

21BCE3892

V.PARDHASARADHI

DIGITAL ASSIGNMENT-1

Market Basket Analysis



1 Introduction(BRIEF)

Market Basket Analysis (MBA) is a data mining technique used in data science to uncover associations between items purchased together. It is widely used in retail and e-commerce industries to understand customer purchasing behavior and improve sales strategies. The primary goal of Market Basket Analysis is to identify relationships between products that are frequently purchased together, enabling businesses to make informed decisions about product placement, promotions, and cross-selling opportunities.

Key components of Market Basket Analysis include:

1. Association Rule Mining: The core of Market Basket Analysis involves discovering association rules, typically in the form of "If {A} then {B}". These rules indicate the likelihood of one product being purchased when another product is bought.

2. Support, Confidence, and Lift: These are metrics used to evaluate the strength of association rules:

- Support: Measures the frequency of co-occurrence of items in transactions.

- Confidence: Measures the likelihood that if item A is purchased, item B will also be purchased.

- Lift: Indicates how much more likely item B is to be bought when item A is purchased, compared to its individual likelihood of being purchased.

3.Frequent Itemsets: These are sets of items that appear together in transactions with a frequency above a specified threshold. Identifying frequent itemsets is the first step in finding association rules.

4. Apriori Algorithm: A popular algorithm used to efficiently mine association rules by iteratively finding frequent itemsets and generating rules from them. Performing a Market Basket Analysis (MBA) project in data science offers several compelling reasons:

WHY THIS PROJECT:

1. ****Optimizing Inventory Management****: MBA can assist businesses in optimizing their inventory management processes by identifying which products tend to be bought together. This insight helps businesses streamline their stock ordering processes, reduce inventory holding costs, and avoid stockouts or overstock situations.

2. ****Personalized Recommendations****: MBA enables businesses to provide personalized recommendations to customers based on their purchase history. By analyzing past transactions, businesses can recommend complementary or relevant products to customers, enhancing the overall shopping experience and driving customer loyalty.

3. **Promotion Planning**: MBA can inform businesses about which products are commonly purchased together during promotional periods. This insight allows businesses to design targeted promotions and discounts that encourage customers to buy related products, ultimately increasing sales and profitability.

4. **Market Basket Optimization**: MBA can help businesses optimize their product assortments and pricing strategies. By analyzing which products are frequently bought together and their associated purchase patterns, businesses can make informed decisions about product bundling, pricing adjustments, and assortment planning to maximize sales and profit margins.

In summary, conducting a Market Basket Analysis project in data science provides businesses with valuable insights into customer behavior, cross-selling opportunities, inventory management optimization, personalized recommendations, promotion planning, market basket optimization, and a competitive advantage in the marketplace. These insights enable businesses to make data-driven decisions that enhance customer satisfaction, drive revenue growth, and improve overall business performance.



2 Association rules

The Apriori algorithm generates association rules for a given data set. An association rule implies that if an item A occurs, then item B also occurs with a certain probability. Let’s see an example,

Transaction	Items
t1	{T-shirt, Trousers, Belt}
t2	{T-shirt, Jacket}
t3	{Jacket, Gloves}
t4	{T-shirt, Trousers, Jacket}
t5	{T-shirt, Trousers, Sneakers, Jacket, Belt}
t6	{Trousers, Sneakers, Belt}
t7	{Trousers, Belt, Sneakers}

In the table above we can see seven transactions from a clothing store. Each transaction shows items bought in that transaction. We can represent our items as an **item set** as follows:

$$I=\{i_1,i_2,...,i_k\}$$

In our case it corresponds to:

$$I=\{T\text{-shirt},Trousers,Belt,Jacket,Gloves,Sneakers\}$$

A **transaction** is represented by the following expression:

$$T=\{t_1,t_2,...,t_n\}$$

Then, an **association rule** is defined as an implication of the form:

$$X \Rightarrow Y, \text{ where } X \subset I, Y \subset I \text{ and } X \cap Y = \emptyset$$

For example,

$$\{T\text{-shirt},Trousers\}\Rightarrow\{Belt\}$$

In the following sections we are going to define four metrics to measure the precision of a rule

2.1 Support

Support is an indication of how frequently the item set appears in the data set.

$$supp(X \Rightarrow Y) = \frac{|X \cup Y|}{n}$$

In other words, it's the number of transactions with both X and Y divided by the total number of transactions. The rules are not useful for low support values. Let's see different examples using the clothing store transactions from the previous table.

- $supp(T\text{-shirt} \Rightarrow Trousers) = \frac{3}{7} = 43\%$
- $supp(Trousers \Rightarrow Belt) = \frac{4}{7} = 57\%$
- $supp(T\text{-shirt} \Rightarrow Belt) = \frac{2}{7} = 28\%$
- $supp(\{T\text{-shirt}, Trousers\} \Rightarrow \{Belt\}) = \frac{2}{7} = 28\%$

2.2 Confidence

For a rule $X \Rightarrow Y$, confidence shows the percentage in which Y is bought with X. It's an indication of how often the rule has been found to be true.

$$conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)}$$

For example, the rule $T\text{-shirt} \Rightarrow Trousers$ has a confidence of 3/4, which means that for 75% of the transactions containing a t-shirt the rule is correct (75% of the times a customer buys a t-shirt, trousers are bought as well). Three more examples:

- $conf(Trousers \Rightarrow Belt) = \frac{4/7}{5/7} = 80\%$
- $conf(T\text{-shirt} \Rightarrow Belt) = \frac{2/7}{4/7} = 50\%$
- $conf(\{T\text{-shirt}, Trousers\} \Rightarrow \{Belt\}) = \frac{2/7}{3/7} = 66\%$

2.3 Lift

The lift of a rule is the ratio of the observed support to that expected if X and Y were independent, and is defined as

$$lift(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)supp(Y)}$$

Greater lift values indicate stronger associations. Let's see some examples:

- $lift(T-shirt \Rightarrow Trousers) = \frac{3/7}{(4/7)(5/7)} = 1.05$
- $lift(Trousers \Rightarrow Belt) = \frac{4/7}{(5/7)(4/7)} = 1.4$
- $lift(T-shirt \Rightarrow Belt) = \frac{2/7}{(4/7)(4/7)} = 0.875$
- $lift(\{T-shirt, Trousers\} \Rightarrow \{Belt\}) = \frac{2/7}{(3/7)(4/7)} = 1.17$

2.4 Conviction

The conviction of a rule is defined as

$$conv(X \Rightarrow Y) = \frac{1 - supp(Y)}{1 - conf(X \Rightarrow Y)}$$

It can be interpreted as the ratio of the expected frequency that X occurs without Y if X and Y were independent divided by the observed frequency of incorrect predictions. A high value means that the consequent depends strongly on the antecedent. Let's see some examples:

- $conv(T-shirt \Rightarrow Trousers) = \frac{1 - 5/7}{1 - 3/4} = 1.14$
- $conv(Trousers \Rightarrow Belt) = \frac{1 - 4/7}{1 - 4/5} = 2.14$
- $conv(T-shirt \Rightarrow Belt) = \frac{1 - 4/7}{1 - 1/2} = 0.86$
- $conv(\{T-shirt, Trousers\} \Rightarrow \{Belt\}) = \frac{1 - 4/7}{1 - 2/3} = 1.28$

3 Loading Data

First we need to load some libraries and import our data. We can use the function `read.transactions()` from the `arules` package to create a `transactions` object.

Hide

```
# Load libraries
```

```
library(tidyverse) # data manipulation
```

```
library(arules) # mining association rules and frequent itemsets
```

```
library(arulesViz) # visualization techniques for association rules
```

```
library(knitr) # dynamic report generation
```

```
library(gridExtra) # provides a number of user-level functions to work with "grid" graphics
```

```
library(lubridate) # work with dates and times
```

```
# Read the data
```

```
trans <- read.transactions("../input/BreadBasket_DMS.csv", format="single", cols=c(3,4), sep=",", rm.duplicates=TRUE)
```

Let's get an idea of what we're working with.

2) Summary

```
## transactions as itemMatrix in sparse format with
```

Glimpse

Structure

Transaction object

4 Data Dictionary

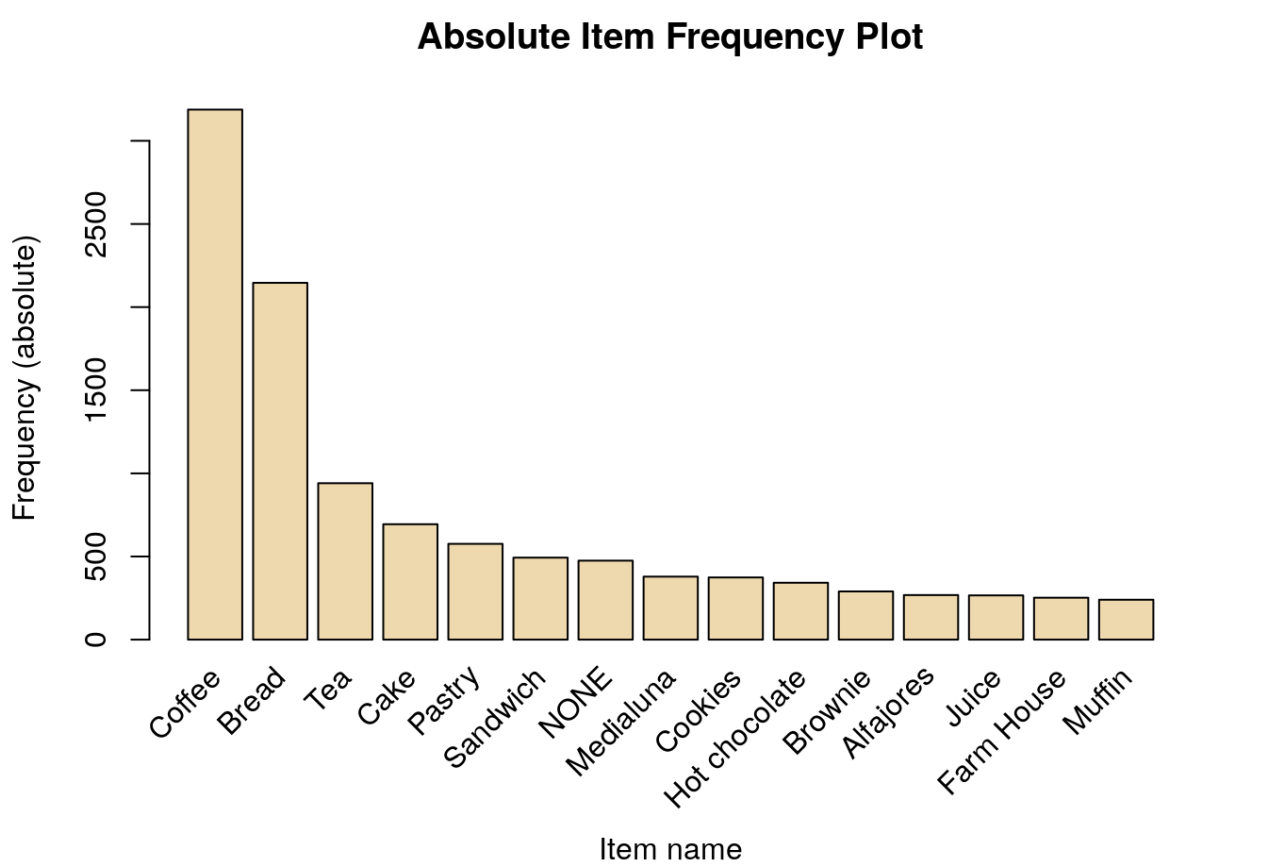
The data set contains 15.010 observations and the following columns,

- Date. Categorical variable that tells us the date of the transactions (YYYY-MM-DD format). The column includes dates from 30/10/2016 to 09/04/2017.
- Time. Categorical variable that tells us the time of the transactions (HH:MM:SS format).
- Transaction. Quantitative variable that allows us to differentiate the transactions. The rows that share the same value in this field belong to the same transaction, that's why the data set has less transactions than observations.
- Item. Categorical variable containing the products.

5 Data Analysis

Before applying the Apriori algorithm on the data set, we are going to show some visualizations to learn more about the transactions. For example, we can use the `itemFrequencyPlot()` function to create an item frequency bar plot, in order to view the distribution of products.

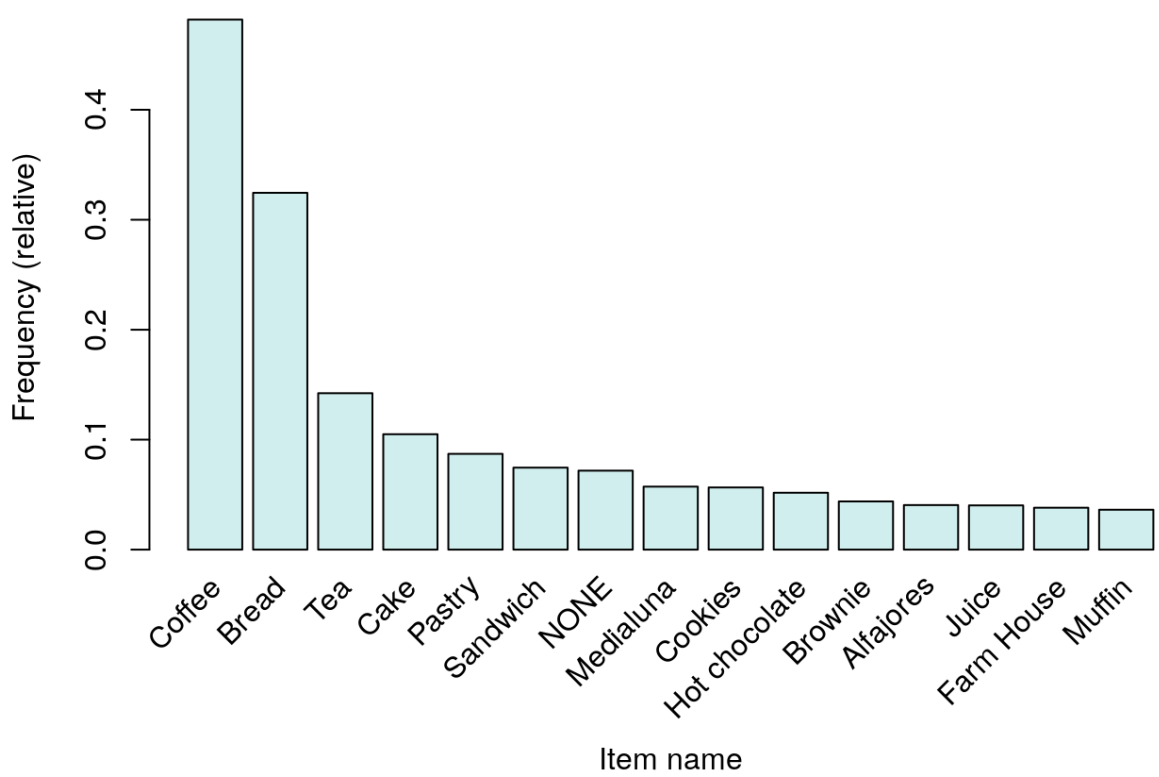
```
# Absolute Item Frequency Plot
itemFrequencyPlot(trans, topN=15, type="absolute", col="wheat2",xlab="Item name",
                  ylab="Frequency (absolute)", main="Absolute Item Frequency Plot"
)
```



The `itemFrequencyPlot()` allows us to show the absolute or relative values. If absolute it will plot numeric frequencies of each item independently. If relative it will plot how many times these items have appeared as compared to others, as it's shown in the following plot.

```
# Relative Item Frequency Plot
itemFrequencyPlot(trans, topN=15, type="relative", col="lightcyan2", xlab="Item name",
                  ylab="Frequency (relative)", main="Relative Item Frequency Plot"
)
```

Relative Item Frequency Plot

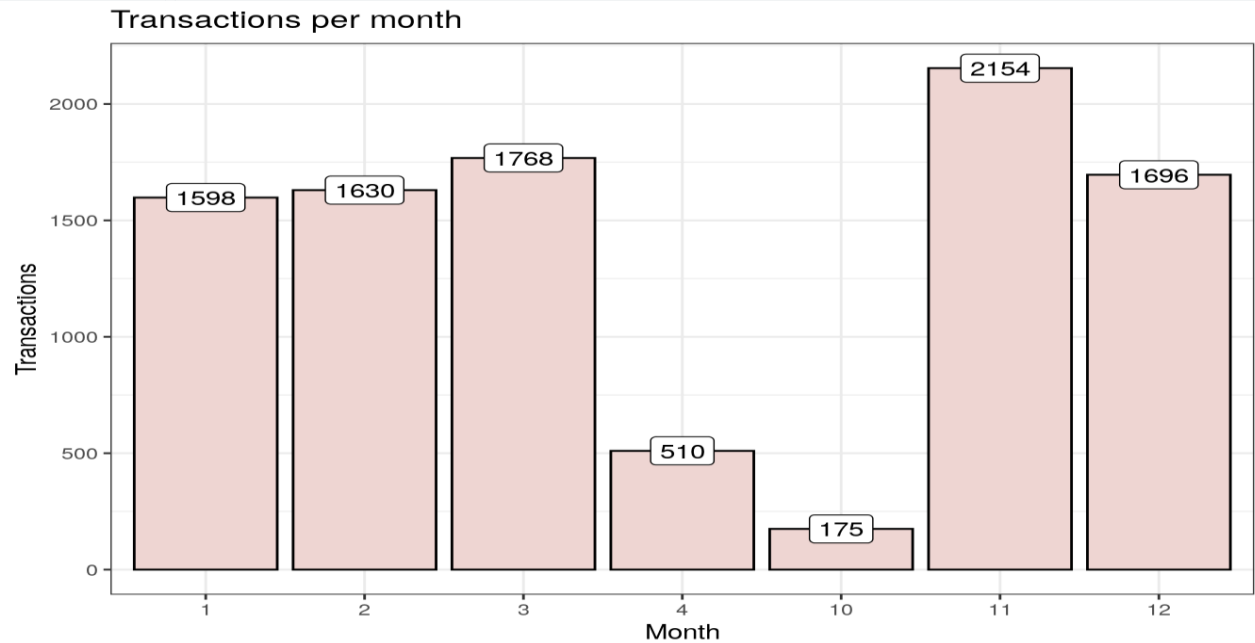


Coffee is the best-selling product by far, followed by bread and tea. Let's display some other visualizations describing the time distribution using the `ggplot()` function.

- Transactions per month
- Transactions per weekday
- Transactions per hour

```
# Load data
trans_csv <- read.csv("../input/BreadBasket_DMS.csv")

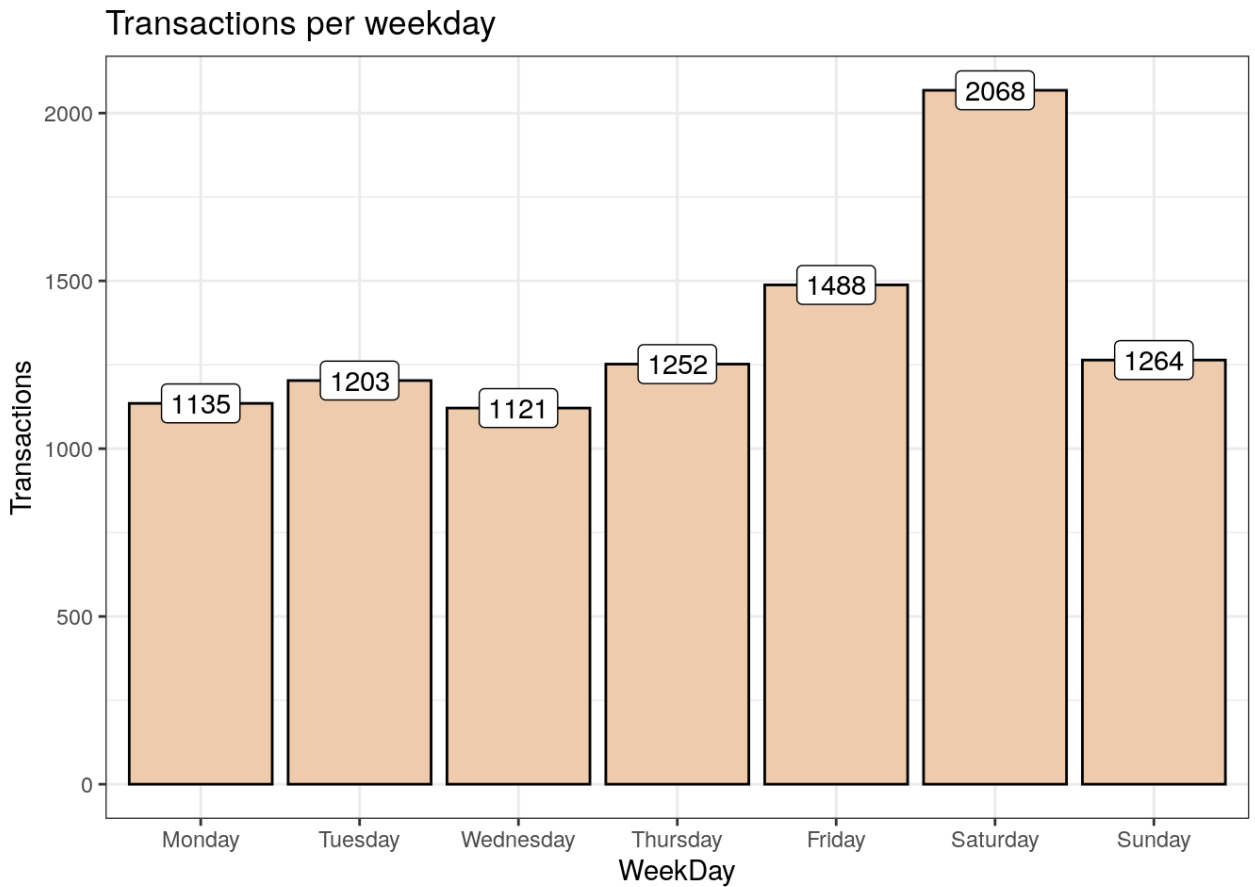
# Visualization - Transactions per month
trans_csv %>%
  mutate(Month=as.factor(month(Date))) %>%
  group_by(Month) %>%
  summarise(Transactions=n_distinct(Transaction)) %>%
  ggplot(aes(x=Month, y=Transactions)) +
  geom_bar(stat="identity", fill="mistyrose2",
           show.legend=FALSE, colour="black") +
  geom_label(aes(label=Transactions)) +
  labs(title="Transactions per month") +
  theme_bw()
```



The data set includes dates from 30/10/2016 to 09/04/2017, that's why we have so few transactions in October and April.

Visualization - Transactions per weekday

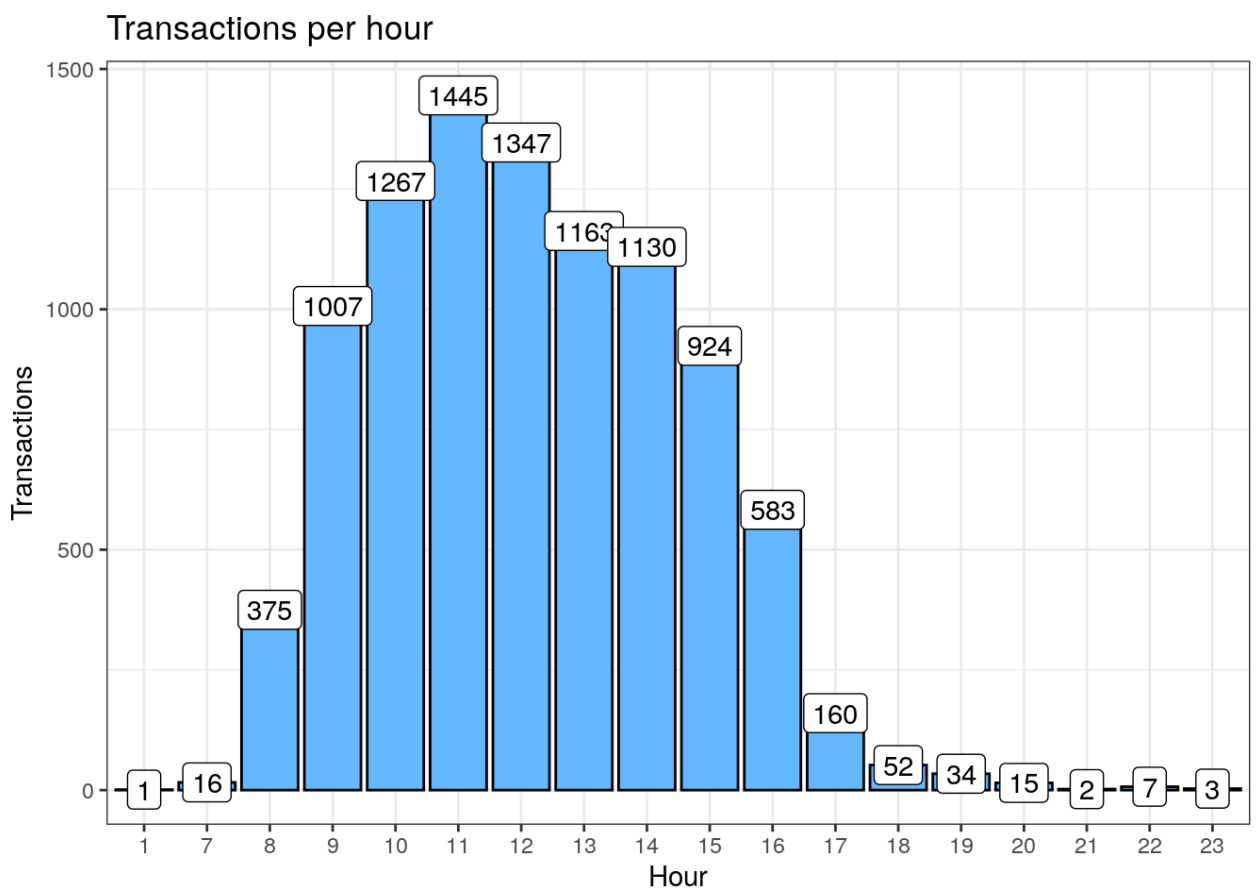
```
trans_csv %>%  
  mutate(WeekDay=as.factor(weekdays(as.Date(Date)))) %>%  
  group_by(WeekDay) %>%  
  summarise(Transactions=n_distinct(Transaction)) %>%  
  ggplot(aes(x=WeekDay, y=Transactions)) +  
  geom_bar(stat="identity", fill="peachpuff2",  
          show.legend=FALSE, colour="black") +  
  geom_label(aes(label=Transactions)) +  
  labs(title="Transactions per weekday") +  
  scale_x_discrete(limits=c("Monday", "Tuesday", "Wednesday", "Thursday",  
                           "Friday", "Saturday", "Sunday")) +  
  theme_bw()
```



As we can see, Saturday is the busiest day in the bakery. Conversely, Wednesday is the day with fewer transactions.

Visualization - Transactions per hour

```
trans_csv %>%  
  mutate(Hour=as.factor(hour(hms(Time)))) %>%  
  group_by(Hour) %>%  
  summarise(Transactions=n_distinct(Transaction)) %>%  
  ggplot(aes(x=Hour, y=Transactions)) +  
  geom_bar(stat="identity", fill="steelblue1", show.legend=FALSE, colour="black")  
+  
  geom_label(aes(label=Transactions)) +  
  labs(title="Transactions per hour") +  
  theme_bw()
```



There's not much to discuss with this visualization. The results are logical and expected.

6 Apriori algorithm

6.1 Choice of support and confidence

The first step in order to create a set of association rules is to determine the optimal thresholds for support and confidence. If we set these values too low, then the algorithm will take longer to execute and we will get a lot of rules (most of them will not be useful). Then, what values do we choose? We can try different values of support and confidence and see graphically how many rules are generated for each combination.

```
# Support and confidence values
supportLevels <- c(0.1, 0.05, 0.01, 0.005)
confidenceLevels <- c(0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1)

# Empty integers
rules_sup10 <- integer(length=9)
rules_sup5 <- integer(length=9)
rules_sup1 <- integer(length=9)
rules_sup0.5 <- integer(length=9)

# Apriori algorithm with a support level of 10%
for (i in 1:length(confidenceLevels)) {

  rules_sup10[i] <- length(apriori(trans, parameter=list(sup=supportLevels[1],
                                                         conf=confidenceLevels[i], target="rules")))

}

# Apriori algorithm with a support level of 5%
for (i in 1:length(confidenceLevels)){

  rules_sup5[i] <- length(apriori(trans, parameter=list(sup=supportLevels[2],
                                                         conf=confidenceLevels[i], target="rules")))

}

# Apriori algorithm with a support level of 1%
for (i in 1:length(confidenceLevels)){
```

```

rules_sup1[i] <- length(apriori(trans, parameter=list(sup=supportLevels[3],
  conf=confidenceLevels[i], target="rules")))
}

# Apriori algorithm with a support level of 0.5%
for (i in 1:length(confidenceLevels)){
  rules_sup0.5[i] <- length(apriori(trans, parameter=list(sup=supportLevels[4],
    conf=confidenceLevels[i], target="rules")))
}

```

In the following graphs we can see the number of rules generated with a support level of 10%, 5%, 1% and 0.5%.

Hide

```

# Number of rules found with a support level of 10%
plot1 <- qplot(confidenceLevels, rules_sup10, geom=c("point", "line"),
  xlab="Confidence level", ylab="Number of rules found",
  main="Apriori with a support level of 10%") +theme_bw()

# Number of rules found with a support level of 5%
plot2 <- qplot(confidenceLevels, rules_sup5, geom=c("point", "line"),
  xlab="Confidence level", ylab="Number of rules found",
  main="Apriori with a support level of 5%") +
  scale_y_continuous(breaks=seq(0, 10, 2)) +theme_bw()

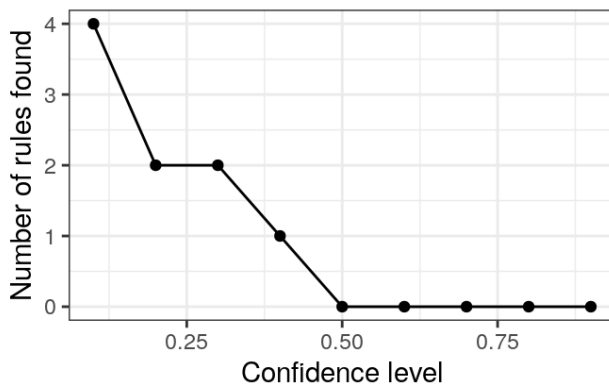
# Number of rules found with a support level of 1%
plot3 <- qplot(confidenceLevels, rules_sup1, geom=c("point", "line"),
  xlab="Confidence level", ylab="Number of rules found",
  main="Apriori with a support level of 1%") +
  scale_y_continuous(breaks=seq(0, 50, 10)) +theme_bw()

# Number of rules found with a support level of 0.5%
plot4 <- qplot(confidenceLevels, rules_sup0.5, geom=c("point", "line"),
  xlab="Confidence level", ylab="Number of rules found",
  main="Apriori with a support level of 0.5%") +
  scale_y_continuous(breaks=seq(0, 130, 20)) +theme_bw()

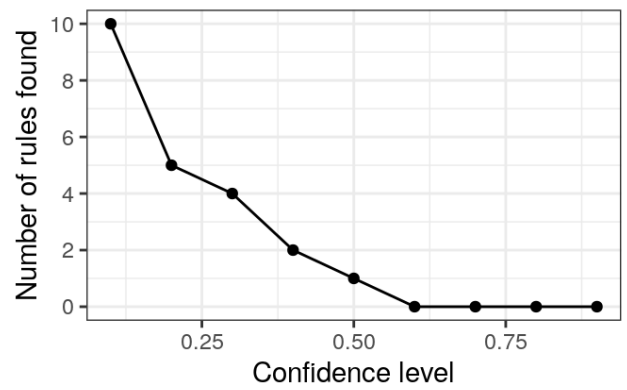
# Subplot
grid.arrange(plot1, plot2, plot3, plot4, ncol=2)

```

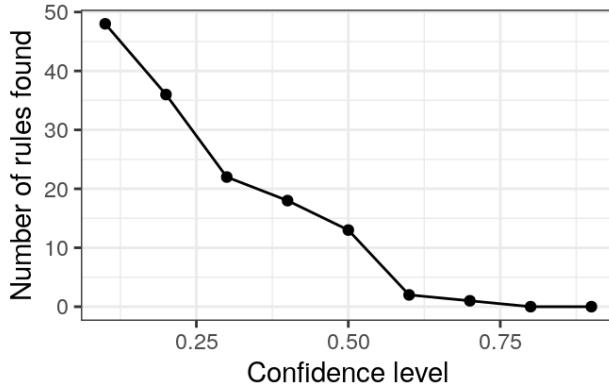
Apriori with a support level of 10%



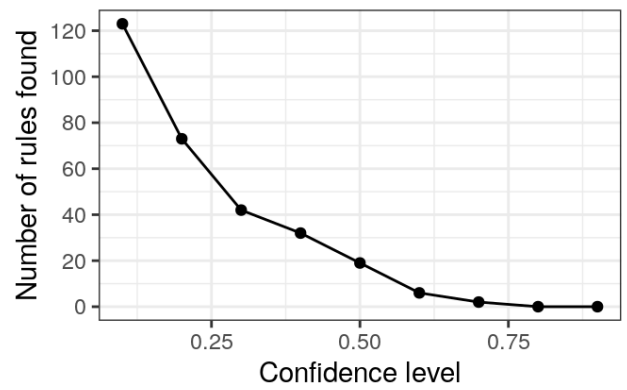
Apriori with a support level of 5%



Apriori with a support level of 1%



Apriori with a support level of 0.5%



We can join the four lines to improve the visualization.

```
# Data frame
num_rules <- data.frame(rules_sup10, rules_sup5, rules_sup1, rules_sup0.5, confidenceLevels)

# Number of rules found with a support level of 10%, 5%, 1% and 0.5%
ggplot(data=num_rules, aes(x=confidenceLevels)) +

  # Plot line and points (support level of 10%)
  geom_line(aes(y=rules_sup10, colour="Support level of 10%")) +
  geom_point(aes(y=rules_sup10, colour="Support level of 10%")) +

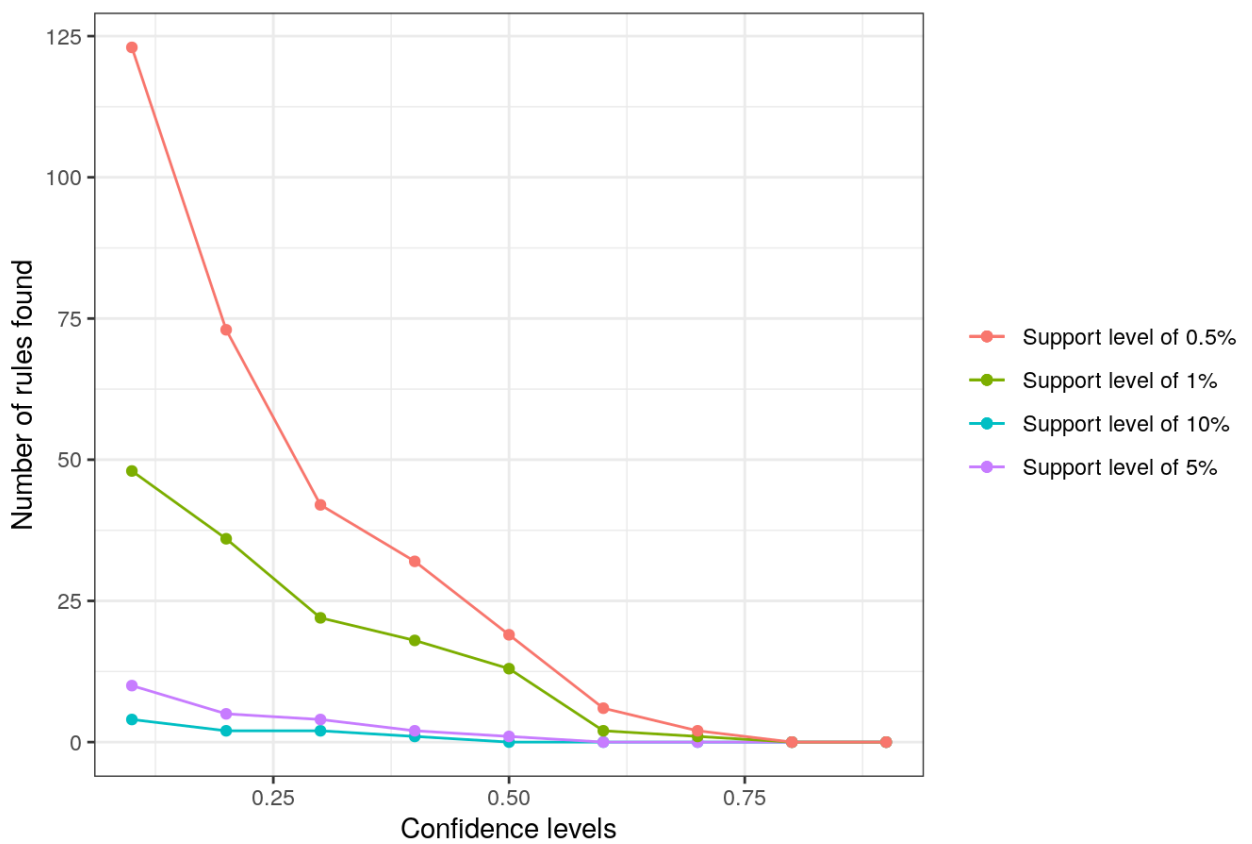
  # Plot line and points (support level of 5%)
  geom_line(aes(y=rules_sup5, colour="Support level of 5%")) +
  geom_point(aes(y=rules_sup5, colour="Support level of 5%")) +

  # Plot line and points (support level of 1%)
  geom_line(aes(y=rules_sup1, colour="Support level of 1%")) +
  geom_point(aes(y=rules_sup1, colour="Support level of 1%")) +

  # Plot line and points (support level of 0.5%)
  geom_line(aes(y=rules_sup0.5, colour="Support level of 0.5%")) +
  geom_point(aes(y=rules_sup0.5, colour="Support level of 0.5%")) +

  # Labs and theme
  labs(x="Confidence levels", y="Number of rules found",
       title="Apriori algorithm with different support levels") +
  theme_bw() +
  theme(legend.title=element_blank())
```

Apriori algorithm with different support levels



Let's analyze the results,

- **Support level of 10%.** We only identify a few rules with very low confidence levels. This means that there are no relatively frequent associations in our data set. We can't choose this value, the resulting rules are unrepresentative.
- **Support level of 5%.** We only identify a rule with a confidence of at least 50%. It seems that we have to look for support levels below 5% to obtain a greater number of rules with a reasonable confidence.
- **Support level of 1%.** We started to get dozens of rules, of which 13 have a confidence of at least 50%.
- **Support level of 0.5%.** Too many rules to analyze!

To sum up, we are going to use a support level of 1% and a confidence level of 50%.

6.2 Execution

Let's execute the Apriori algorithm with the values obtained in the previous section.

Apriori algorithm execution with a support level of 1% and a confidence level of 50%

```
rules_sup1_conf50 <- apriori(trans, parameter=list(sup=supportLevels[3],  
                                                  conf=confidenceLevels[5], target="rules"))
```

The generated association rules are the following,

Inspect association rules

```
inspect(rules_sup1_conf50)
```

##	lhs	rhs	support	confidence	lift	count
## [1]	{Tiffin}	=> {Coffee}	0.01058361	0.5468750	1.134577	70
## [2]	{Spanish Brunch}	=> {Coffee}	0.01406108	0.6326531	1.312537	93
## [3]	{Scone}	=> {Coffee}	0.01844572	0.5422222	1.124924	122
## [4]	{Toast}	=> {Coffee}	0.02570305	0.7296137	1.513697	170
## [5]	{Alfajores}	=> {Coffee}	0.02237678	0.5522388	1.145705	148
## [6]	{Juice}	=> {Coffee}	0.02131842	0.5300752	1.099723	141
## [7]	{Hot chocolate}	=> {Coffee}	0.02721500	0.5263158	1.091924	180
## [8]	{Medialuna}	=> {Coffee}	0.03296039	0.5751979	1.193337	218
## [9]	{Cookies}	=> {Coffee}	0.02978530	0.5267380	1.092800	197
## [10]	{NONE}	=> {Coffee}	0.04172966	0.5810526	1.205484	276
## [11]	{Sandwich}	=> {Coffee}	0.04233444	0.5679513	1.178303	280
## [12]	{Pastry}	=> {Coffee}	0.04868461	0.5590278	1.159790	322
## [13]	{Cake}	=> {Coffee}	0.05654672	0.5389049	1.118042	374

We can also create an HTML table widget using the `inspectDT()` function from the `arulesViz` package. Rules can be interactively filtered and sorted.

How do we interpret these rules?

- 52% of the customers who bought a hot chocolate also bought a coffee.
- 63% of the customers who bought a spanish brunch also bought a coffee.
- 73% of the customers who bought a toast also bought a coffee.

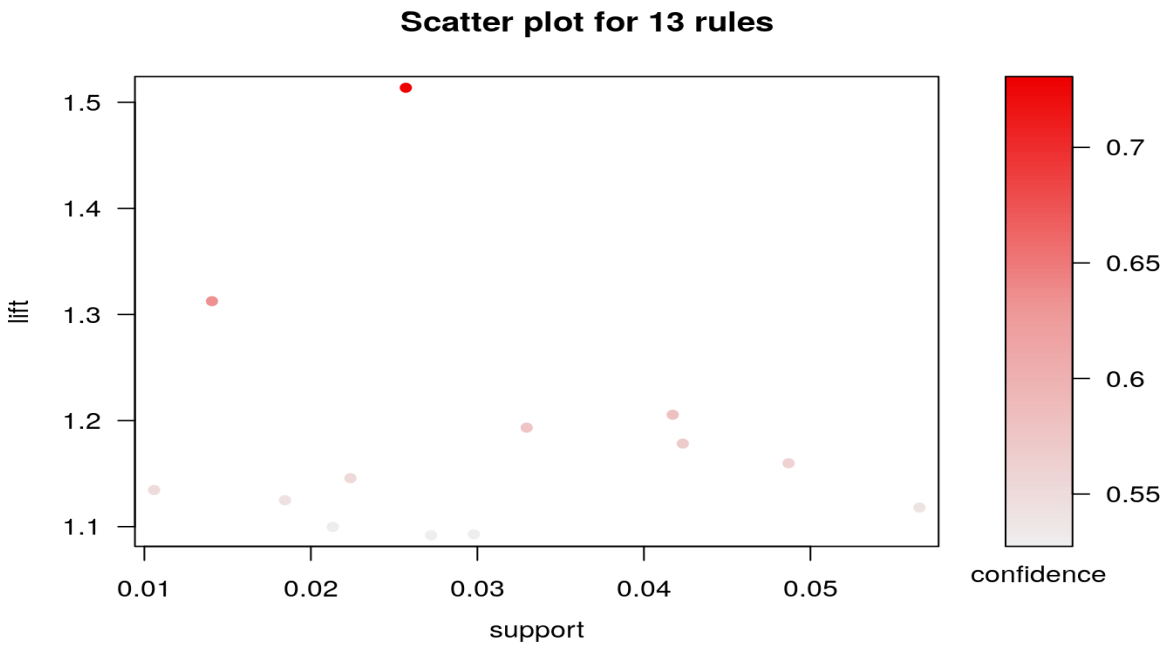
And so on. It seems that in this bakery there are many coffee lovers!

6.3 Visualize association rules

We are going to use the `arulesViz` package to create the visualizations. Let's begin with a simple scatter plot with different measures of interestingness on the axes (lift and support) and a third measure (confidence) represented by the color of the points.

Scatter plot

```
plot(rules_sup1_conf50, measure=c("support", "lift"), shading="confidence")
```



The following visualization represents the rules as a graph with items as labeled vertices, and rules represented as vertices connected to items using arrows.

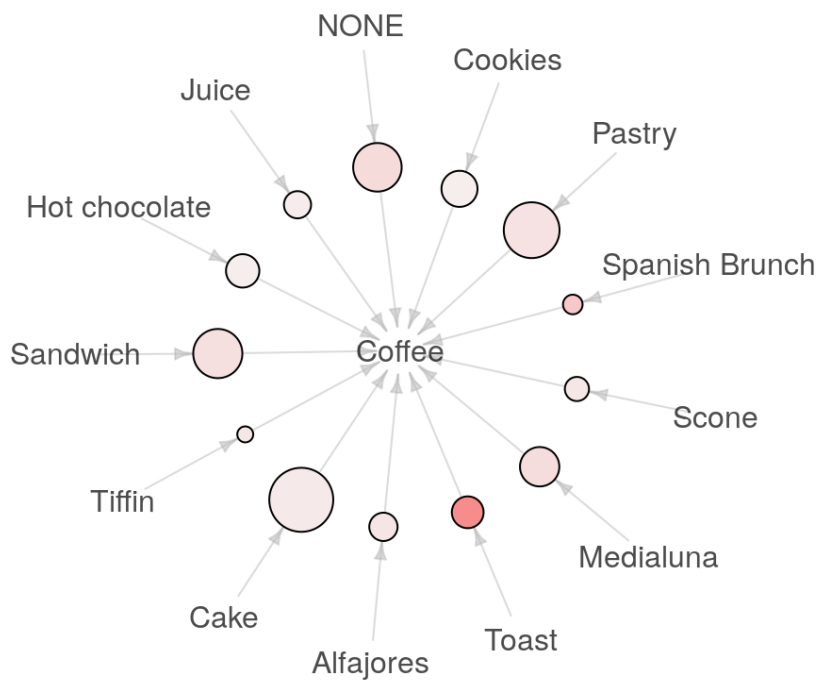
Hide

Graph (default Layout)

```
plot(rules_sup1_conf50, method="graph")
```

Graph for 13 rules

size: support (0.011 - 0.057)
color: lift (1.092 - 1.514)



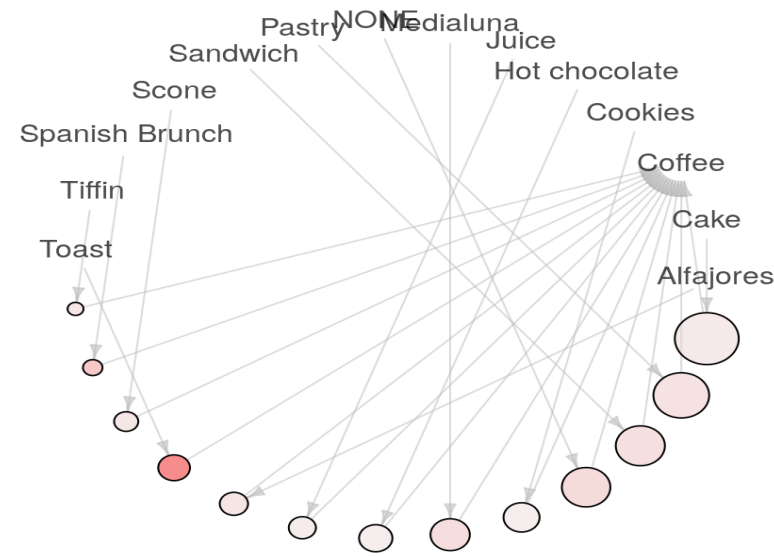
We can also change the graph layout.

Hide

```
# Graph (circular layout)
plot(rules_sup1_conf50, method="graph", control=list(layout=igraph::in_circle()))
```

Graph for 13 rules

size: support (0.011 - 0.057)
color: lift (1.092 - 1.514)

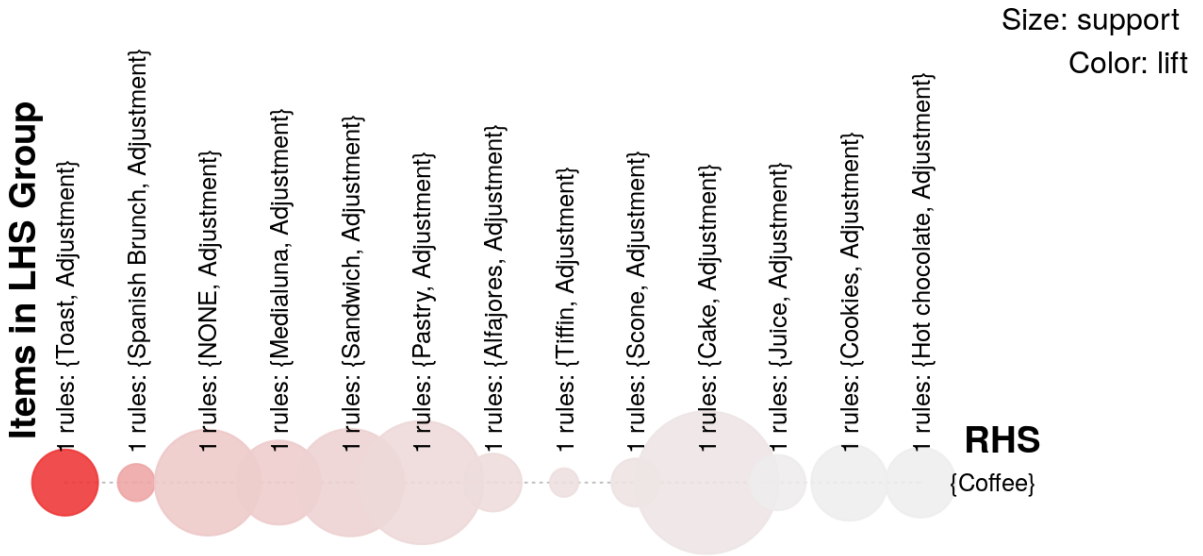


What else can we do? We can represent the rules as a grouped matrix-based visualization. The support and lift measures are represented by the size and color of the balloons, respectively. In this case it's not a very useful visualization, since we only have coffee on the right-hand-side of the rules.

Hide

```
# Grouped matrix plot
plot(rules_sup1_conf50, method="grouped")
```


Grouped Matrix for 13 Rules



There's an awesome function called `ruleExplorer()` that explores association rules using interactive manipulations and visualization using shiny. Unfortunately, R Markdown still doesn't support shiny app objects.

6.4 Another execution

We have executed the Apriori algorithm with the appropriate support and confidence values. What happens if we execute it with low values? How do the visualizations change? Let's try with a support level of 0.5% and a confidence level of 10%.

Hide

```
# Apriori algorithm execution with a support level of 0.5% and a confidence level of 10%
```

```
rules_sup0.5_conf10 <- apriori(trans, parameter=list(sup=supportLevels[4], conf=confidenceLevels[9], target="rules"))
```

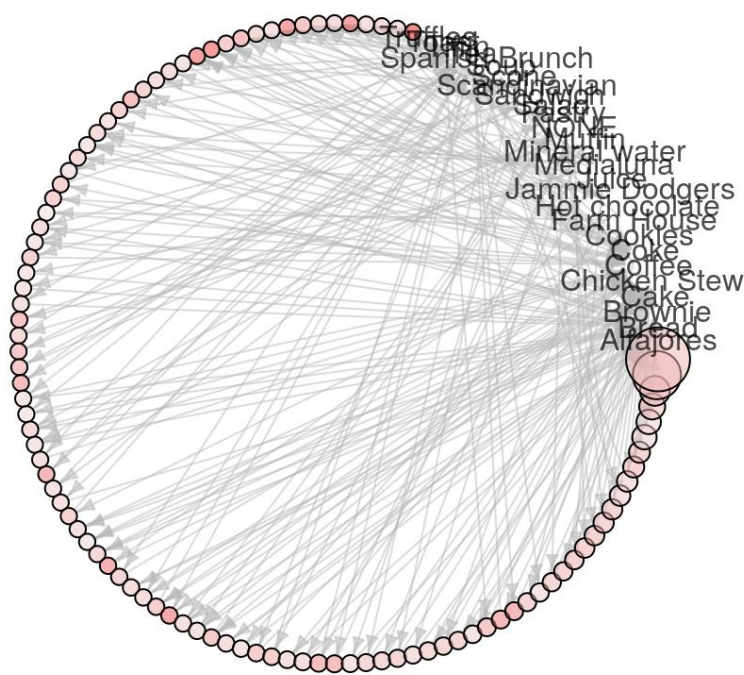
It's impossible to analyze these visualizations! For larger rule sets visual analysis becomes difficult. Furthermore, most of the rules are useless. That's why we have to carefully select the right values of support and confidence.

```
# Graph (circular layout)
```

```
plot(rules_sup0.5_conf10, method="graph", control=list(layout=igraph::in_circle()))
```

Graph for 100 rules

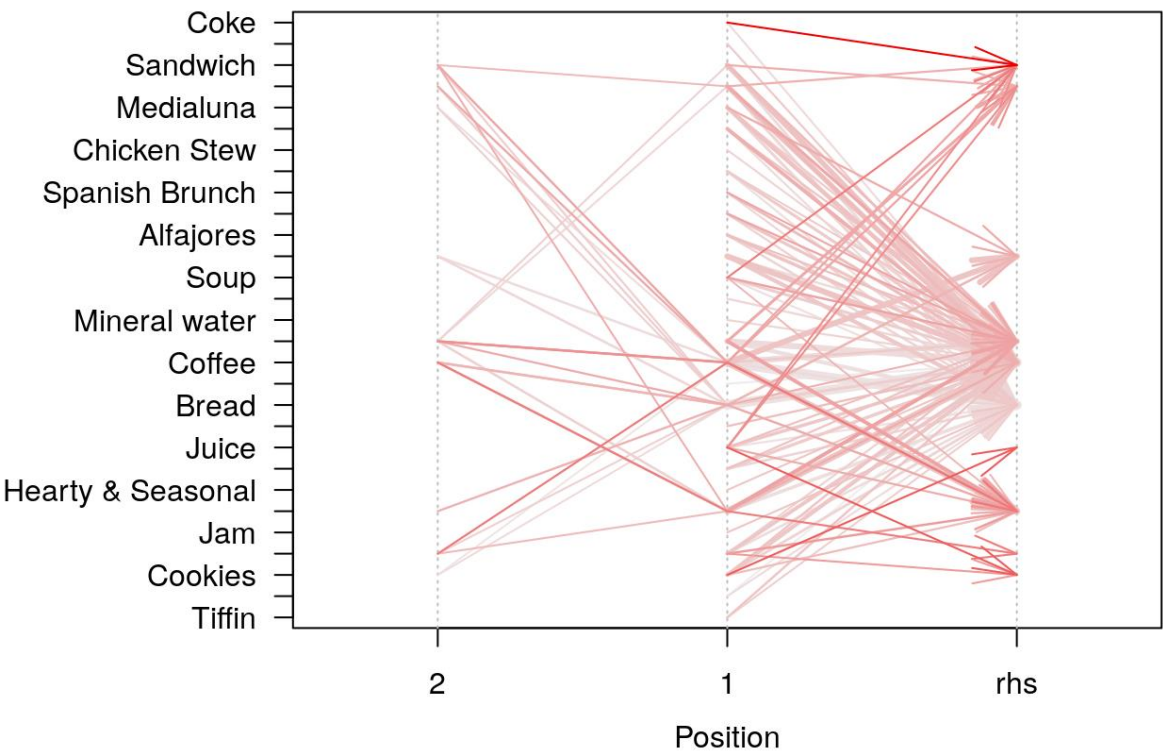
size: support (0.006 - 0.482)
color: lift (0.42 - 2.593)



Parallel coordinates plot

```
# Parallel coordinates plot
plot(rules_sup0.5_conf10, method="paracoord", control=list(reorder=TRUE))
```

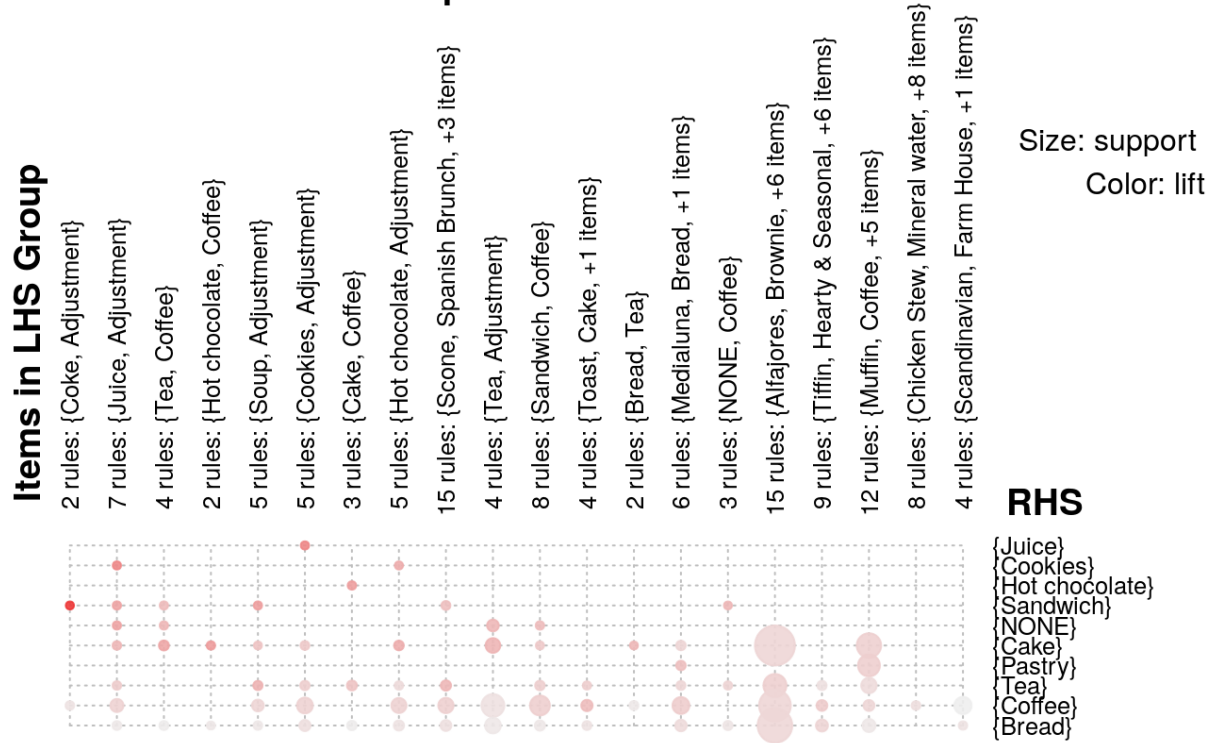
Parallel coordinates plot for 123 rules



Group matrix plot

```
# Grouped matrix plot
plot(rules_sup0.5_conf10, method="grouped")
```

Grouped Matrix for 123 Rules



Scatter plot

```
# Scatter plot
plot(rules_sup0.5_conf10, measure=c("support", "lift"), shading="confidence", jitter=0)
```

Scatter plot for 123 rules

