MARKET BASKET INSIGHT

Phase 3 : DEVELOPMENT PART1

1 Introduction

Hi! In this kernel we are going to use the **Apriori algorithm** to perform a **Market Basket Analysis**. A Market what? Is a technique used by large retailers to uncover associations between items. It works by looking for combinations of items that occur together frequently in transactions, providing information to understand the purchase behavior. The outcome of this type of technique is, in simple terms, a set of **rules** that can be understood as **“if this, then that”**. For more information about these topics, please check in the following links:

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,

In the table above we can see seven transactions from a clothing store. Each transaction

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

shows items bought in that transaction. We can represent our items as an **item set** as follows:

$I=\{i_1,i_2,\ldots,i_k\}$�={�1,�2,...,��}

In our case it corresponds to:

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

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

$T=\{t_1,t_2,\ldots,t_n\}$�={�1,�2,...,��}

For example,

$$t_1=\{\text{T-shirt},\text{Trousers},\text{Belt}\}$$ �1={�-�ℎ ���,��������,����}

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

$X \Rightarrow Y$ �⇒�, where $X \subset I$ �⊂�, $Y \subset I$ �⊂� and $X \cap Y = 0$ �∩�=0

For example,

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

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

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.

Code

```
# 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)
```

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.

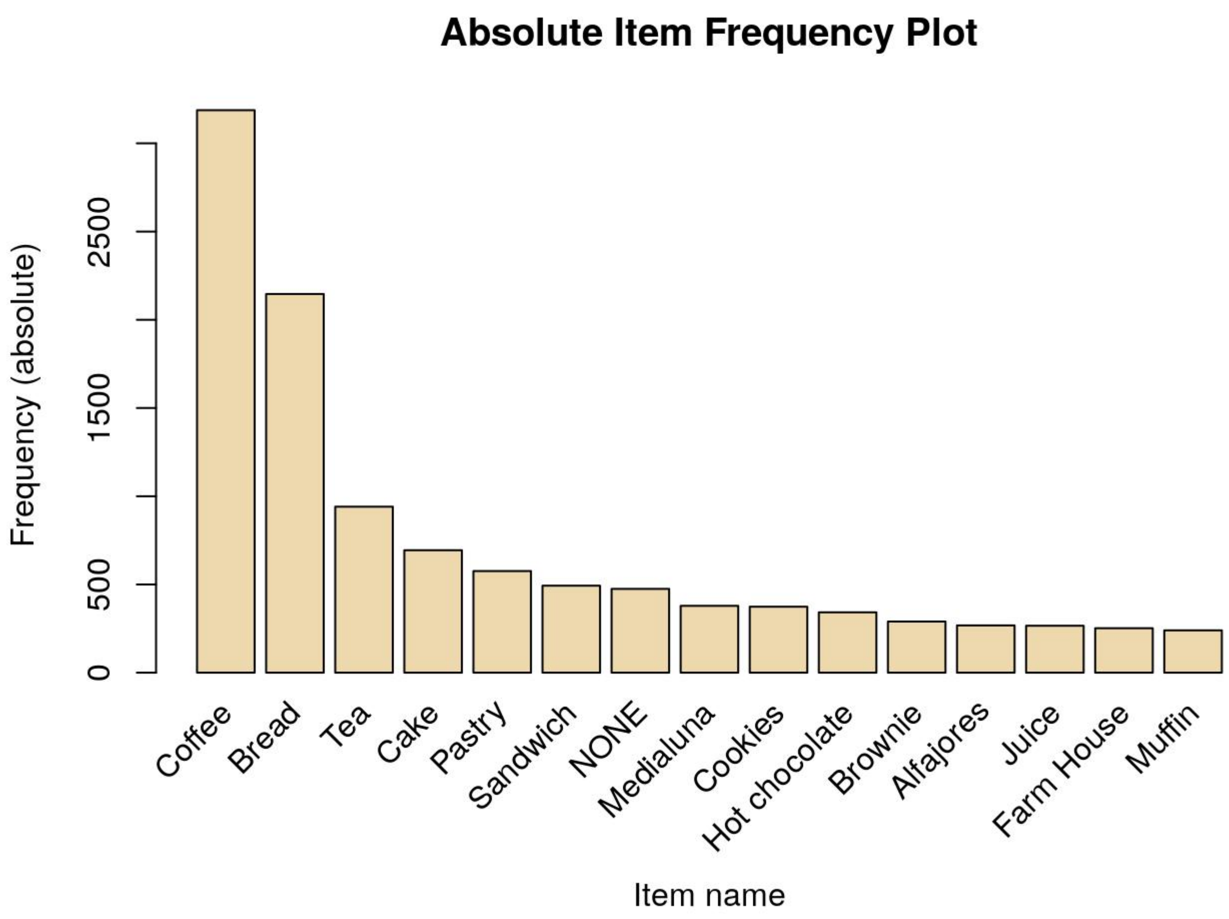
5 Data Analysis

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.

Code

```
# Absolute Item Frequency Plot

itemFrequencyPlot(trans, topN=15, type="absolute", col="wheat2",xlab="Item name",

              ylab="Frequency (absolute)", main="Absolute Item Frequency Plot")
```

OUTPUT:

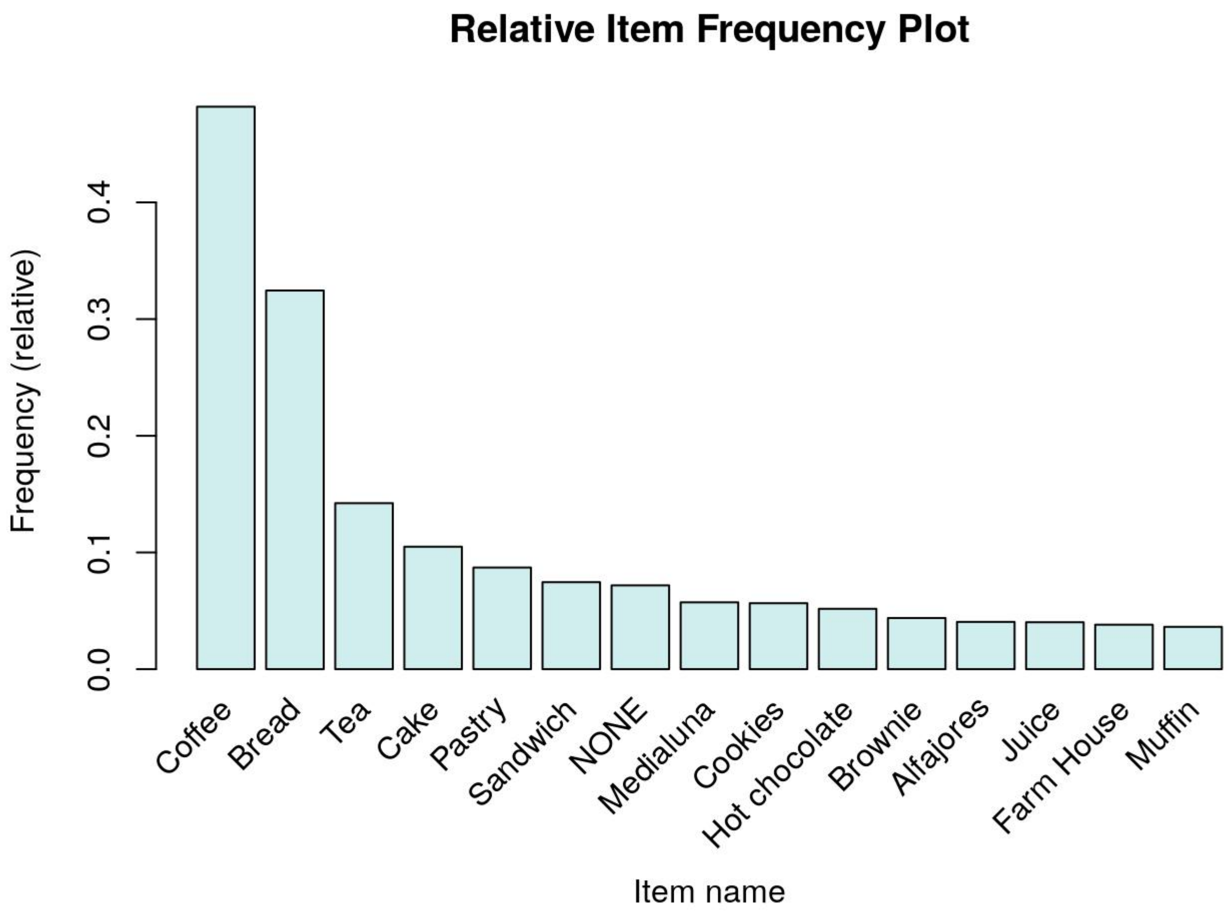


The itemFrequencyPlot() allows us to show the absolute or relative values. If absolute it will plot numeric frequencies of each item independently. If relative it will plot how many times these items have appeared as compared to others, as it's shown in the following plot.

Code

```
# Relative Item Frequency Plot

itemFrequencyPlot(trans, topN=15, type="relative", col="lightcyan2", xlab="Item name",

              ylab="Frequency (relative)", main="Relative Item Frequency Plot")
```

OUTPUT:



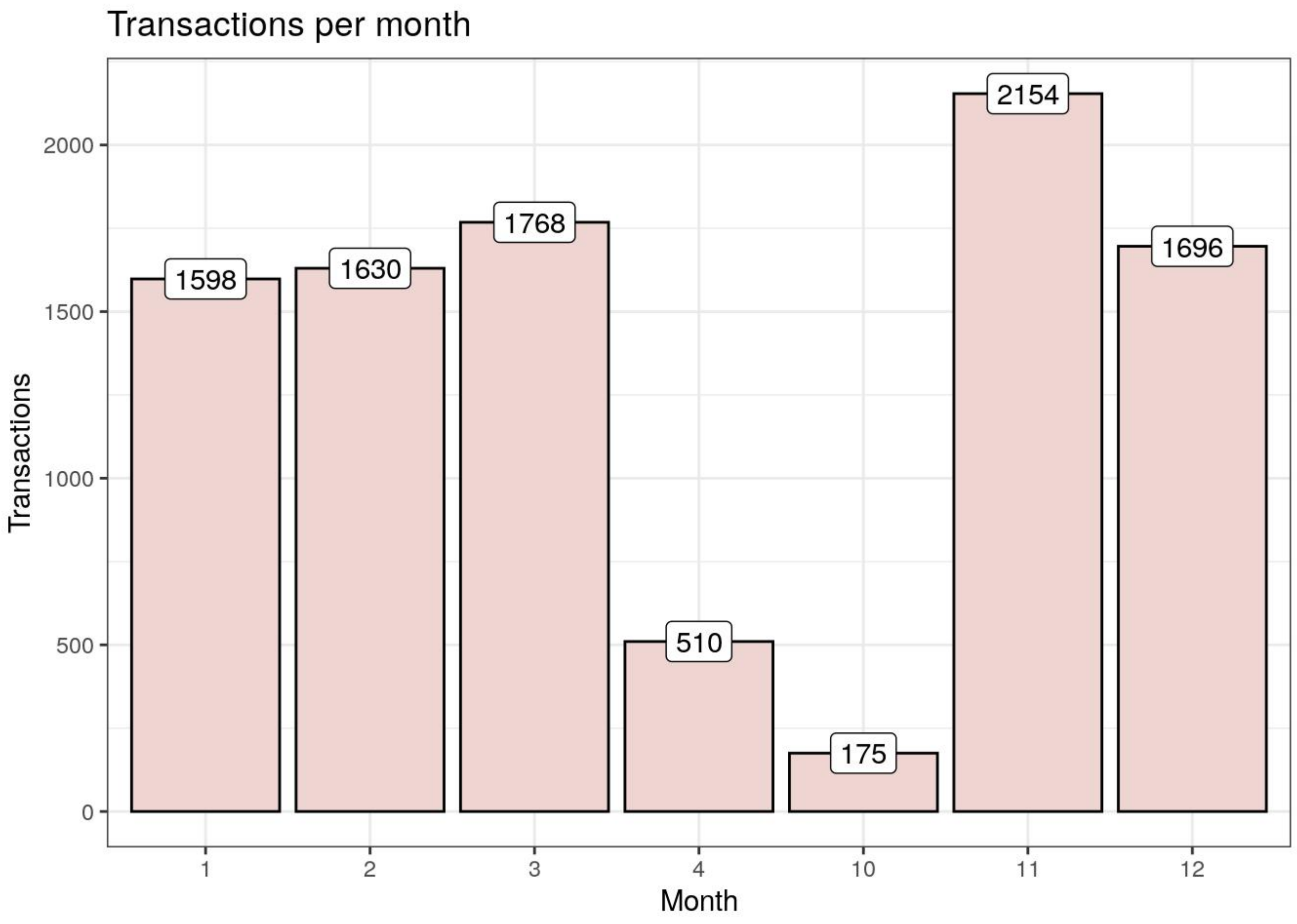
Coffee is the best-selling product by far, followed by bread and tea. Let's display some other visualizations describing the time distribution using the ggplot() function.

Code

```
# 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()
```

OUTPUT:

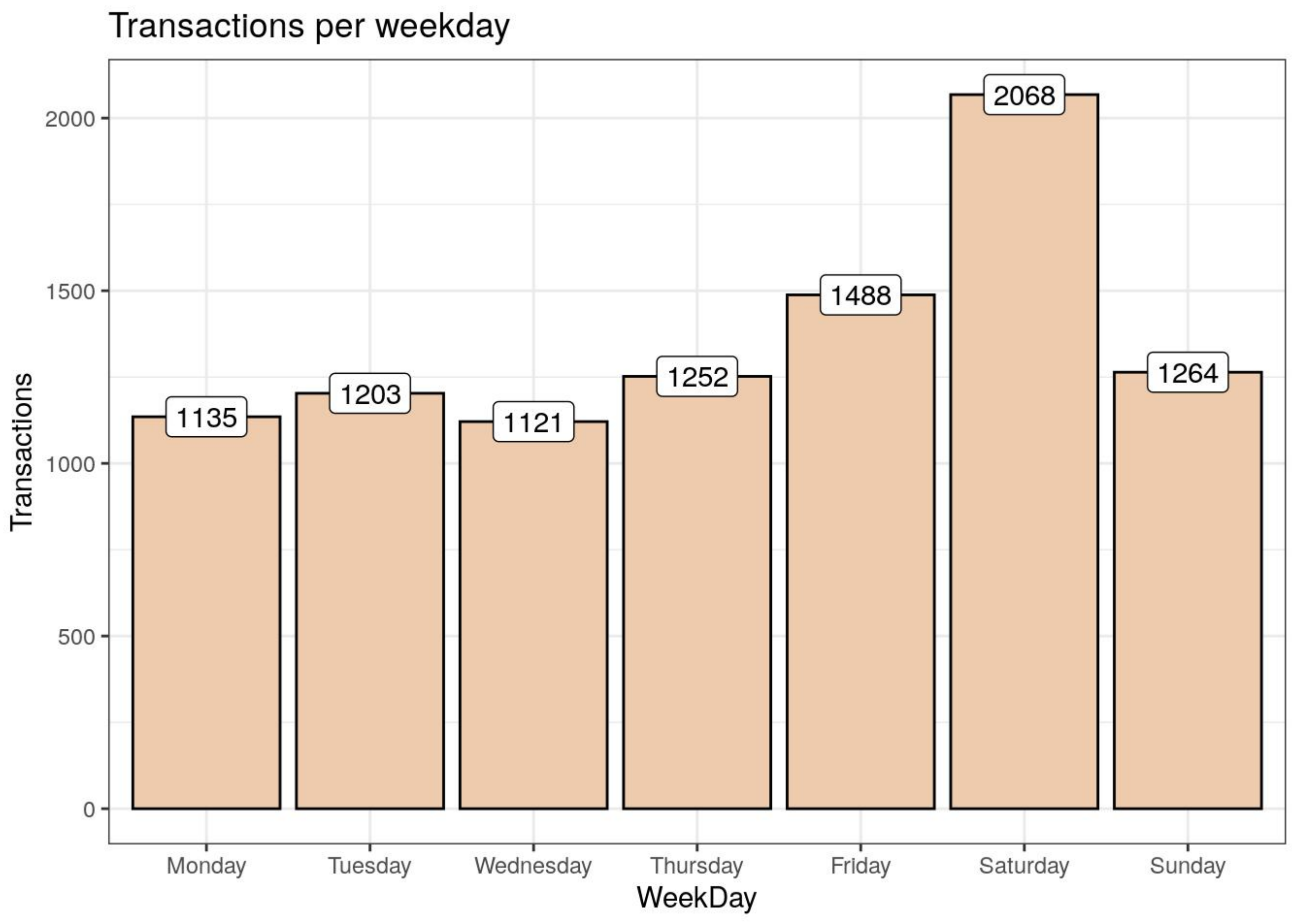


The data set includes dates from 30/10/2016 to 09/04/2017, that’s why we have so few transactions in October and April.

Code

```
# 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()
```

OUTPUT:

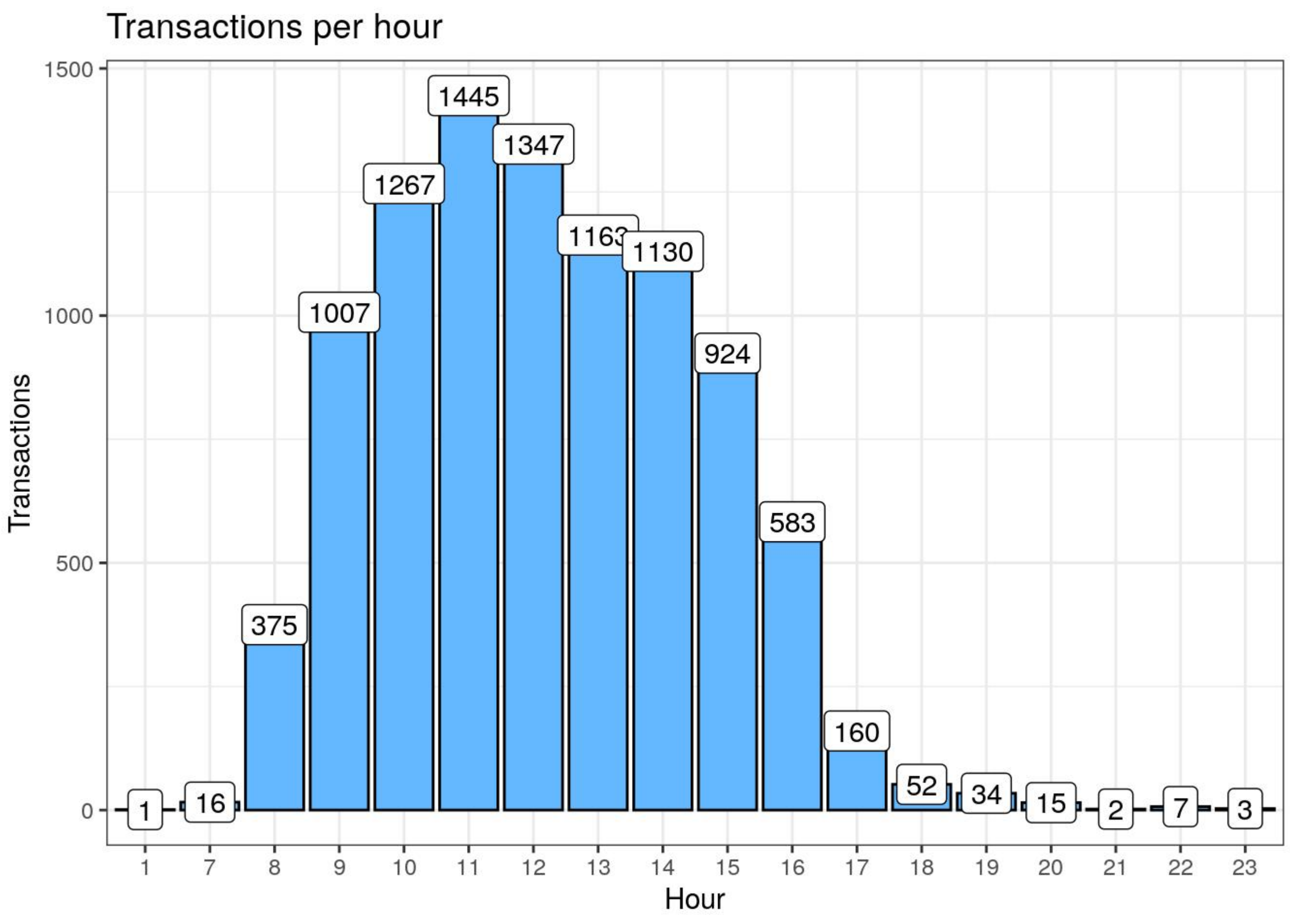


As we can see, Saturday is the busiest day in the bakery. Conversely, Wednesday is the day with fewer transactions.

Code

```
# 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()
```

OUTPUT:



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

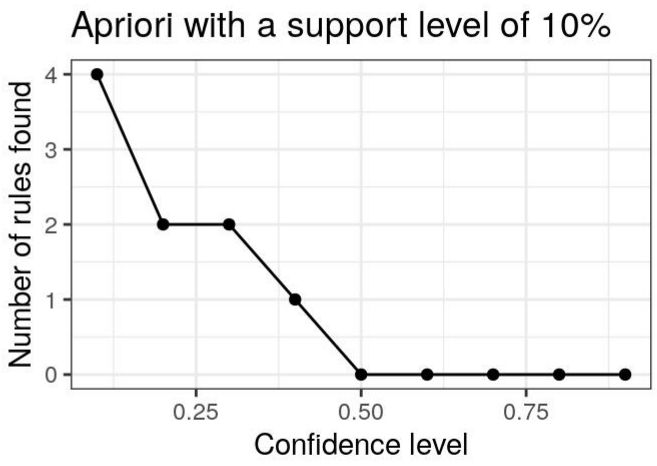
The first step in order to create a set of association rules is to determine the optimal thresholds for support and confidence. If we set these values too low, then the algorithm will take longer to execute and we will get a lot of rules (most of them will not be useful). Then, what values do we choose?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

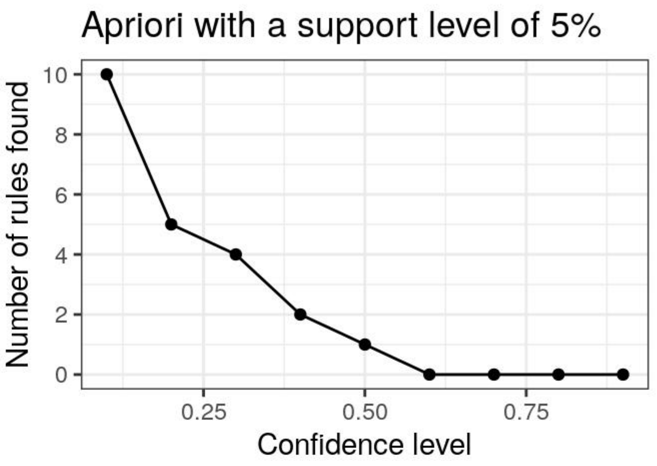
Code

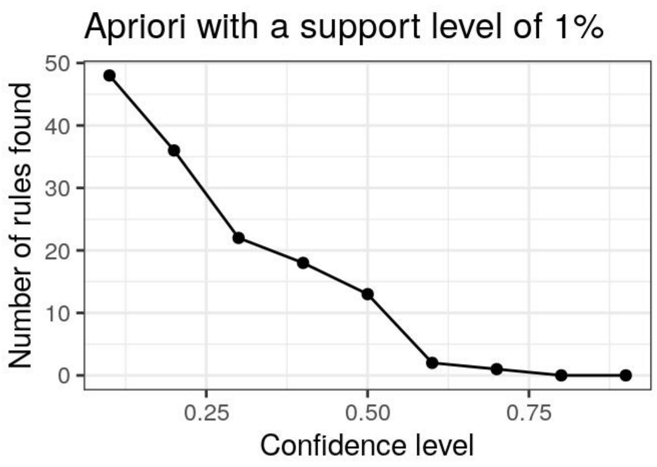
```
# Support and confidence values

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")))


}
```

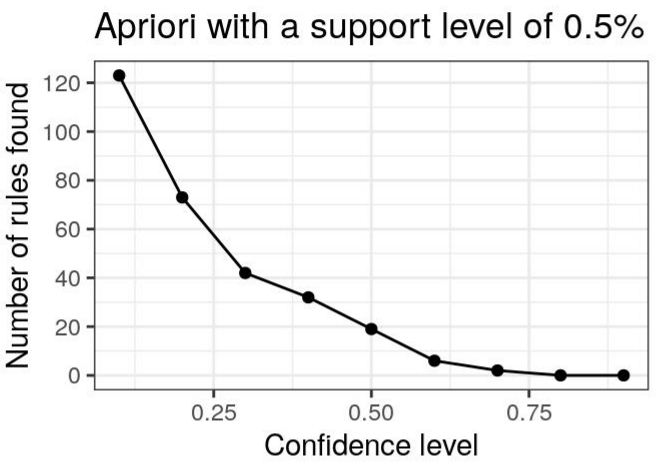
In the following graphs we can see the number of rules generated with a support level of 10%, 5%, 1% and 0.5%.Code

OUTPUT;







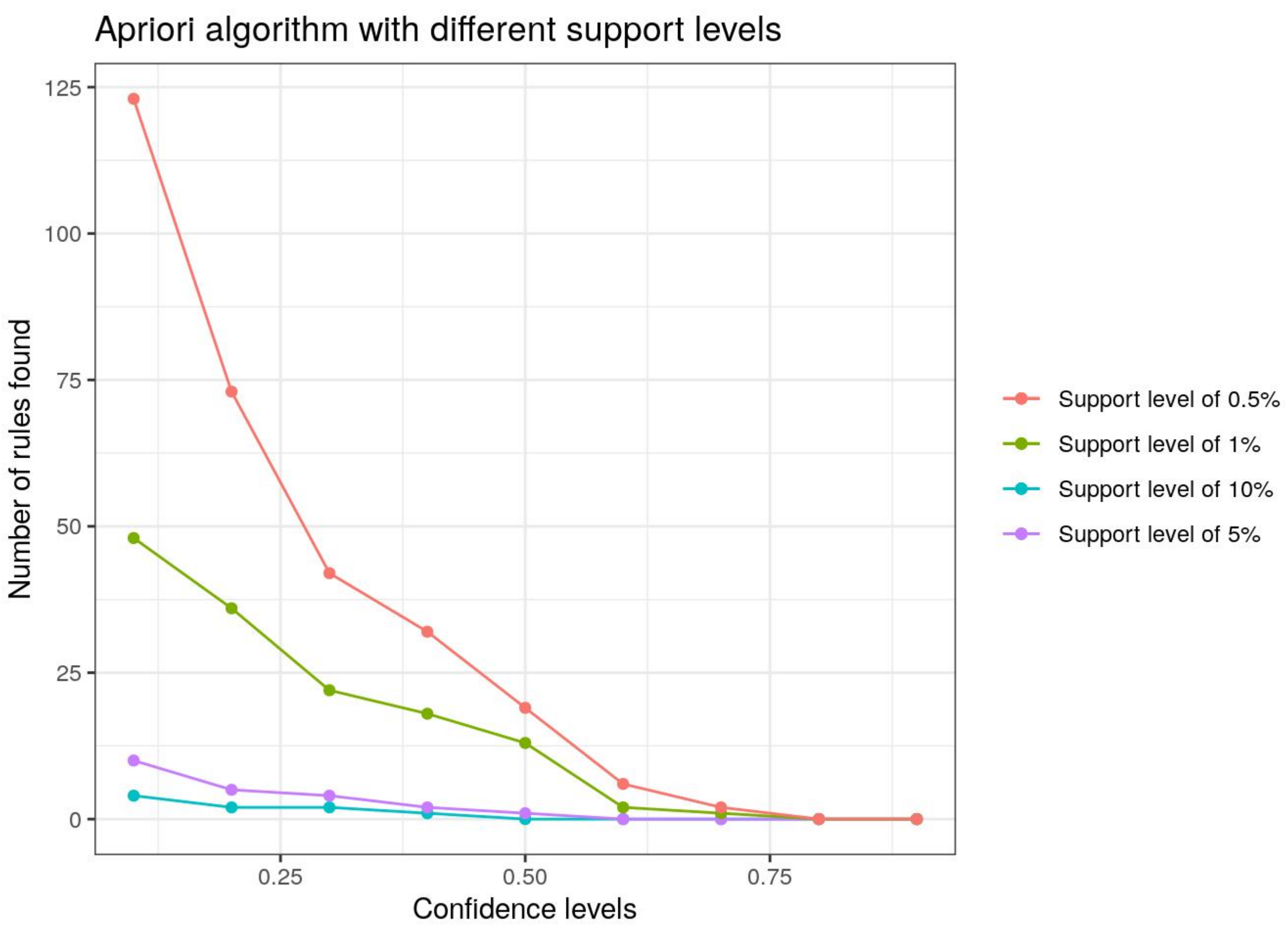


We can join the four lines to improve the visualization.

Code

```
# 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())
```

OUTPUT:



Let's analyze the results,

6.2 Execution

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

Code

```
# 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,

Code

```
# Inspect association rules
inspect(rules_sup1_conf50)
```

OUTPUT:

```
## 	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 aruslesViz package. Rules can be interactively filtered and sorted.

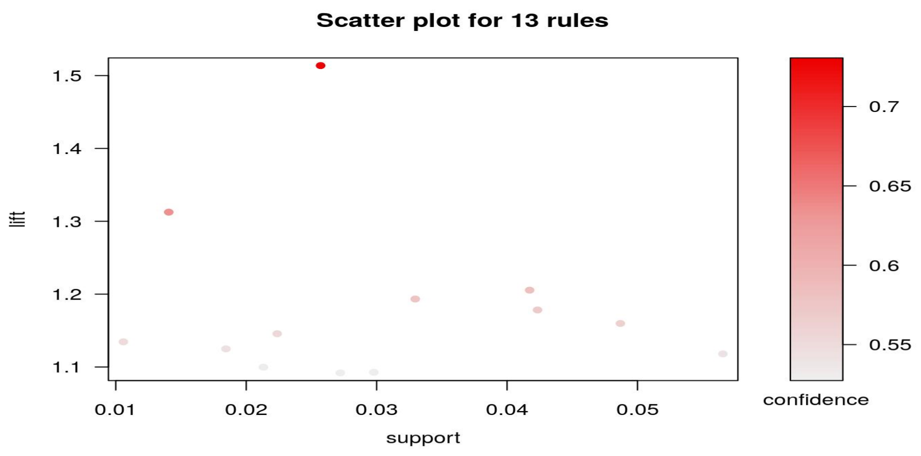
6.3 Visualize association rules

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.

Code

```
# Scatter plot

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

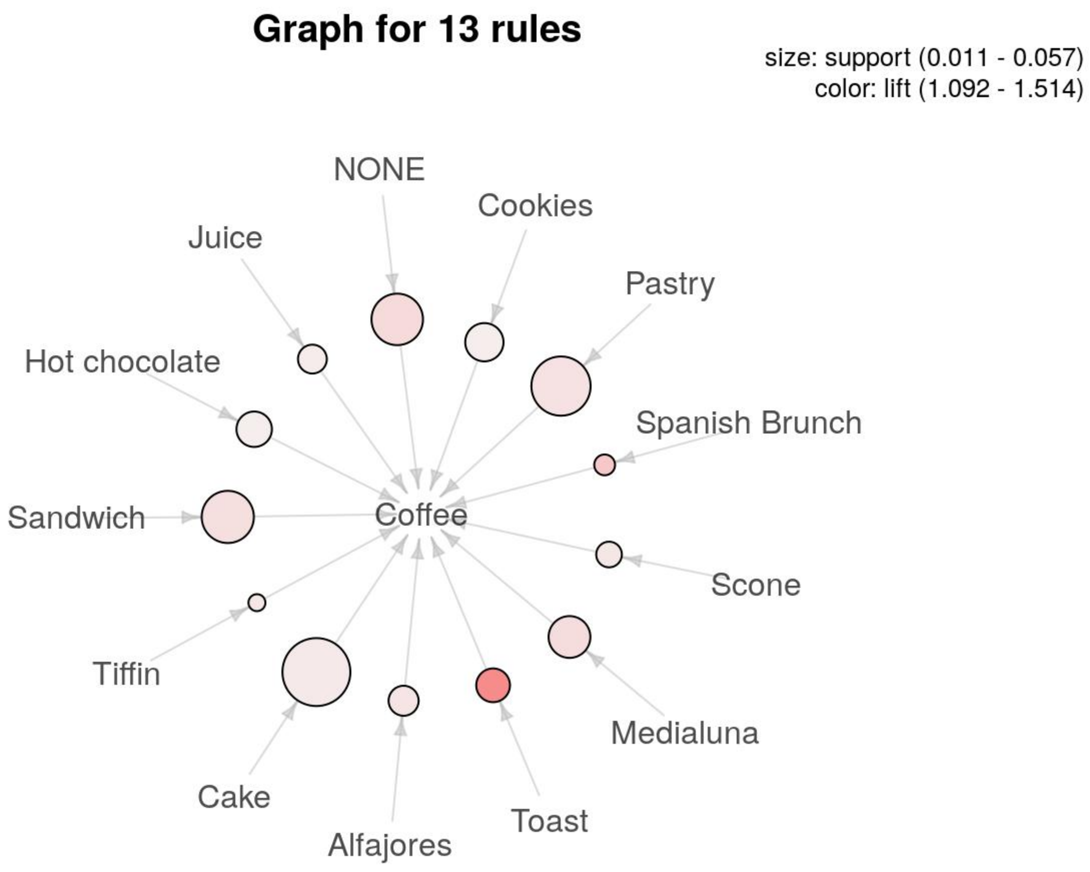
OUTPUT:



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

Code

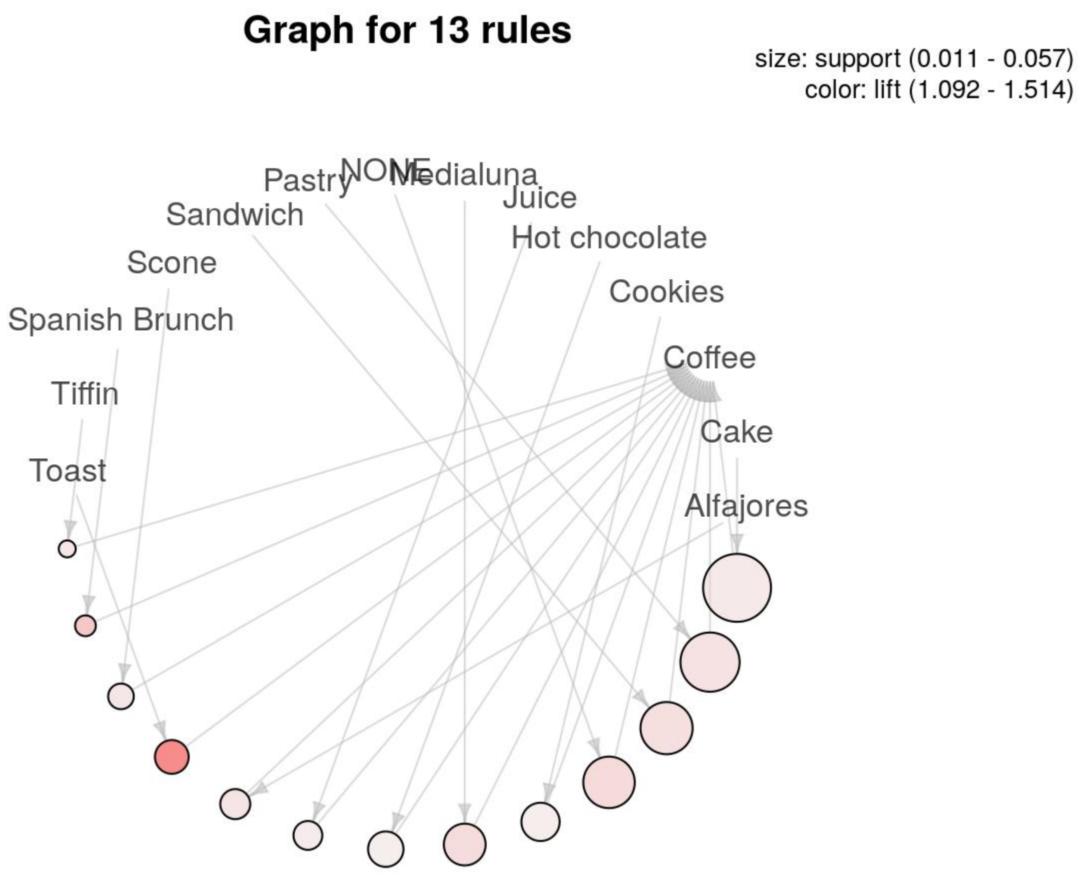
```
# Graph (default layout)

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



We can also change the graph layout.

Code

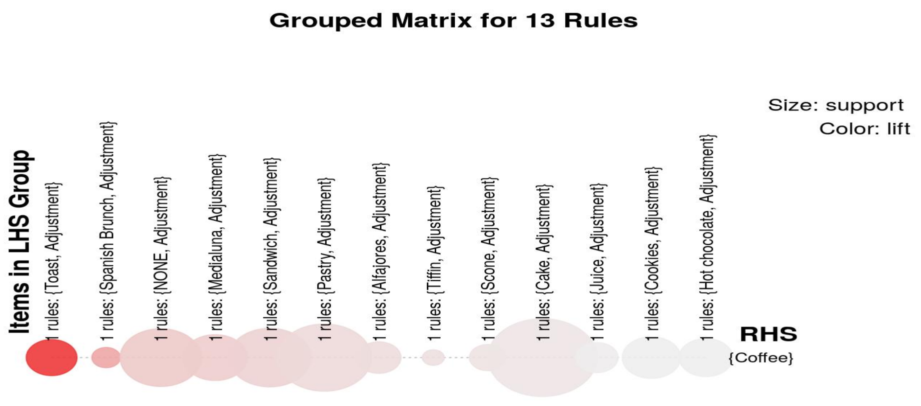
```
# Graph (circular layout)

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



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 ballons, respectively. In this case it’s not a very useful visualization, since we only have coffe on the right-hand-side of the rules.

Code

```
# Grouped matrix plot

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



Conclusion:

- In the quest to build a house price prediction model, we have embarked on a critical journey that begins with loading and preprocessing the dataset.We have traversed through essential steps, starting with importing the necessary libraries to facilitate data manipulation and analysis.

- Understanding the data's structure, characteristics, and any potential issues through exploratory data analysis (EDA) is essential for informed decision-making.

- Data preprocessing emerged as a pivotal aspect of this process. Itinvolves cleaning, transforming, and refining the dataset to ensurethat it aligns with the requirements of machine learningalgorithms.

- With these foundational steps completed, our dataset is now primed for the