply(cafe_mba,'bill_number', function(df1)paste(df1$Item_Desc,collapse
= ","))
head(df_itemList)
```

```
##   bill_number                  V1
## 1           1           CARLSBERG
## 2           2           CARLSBERG
## 3           3              TUBORG
## 4           4              TUBORG
## 5           5 1+1 GLS 4SEASON RED
## 6           6           CARLSBERG
```

Now, we remove the column ‘bill- number’.

```
df_itemList$bill_number <- NULL
head(df_itemList)
```

```
##                   V1
## 1           CARLSBERG
## 2           CARLSBERG
## 3              TUBORG
## 4              TUBORG
## 5 1+1 GLS 4SEASON RED
## 6           CARLSBERG
```

```
colnames(df_itemList) <- c("Item List")
head(df_itemList)
```

```
##             Item List
## 1           CARLSBERG
## 2           CARLSBERG
## 3              TUBORG
## 4              TUBORG
## 5 1+1 GLS 4SEASON RED
## 6           CARLSBERG
```

```
write.csv(df_itemList,"ItemList.csv", quote = FALSE, row.names = TRUE)
```

```
library(arules)
trans2 = read.transactions(file="ItemList.csv", rm.duplicates= FALSE, format="basket",
sep=",",cols= 1);
```

```
## Warning in asMethod(object): removing duplicated items in transactions
```

```
inspect(head(trans2))
```

```
##     items                 transactionID
## [1] {Item List}                        
## [2] {CARLSBERG}           1            
## [3] {CARLSBERG}           2            
## [4] {TUBORG}              3            
## [5] {TUBORG}              4            
## [6] {1+1 GLS 4SEASON RED} 5
```

```
#trans$items <- gsub("\\d+", "", df),trans$items)
#trans@itemInfo$labels <- gsub("","",trans@itemInfo$labels)
```

```
print(head(itemInfo(trans2)))
```

```
##                    labels
## 1            1 AXE TWIST
## 2  1+1 BTL4 SEASON WHITE
## 3    1+1 GLS 4SEASON RED
## 4 1+1 GLS 4SEASONS WHITE
## 5         1+1 KF 1 LITER
## 6       1+1 KF 1/2 LITER
```

```
print(levels(itemInfo(trans2)[["level1"]]))
```

```
## NULL
```

```
print(levels(itemInfo(trans2)[["level6"]]))
```

```
## NULL
```

```
print(dim(trans2)[1])
```

```
## [1] 69983
```

```
print(dim(trans2)[2])
```

```
## [1] 581
```

Generate Rules

```
first.rules <- apriori(trans2,
                       parameter = list(support = 0.001, confidence = 0.01))
```

```
## Apriori
## 
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.01    0.1    1 none FALSE            TRUE       5   0.001      1
##  maxlen target   ext
##      10  rules FALSE
## 
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
## 
## Absolute minimum support count: 69 
## 
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[581 item(s), 69983 transaction(s)] done [0.04s].
## sorting and recoding items ... [210 item(s)] done [0.00s].
## creating transaction tree ... done [0.05s].
## checking subsets of size 1 2 3 done [0.01s].
## writing ... [677 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

Rules Summary

```
summary(first.rules)
```

```
## set of 677 rules
##
## rule length distribution (lhs + rhs):sizes
##   1   2
##  50 627
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   2.000   1.926   2.000   2.000
##
## summary of quality measures:
##     support           confidence           lift              count
##  Min.   :0.001000   Min.   :0.01007   Min.   : 0.1464   Min.   :  70.0
##  1st Qu.:0.001200   1st Qu.:0.02939   1st Qu.: 0.9843   1st Qu.:  84.0
##  Median :0.001529   Median :0.04755   Median : 1.2857   Median : 107.0
##  Mean   :0.003817   Mean   :0.05607   Mean   : 1.8382   Mean   : 267.1
##  3rd Qu.:0.002343   3rd Qu.:0.07025   3rd Qu.: 1.7865   3rd Qu.: 164.0
##  Max.   :0.119872   Max.   :0.54589   Max.   :79.3655   Max.   :8389.0
##
## mining info:
##    data ntransactions support confidence
##  trans2         69983   0.001       0.01
```

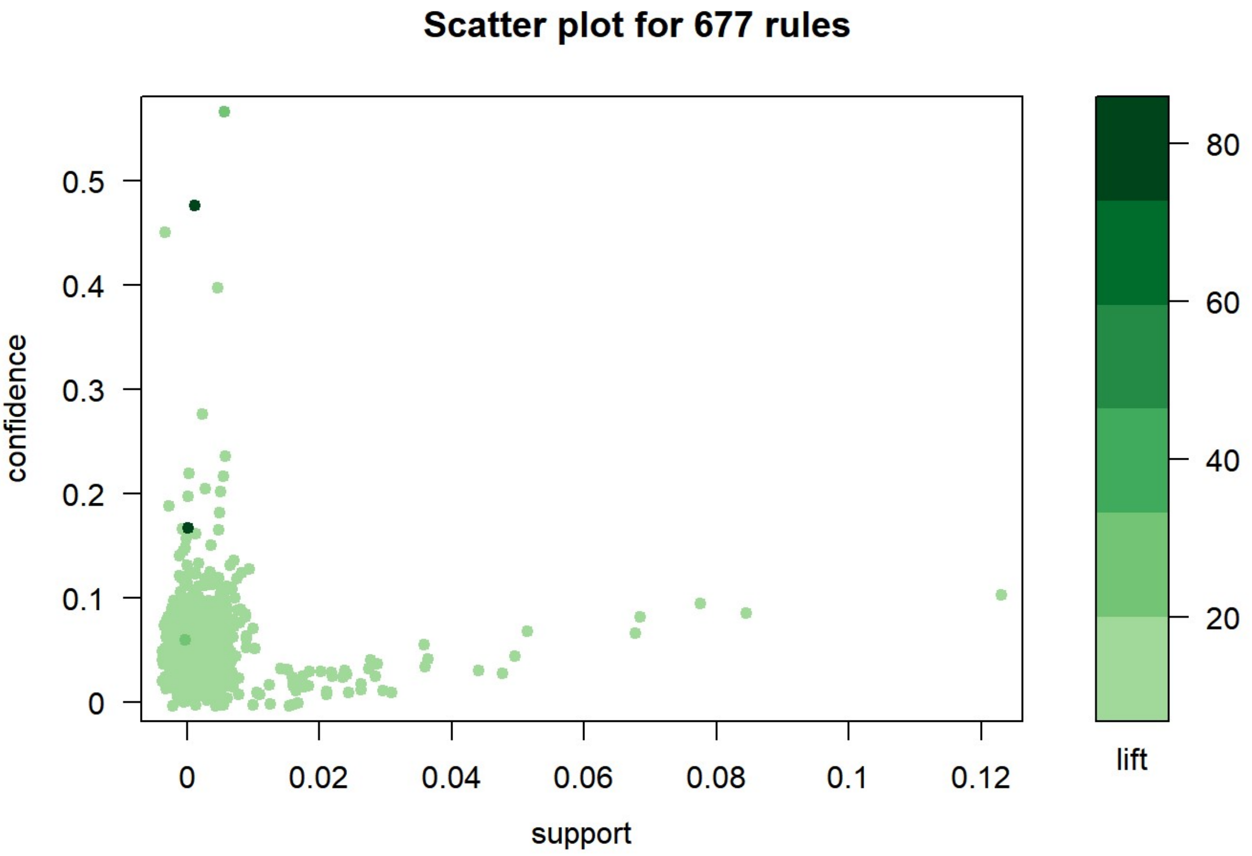
```
second.rules <- apriori(trans2,
                        parameter = list(support = 0.025, confidence = 0.04))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.04    0.1    1 none FALSE            TRUE       5   0.025      1
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 1749
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[581 item(s), 69983 transaction(s)] done [0.07s].
## sorting and recoding items ... [22 item(s)] done [0.00s].
## creating transaction tree ... done [0.04s].
## checking subsets of size 1 2 done [0.00s].
## writing ... [9 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

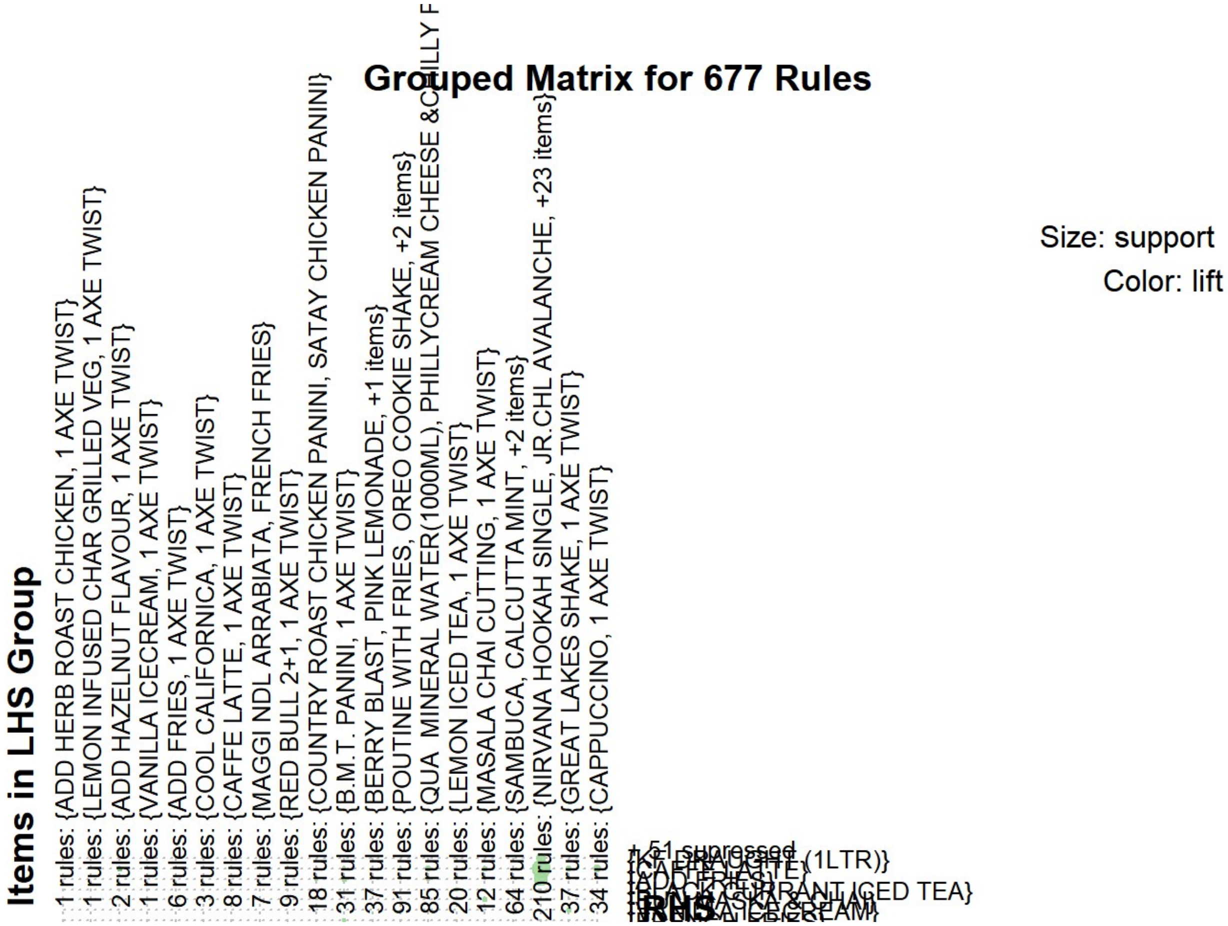
```
print(summary(second.rules))
```

```
## set of 9 rules
##
## rule length distribution (lhs + rhs):sizes
## 1
## 9
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1       1       1       1       1       1
##
## summary of quality measures:
##     support          confidence           lift       count
##  Min.   :0.04725   Min.   :0.04725   Min.   :1   Min.   :3307
##  1st Qu.:0.04755   1st Qu.:0.04755   1st Qu.:1   1st Qu.:3328
##  Median :0.06320   Median :0.06320   Median :1   Median :4423
##  Mean   :0.06719   Mean   :0.06719   Mean   :1   Mean   :4702
##  3rd Qu.:0.07813   3rd Qu.:0.07813   3rd Qu.:1   3rd Qu.:5468
##  Max.   :0.11987   Max.   :0.11987   Max.   :1   Max.   :8389
##
## mining info:
##    data ntransactions support confidence
##  trans2         69983   0.025       0.04
```

```
plot(first.rules,
     control=list(jitter=2, col = rev(brewer.pal(9, "Greens")[4:9])),
     shading = "lift")
```



```
plot(first.rules, method="grouped",
     control=list(col = rev(brewer.pal(9, "Greens")[4:9])))
```



```
bev.rules <- subset(first.rules, subset = rhs %pin% "CAFFE LATTE")
inspect(bev.rules)
```

```
##     lhs                              rhs           support     confidence
## [1] {}                            => {CAFFE LATTE} 0.029935842 0.02993584
## [2] {ADD HAZELNUT FLAVOUR}        => {CAFFE LATTE} 0.001614678 0.54589372
## [3] {ULTIMATE HOT CHOCOLATE}      => {CAFFE LATTE} 0.001014532 0.06641721
## [4] {MASALA CHAI CUTTING}         => {CAFFE LATTE} 0.001400340 0.04386750
## [5] {B.M.T. PANINI}               => {CAFFE LATTE} 0.001057400 0.02840691
## [6] {QUA  MINERAL WATER(1000ML)} => {CAFFE LATTE} 0.002286270 0.04807692
## [7] {CAPPUCCINO}                  => {CAFFE LATTE} 0.004243888 0.05431602
## [8] {GREAT LAKES SHAKE}           => {CAFFE LATTE} 0.002529186 0.03633751
##     lift       count
## [1]  1.0000000 2095
## [2] 18.2354559  113
## [3]  2.2186519   71
## [4]  1.4653840   98
## [5]  0.9489264   74
## [6]  1.6059987  160
## [7]  1.8144143  297
## [8]  1.2138462  177
```

```
fv.rules <- subset(first.rules, subset = rhs %pin% "ADD FRIES")
inspect(fv.rules)
```

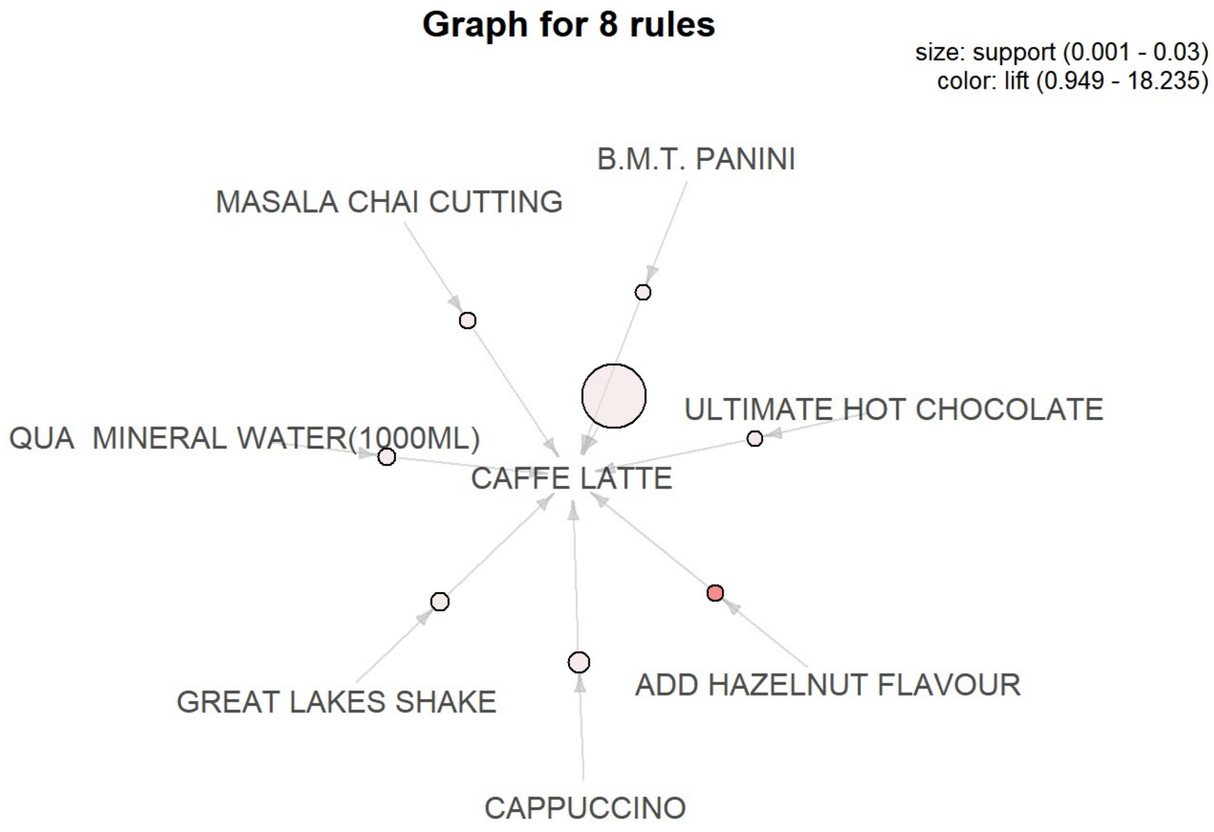
```
##     lhs                                 rhs         support     confidence
## [1] {}                               => {ADD FRIES} 0.010745467 0.01074547
## [2] {SATAY CHICKEN PANINI}           => {ADD FRIES} 0.001128846 0.08036623
## [3] {COUNTRY ROAST CHICKEN PANINI}   => {ADD FRIES} 0.001243159 0.06580938
## [4] {COTTAGE CHEESE PANINI}          => {ADD FRIES} 0.001043110 0.04528536
## [5] {PHILLYCREAM CHEESE &CHILLY PAN} => {ADD FRIES} 0.001171713 0.04408602
## [6] {B.M.T. PANINI}                  => {ADD FRIES} 0.002157667 0.05796545
## [7] {QUA  MINERAL WATER(1000ML)}     => {ADD FRIES} 0.001286027 0.02704327
##     lift     count
## [1] 1.000000 752
## [2] 7.479082  79
## [3] 6.124385  87
## [4] 4.214369  73
## [5] 4.102755  82
## [6] 5.394410 151
## [7] 2.516714  90
```

```
top.bev.rules <- head(sort(bev.rules, decreasing = TRUE, by = "lift"), 10)
inspect(top.bev.rules)
```

```
##     lhs                             rhs           support     confidence
## [1] {ADD HAZELNUT FLAVOUR}       => {CAFFE LATTE} 0.001614678 0.54589372
## [2] {ULTIMATE HOT CHOCOLATE}     => {CAFFE LATTE} 0.001014532 0.06641721
## [3] {CAPPUCCINO}                 => {CAFFE LATTE} 0.004243888 0.05431602
## [4] {QUA  MINERAL WATER(1000ML)} => {CAFFE LATTE} 0.002286270 0.04807692
## [5] {MASALA CHAI CUTTING}        => {CAFFE LATTE} 0.001400340 0.04386750
## [6] {GREAT LAKES SHAKE}          => {CAFFE LATTE} 0.002529186 0.03633751
## [7] {}                           => {CAFFE LATTE} 0.029935842 0.02993584
## [8] {B.M.T. PANINI}              => {CAFFE LATTE} 0.001057400 0.02840691
##     lift       count
## [1] 18.2354559  113
## [2]  2.2186519   71
## [3]  1.8144143  297
## [4]  1.6059987  160
## [5]  1.4653840   98
## [6]  1.2138462  177
## [7]  1.0000000 2095
## [8]  0.9489264   74
```

```
plot(top.bev.rules, method="graph",
     control=list(type="items"),
     shading = "lift")
```

```
## Warning: Unknown control parameters: type
## Available control parameters (with default values):
## main  =  Graph for 8 rules
## nodeColors    =  c("#66CC6680", "#9999CC80")
## nodeCol   =  c("#EE0000FF", "#EE0303FF", "#EE0606FF", "#EE0909FF", "#EE0C0CFF", "#E
E0F0FFF", "#EE1212FF", "#EE1515FF", "#EE1818FF", "#EE1B1BFF", "#EE1E1EFF", "#EE2222F
F", "#EE2525FF", "#EE2828FF", "#EE2B2BFF", "#EE2E2EFF", "#EE3131FF", "#EE3434FF", "#EE
3737FF", "#EE3A3AFF", "#EE3D3DFF", "#EE4040FF", "#EE4444FF", "#EE4747FF", "#EE4A4AFF",
"#EE4D4DFF", "#EE5050FF", "#EE5353FF", "#EE5656FF", "#EE5959FF", "#EE5C5CFF", "#EE5F5F
FF", "#EE6262FF", "#EE6666FF", "#EE6969FF", "#EE6C6CFF", "#EE6F6FFF", "#EE7272FF", "#E
E7575FF",  "#EE7878FF", "#EE7B7BFF", "#EE7E7EFF", "#EE8181FF", "#EE8484FF", "#EE8888F
F", "#EE8B8BFF", "#EE8E8EFF", "#EE9191FF", "#EE9494FF", "#EE9797FF", "#EE9999FF", "#EE
9B9BFF", "#EE9D9DFF", "#EE9F9FFF", "#EEA0A0FF", "#EEA2A2FF", "#EEA4A4FF", "#EEA5A5FF",
"#EEA7A7FF", "#EEA9A9FF", "#EEABABFF", "#EEACACFF", "#EEAEAEFF", "#EEB0B0FF", "#EEB1B1
FF", "#EEB3B3FF", "#EEB5B5FF", "#EEB7B7FF", "#EEB8B8FF", "#EEBABAFF", "#EEBCBCFF", "#E
EBDBDFF", "#EEBFBFFF", "#EEC1C1FF", "#EEC3C3FF", "#EEC4C4FF", "#EEC6C6FF", "#EEC8C8F
F",  "#EEC9C9FF", "#EECBCBFF", "#EECDCDFF", "#EECFCFFF", "#EED0D0FF", "#EED2D2FF", "#E
ED4D4FF", "#EED5D5FF", "#EED7D7FF", "#EED9D9FF", "#EEDBDBFF", "#EEDCDCFF", "#EEDEDEF
F", "#EEE0E0FF", "#EEE1E1FF", "#EEE3E3FF", "#EEE5E5FF", "#EEE7E7FF", "#EEE8E8FF", "#EE
EAEAFF", "#EEECECFF", "#EEEEEEFF")
## edgeCol   =  c("#474747FF", "#494949FF", "#4B4B4BFF", "#4D4D4DFF", "#4F4F4FFF", "#5
15151FF", "#535353FF", "#555555FF", "#575757FF", "#595959FF", "#5B5B5BFF", "#5E5E5EF
F", "#606060FF", "#626262FF", "#646464FF", "#666666FF", "#686868FF", "#6A6A6AFF", "#6C
6C6CFF", "#6E6E6EFF", "#707070FF", "#727272FF", "#747474FF", "#767676FF", "#787878FF",
"#7A7A7AFF", "#7C7C7CFF", "#7E7E7EFF", "#808080FF", "#828282FF", "#848484FF", "#868686
FF", "#888888FF", "#8A8A8AFF", "#8C8C8CFF", "#8D8D8DFF", "#8F8F8FFF", "#919191FF", "#9
39393FF",  "#959595FF", "#979797FF", "#999999FF", "#9A9A9AFF", "#9C9C9CFF", "#9E9E9EF
F", "#A0A0A0FF", "#A2A2A2FF", "#A3A3A3FF", "#A5A5A5FF", "#A7A7A7FF", "#A9A9A9FF", "#AA
AAAAFF", "#ACACACFF", "#AEAEAEFF", "#AFAFAFFF", "#B1B1B1FF", "#B3B3B3FF", "#B4B4B4FF",
"#B6B6B6FF", "#B7B7B7FF", "#B9B9B9FF", "#BBBBBBFF", "#BCBCBCFF", "#BEBEBEFF", "#BFBFBF
FF", "#C1C1C1FF", "#C2C2C2FF", "#C3C3C4FF", "#C5C5C5FF", "#C6C6C6FF", "#C8C8C8FF", "#C
9C9C9FF", "#CACACAFF", "#CCCCCCFF", "#CDCDCDFF", "#CECECEFF", "#CFCFCFFF", "#D1D1D1F
F",  "#D2D2D2FF", "#D3D3D3FF", "#D4D4D4FF", "#D5D5D5FF", "#D6D6D6FF", "#D7D7D7FF", "#D
8D8D8FF", "#D9D9D9FF", "#DADADAFF", "#DBDBDBFF", "#DCDCDCFF", "#DDDDDDFF", "#DEDEDEF
F", "#DEDEDEFF", "#DFDFDFFF", "#E0E0E0FF", "#E0E0E0FF", "#E1E1E1FF", "#E1E1E1FF", "#E2
E2E2FF", "#E2E2E2FF", "#E2E2E2FF")
## alpha     =  0.5
## cex   =  1
## itemLabels    =  TRUE
## labelCol  =  #000000B3
## measureLabels     =  FALSE
## precision     =  3
## layout    =  NULL
## layoutParams  =  list()
## arrowSize     =  0.5
## engine    =  igraph
## plot  =  TRUE
## plot_options  =  list()
## max    =  100
## verbose    =  FALSE
```

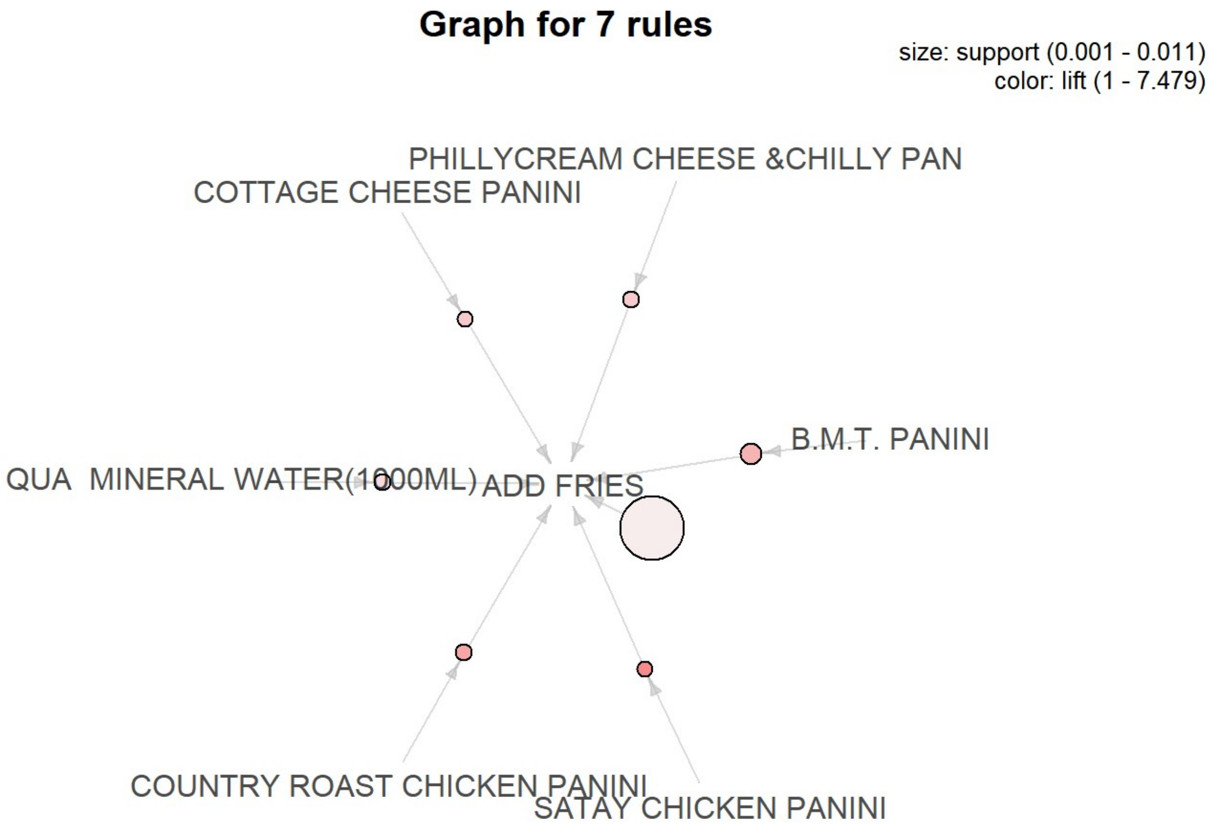


```
top.fv.rules <- head(sort(fv.rules, decreasing = TRUE, by = "lift"), 10)
inspect(top.fv.rules)
```

```
##     lhs                                  rhs         support     confidence
## [1] {SATAY CHICKEN PANINI}            => {ADD FRIES} 0.001128846 0.08036623
## [2] {COUNTRY ROAST CHICKEN PANINI}    => {ADD FRIES} 0.001243159 0.06580938
## [3] {B.M.T. PANINI}                   => {ADD FRIES} 0.002157667 0.05796545
## [4] {COTTAGE CHEESE PANINI}           => {ADD FRIES} 0.001043110 0.04528536
## [5] {PHILLYCREAM CHEESE &CHILLY PAN} => {ADD FRIES} 0.001171713 0.04408602
## [6] {QUA  MINERAL WATER(1000ML)}      => {ADD FRIES} 0.001286027 0.02704327
## [7] {}                                => {ADD FRIES} 0.010745467 0.01074547
##     lift     count
## [1] 7.479082  79
## [2] 6.124385  87
## [3] 5.394410 151
## [4] 4.214369  73
## [5] 4.102755  82
## [6] 2.516714  90
## [7] 1.000000 752
```

```
plot(top.fv.rules, method="graph",
     control=list(type="items"),
     shading = "lift")
```

```
## Warning: Unknown control parameters: type
## Available control parameters (with default values):
## main  =  Graph for 7 rules
## nodeColors    =  c("#66CC6680", "#9999CC80")
## nodeCol   =  c("#EE0000FF", "#EE0303FF", "#EE0606FF", "#EE0909FF", "#EE0C0CFF", "#E
E0F0FFF", "#EE1212FF", "#EE1515FF", "#EE1818FF", "#EE1B1BFF", "#EE1E1EFF", "#EE2222F
F", "#EE2525FF", "#EE2828FF", "#EE2B2BFF", "#EE2E2EFF", "#EE3131FF", "#EE3434FF", "#EE
3737FF", "#EE3A3AFF", "#EE3D3DFF", "#EE4040FF", "#EE4444FF", "#EE4747FF", "#EE4A4AFF",
"#EE4D4DFF", "#EE5050FF", "#EE5353FF", "#EE5656FF", "#EE5959FF", "#EE5C5CFF", "#EE5F5F
FF", "#EE6262FF", "#EE6666FF", "#EE6969FF", "#EE6C6CFF", "#EE6F6FFF", "#EE7272FF", "#E
E7575FF",  "#EE7878FF", "#EE7B7BFF", "#EE7E7EFF", "#EE8181FF", "#EE8484FF", "#EE8888F
F", "#EE8B8BFF", "#EE8E8EFF", "#EE9191FF", "#EE9494FF", "#EE9797FF", "#EE9999FF", "#EE
9B9BFF", "#EE9D9DFF", "#EE9F9FFF", "#EEA0A0FF", "#EEA2A2FF", "#EEA4A4FF", "#EEA5A5FF",
"#EEA7A7FF", "#EEA9A9FF", "#EEABABFF", "#EEACACFF", "#EEAEAEFF", "#EEB0B0FF", "#EEB1B1
FF", "#EEB3B3FF", "#EEB5B5FF", "#EEB7B7FF", "#EEB8B8FF", "#EEBABAFF", "#EEBCBCFF", "#E
EBDBDFF", "#EEBFBFFF", "#EEC1C1FF", "#EEC3C3FF", "#EEC4C4FF", "#EEC6C6FF", "#EEC8C8F
F",  "#EEC9C9FF", "#EECBCBFF", "#EECDCDFF", "#EECFCFFF", "#EED0D0FF", "#EED2D2FF", "#E
ED4D4FF", "#EED5D5FF", "#EED7D7FF", "#EED9D9FF", "#EEDBDBFF", "#EEDCDCFF", "#EEDEDEF
F", "#EEE0E0FF", "#EEE1E1FF", "#EEE3E3FF", "#EEE5E5FF", "#EEE7E7FF", "#EEE8E8FF", "#EE
EAEAFF", "#EEECECFF", "#EEEEEEFF")
## edgeCol   =  c("#474747FF", "#494949FF", "#4B4B4BFF", "#4D4D4DFF", "#4F4F4FFF", "#5
15151FF", "#535353FF", "#555555FF", "#575757FF", "#595959FF", "#5B5B5BFF", "#5E5E5EF
F", "#606060FF", "#626262FF", "#646464FF", "#666666FF", "#686868FF", "#6A6A6AFF", "#6C
6C6CFF", "#6E6E6EFF", "#707070FF", "#727272FF", "#747474FF", "#767676FF", "#787878FF",
"#7A7A7AFF", "#7C7C7CFF", "#7E7E7EFF", "#808080FF", "#828282FF", "#848484FF", "#868686
FF", "#888888FF", "#8A8A8AFF", "#8C8C8CFF", "#8D8D8DFF", "#8F8F8FFF", "#919191FF", "#9
39393FF",  "#959595FF", "#979797FF", "#999999FF", "#9A9A9AFF", "#9C9C9CFF", "#9E9E9EF
F", "#A0A0A0FF", "#A2A2A2FF", "#A3A3A3FF", "#A5A5A5FF", "#A7A7A7FF", "#A9A9A9FF", "#AA
AAAAFF", "#ACACACFF", "#AEAEAEFF", "#AFAFAFFF", "#B1B1B1FF", "#B3B3B3FF", "#B4B4B4FF",
"#B6B6B6FF", "#B7B7B7FF", "#B9B9B9FF", "#BBBBBBFF", "#BCBCBCFF", "#BEBEBEFF", "#BFBFBF
FF", "#C1C1C1FF", "#C2C2C2FF", "#C3C3C4FF", "#C5C5C5FF", "#C6C6C6FF", "#C8C8C8FF", "#C
9C9C9FF", "#CACACAFF", "#CCCCCCFF", "#CDCDCDFF", "#CECECEFF", "#CFCFCFFF", "#D1D1D1F
F",  "#D2D2D2FF", "#D3D3D3FF", "#D4D4D4FF", "#D5D5D5FF", "#D6D6D6FF", "#D7D7D7FF", "#D
8D8D8FF", "#D9D9D9FF", "#DADADAFF", "#DBDBDBFF", "#DCDCDCFF", "#DDDDDDFF", "#DEDEDEF
F", "#DEDEDEFF", "#DFDFDFFF", "#E0E0E0FF", "#E0E0E0FF", "#E1E1E1FF", "#E1E1E1FF", "#E2
E2E2FF", "#E2E2E2FF", "#E2E2E2FF")
## alpha     =  0.5
## cex   =  1
## itemLabels    =  TRUE
## labelCol  =  #000000B3
## measureLabels     =  FALSE
## precision     =  3
## layout    =  NULL
## layoutParams  =  list()
## arrowSize     =  0.5
## engine    =  igraph
## plot  =  TRUE
## plot_options  =  list()
## max    =  100
## verbose    =  FALSE
```


Graph for 7 rules
size: support (0.001 - 0.011)
color: lift (1 - 7.479)
PHILLYCREAM CHEESE &CHILLY PAN
COTTAGE CHEESE PANINI
B.M.T. PANINI
QUA MINERAL WATER(1000ML)
ADD FRIES
COUNTRY ROAST CHICKEN PANINI
SATAY CHICKEN PANINI