Introduction:

Imagine you and your friends go to the store in a hurry. You have a party to attend and they've run out of food!

The first thing you see in the store is cabbage. Unfortunately, all the other related food items are far away! Now if we're on a time constraint or simply lazy, this makes us unable to buy the things we need and creates losses for the store.

Now consider a situation where cabbage, lettuce and juices are all placed in one basket? The chances of you picking up items that you may not even need, increase.

How can we find what item to place beside which one to maximise our sales and for customers to have ease in exploring our repository? Let's find out.

Table of Contents:

1. The Approach (Apriori Algorithm)

When you go into a store, would you not want the aisles to be ordered in such a way that it reduces your effort to buy things?

For example, when you go to a store and want to purchase toothbrushes, you would automatically assume and wish for the toothpastes to be there as well. It would make no point walking all across the hall just to buy it. We can also put mouthwash and other dental products in one aisle to boost sales and even to make shopping easier for the customer. This is done by a way in which we find associations between items. So, if we see that someone who buys toothbrushes also buys mouthwash and paste, we can place them all in one aisle.
For example, if we see {Milk} as a size 1-Itemset and {Coffee} as another size 1-Itemset, we will use these to find size 2-itemsets in the dataset such as {Milk,Coffee} and then later see our right hand side. If someone buys Coffee with Milk, we will represent it as {Coffee} => {Milk}.

When we use these to explore more size k-itemsets, we might find that {Coffee,Milk} => Tea.
That means, the people who buy Coffee and Milk have a possibility of buying Tea as well.

Apriori envisions an iterative approach where it uses size k-Itemsets to search for size (k+1)-Itemsets. The first 1-Itemsets are found by gathering the count of each item in the set. Then the size 1-Itemsets are used to find size 2-Itemsets and so on until no more size k-Itemsets can be explored. One exploration takes one scan of the complete dataset.

An itemset is a mathematical set of products in the basket.

1.1 Handling and Readying The Dataset

The first part of any analysis is to bring in the dataset. We will be using an inbuilt dataset “Groceries” from the ‘arules’ package to simplify our analysis.

The ‘pacman’ package is an assistor to help load and install the packages.

The p_load() function from “pacman” takes names of packages as arguments.

If your system has those packages, it will load them and if not, it will install and load them.

Example: pacman::p_load(PACKAGE_NAME)

```
pacman::p_load(arules, arulesViz)
data(“Groceries”)
```

1.2 Structural Overview and Prerequisites

```
str(Groceries)
```

The structure of our transaction type dataset shows us that it is internally divided into three slots: Data, itemInfo and itemsetInfo.

The slot Data contains the dimensions, dimension names and other numerical values of number of products sold.

The slot itemInfo contains a Data Frame that has three vectors which

categorizes the food items in the first vector “Labels”.
The second & third vectors divide the food broadly into levels like “baby food”,”bags” etc.

```
Formal class 'transactions' [package "arules"] with 3 slots
  ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
  .. .. ..@ i       : int [1:43367] 13 60 69 78 14 29 98 24 15 29 ...
  .. .. ..@ p       : int [1:9836] 0 4 7 8 12 16 21 22 27 28 ...
  .. .. ..@ Dim     : int [1:2] 169 9835
  .. .. ..@ Dimnames:List of 2
  .. .. .. ..$ : NULL
  .. .. .. ..$ : NULL
  .. .. ..@ factors : list()
  ..@ itemInfo   :'data.frame': 169 obs. of  3 variables:
  .. ..$ labels: chr [1:169] "frankfurter" "sausage" "liver loaf" "ham" ...
  .. ..$ level2: Factor w/ 55 levels "baby food","bags",..: 44 44 44 44 44 44 44 42 42 41 ...
  .. ..$ level1: Factor w/ 10 levels "canned food",..: 6 6 6 6 6 6 6 6 6 6 ...
  ..@ itemsetInfo:'data.frame': 0 obs. of  0 variables
```

summary(Groceries)

```
transactions as itemMatrix in sparse format with
 9835 rows (elements/itemsets/transactions) and
 169 columns (items) and a density of 0.02609146

most frequent items:
      whole milk other vegetables       rolls/buns             soda           yogurt
            2513             1903             1809             1715             1372
         (Other)
           34055

element (itemset/transaction) length distribution:
sizes
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20
2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55   46   29   14   14    9
  21   22   23   24   26   27   28   29   32
  11    4    6    1    1    1    1    3    1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   3.000   4.409   6.000  32.000

includes extended item information - examples:
       labels  level2           level1
1 frankfurter sausage meat and sausage
2     sausage sausage meat and sausage
3  liver loaf sausage meat and sausage
```

The summary statistics show us the top 5 items sold in our

transaction set as “Whole Milk”,”Other Vegetables”,”Rolls/Buns”,”Soda” and “Yogurt”. (Further explained in Section 3)

Before we begin applying the “Apriori” algorithm on our dataset, we need to make sure that it is of the type “Transactions”.

To parse, make sure your dataset has similar slots and then use the as() function in R.

2. Key Terms and Usage

To explain a few terms in the output:

Support: Support is the probability of an event to occur. If we have an event to buy product A, Support(A) is the number of transactions which includes A divided by total number of transactions.

Confidence: The confidence of an event is the conditional probability of the occurrence; the chances of A happening given B has already happened.

Lift: This is the ratio of confidence to expected confidence. The lift value tells us how much better a rule is at predicting something than randomly guessing. The higher the lift, the stronger the association.

```
rules <- apriori(Groceries,
                 parameter = list(supp = 0.001, conf = 0.80))

inspect(rules[1:10])
```

We will set minimum support parameter (minSup) to .001

We can set minimum confidence (minConf) to anywhere between

0.75 and 0.85 for varied results.

```
     lhs                                                rhs                support     confidence
[1]  {liquor,red/blush wine}                         => {bottled beer}     0.001931876 0.9047619
[2]  {curd,cereals}                                  => {whole milk}       0.001016777 0.9090909
[3]  {yogurt,cereals}                                => {whole milk}       0.001728521 0.8095238
[4]  {butter,jam}                                    => {whole milk}       0.001016777 0.8333333
[5]  {soups,bottled beer}                            => {whole milk}       0.001118454 0.9166667
[6]  {napkins,house keeping products}                => {whole milk}       0.001321810 0.8125000
[7]  {whipped/sour cream,house keeping products}     => {whole milk}       0.001220132 0.9230769
[8]  {pastry,sweet spreads}                          => {whole milk}       0.001016777 0.9090909
[9]  {turkey,curd}                                   => {other vegetables} 0.001220132 0.8000000
[10] {rice,sugar}                                    => {whole milk}       0.001220132 1.0000000
     lift
[1]  11.235269
[2]   3.557863
[3]   3.168192
[4]   3.261374
[5]   3.587512
[6]   3.179840
[7]   3.612599
[8]   3.557863
[9]   4.134524
[10]  3.913649
```

As we can see, these are the top 10 rules derived from our Groceries dataset by running the above code. Let's plot all our rules in certain visualisations first to see what goes with what item in our shop.

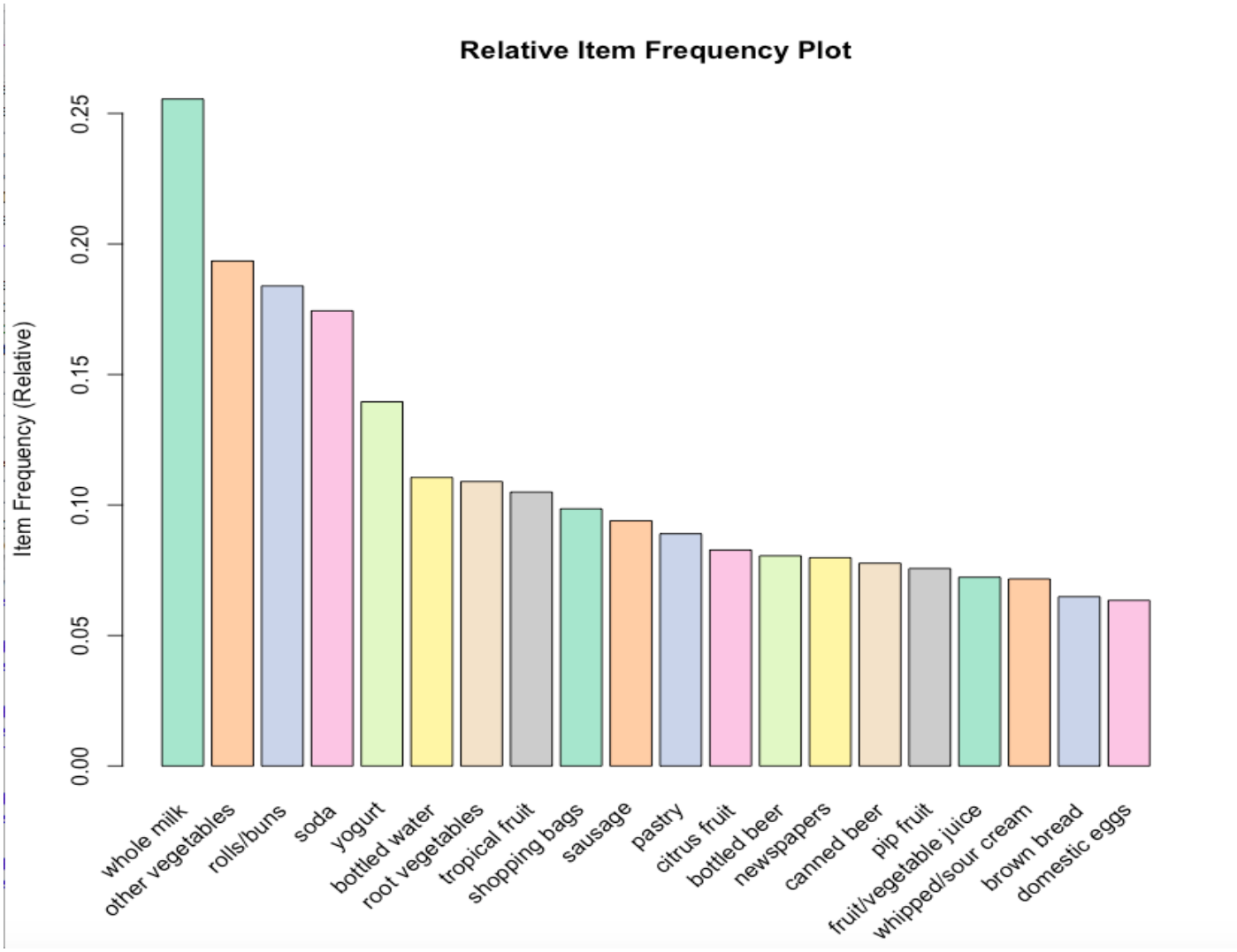
3. Interpretations and Analysis

Let us first identify which products were sold how frequently in our dataset.

3.1 The Item Frequency Histogram

These histograms depict how many times an item has occurred in our dataset as compared to the others.

The relative frequency plot accounts for the fact that "Whole Milk" and "Other Vegetables" constitute around half of the transaction dataset; half the sales of the store are of these items.



arules::itemFrequencyPlot(Groceries,topN=20,col=brewer.pal(8,'Pastel2'),main='Relative Item Frequency Plot',type="relative",ylab="Item Frequency (Relative)")

This would mean that a lot of people are buying milk and vegetables!

What other objects can we place around the more frequently purchased objects to enhance those sales too?

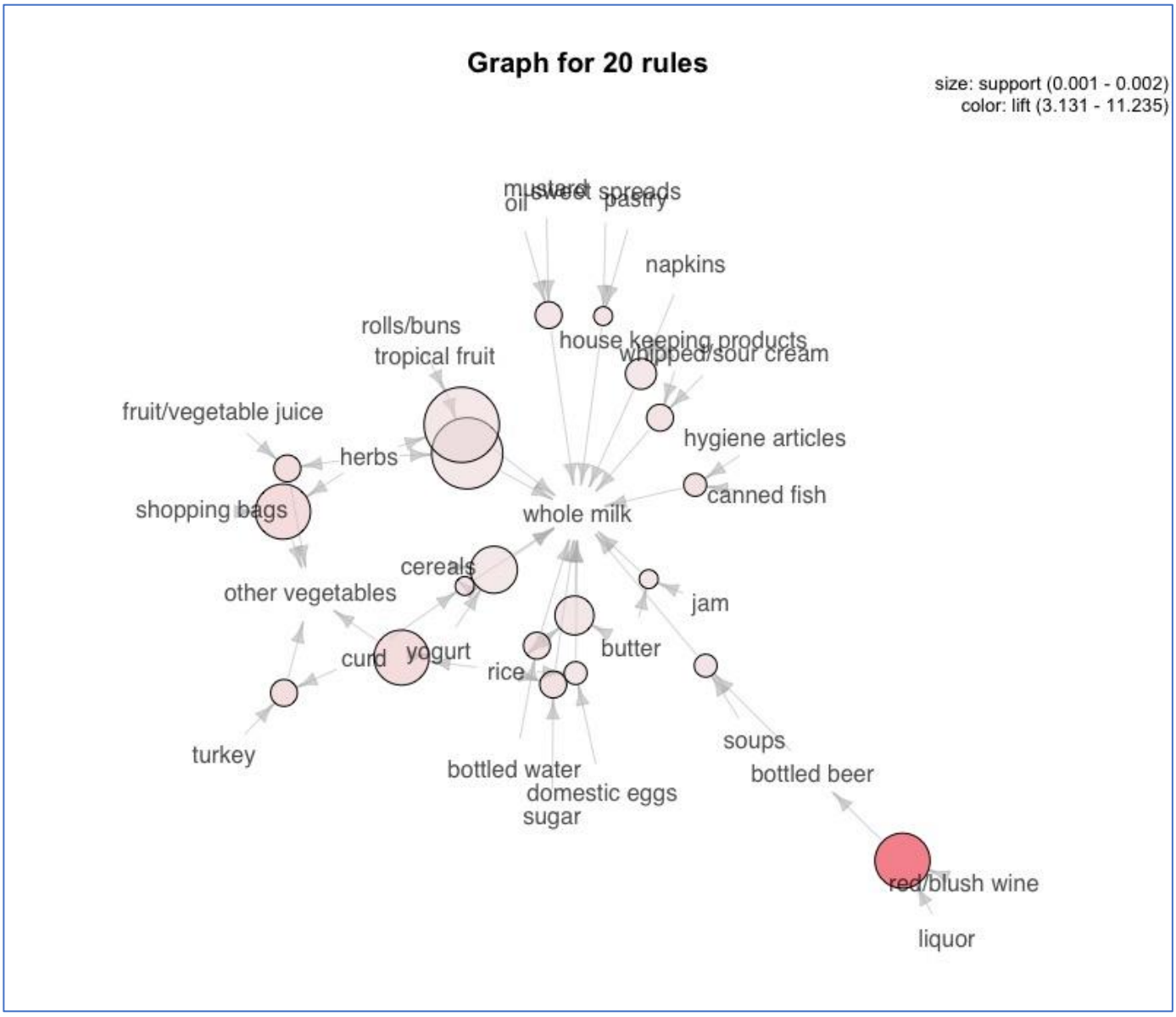
For example, to boost sales of eggs I can place it beside my milk and vegetables.

3.2 Graphical Representation

Moving forward in the visualisation, we can use a graph to highlight the support and lifts of various items in our repository but mostly to see which product is associated with which one in the sales environment.

```
plot(rules[1:20],
    method = "graph",
    control = list(type = "items"))
```

This representation gives us a graph model of items in our dataset.



The above graph shows us that most of our transactions were consolidated around “Whole Milk”.

We also see that all liquor and wine are very strongly associated so we must place these together.

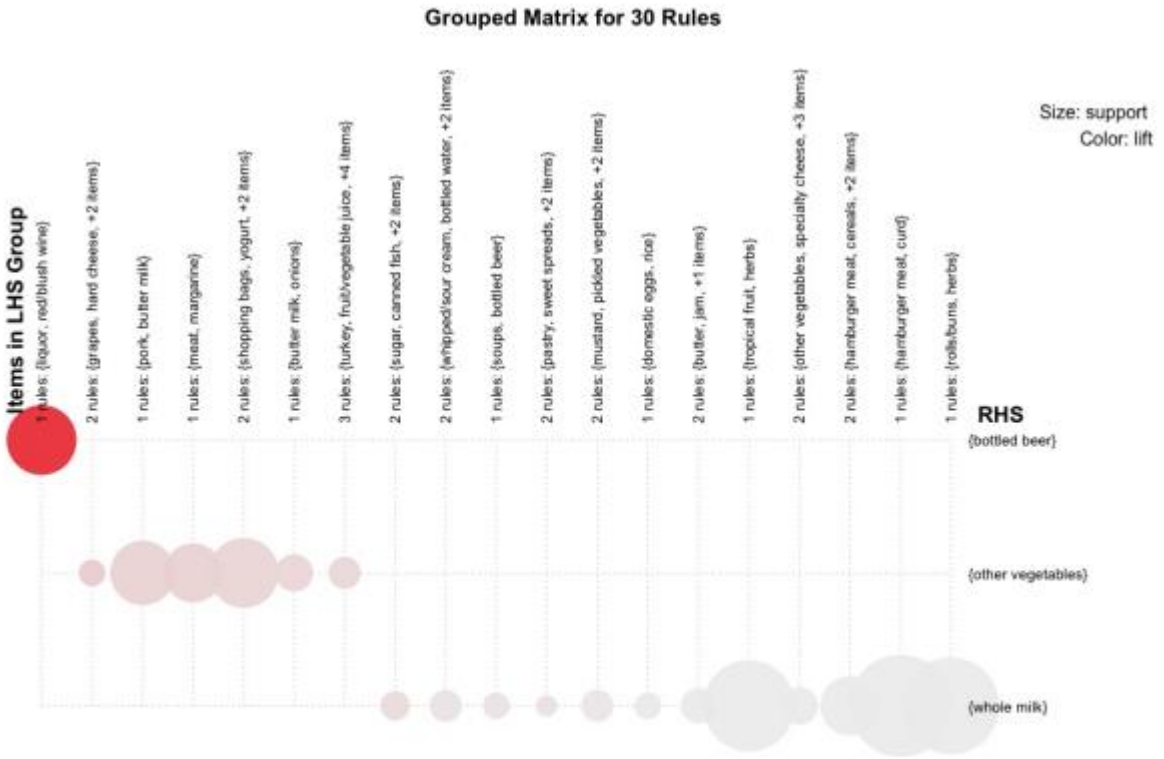
Another association we see from this graph is that the people who buy tropical fruits and herbs also buy rolls and buns. We should place these in an aisle together.

3.3 Grouped Matrix Representation

The grouping will help us see the previous plot with a better understanding.

The third plot gives us a grouped matrix of the top 30 rules to amplify our analysis in the best form.

```
plot(rules[1:30], method = "grouped")
```



This plot reveals the top rules of our dataset very clearly. For example, we can see that when we have ***liquor and red/blush wine*** on our LHS; implying that if we pick up those two items, there is an almost certain possibility of us picking up ***bottled beer*** as well.

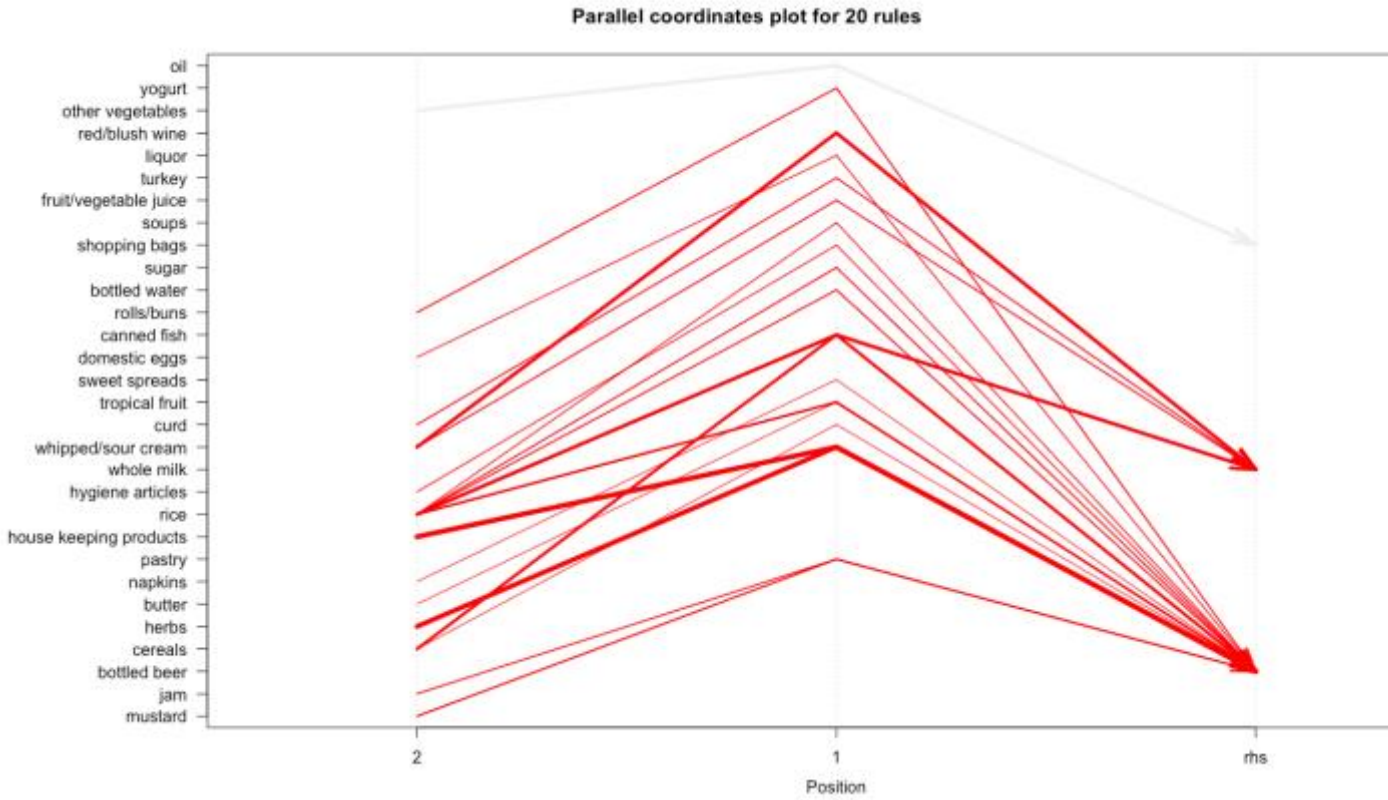
Alternatively, another rule with a similar support is one wherein if a customer picks up butter milk and pork, he is highly likely to pick up other vegetables as well.

3.4 Parallel Coordinate Representation

The next plot offers us a parallel coordinate system of visualisation. It would help us clearly see that which products along with which ones, result in what kinds of sales.

The topmost rule shows us that when I have rolls/buns and other vegetables in my shopping cart, I am highly likely to buy jam to go

along with my breads as well.



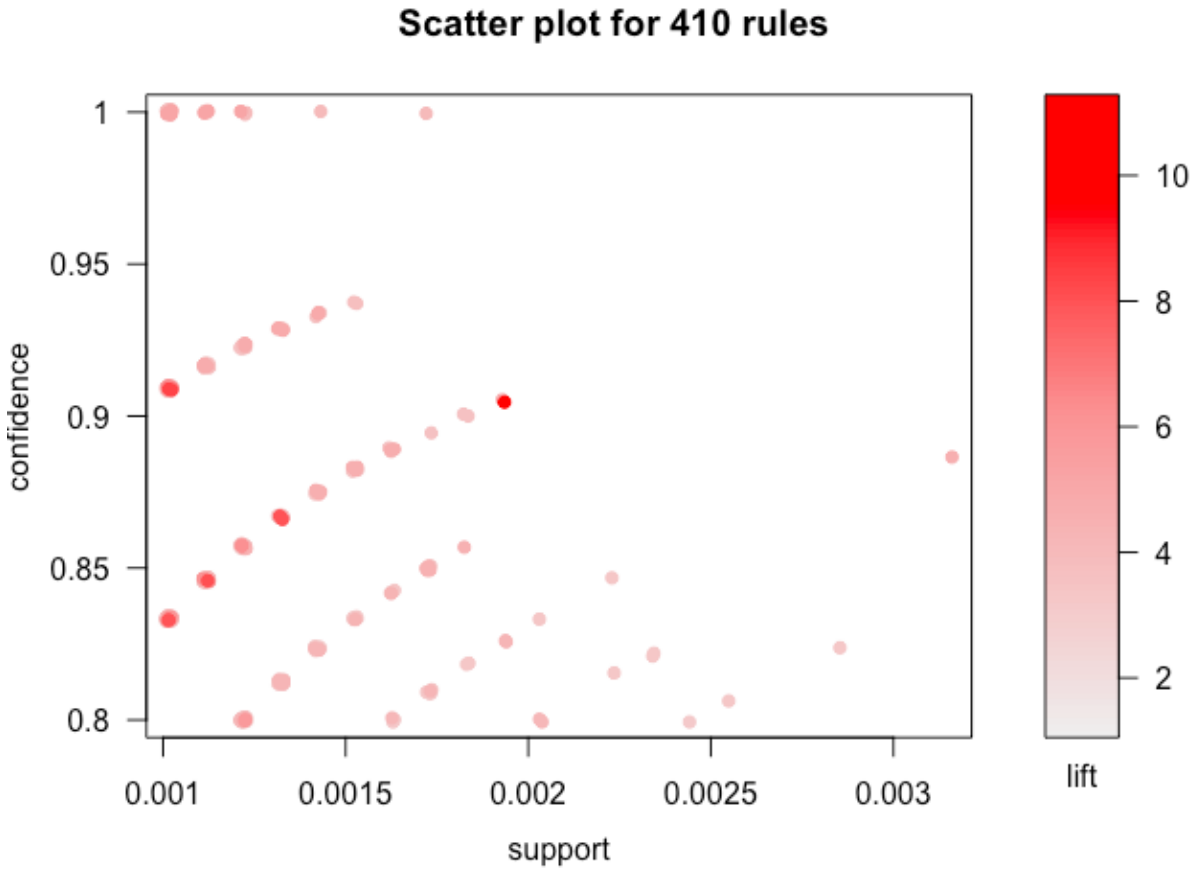

```
plot(rules[1:20],
   method = "paracoord",
   control = list(reorder = TRUE))
```

If we want a matrix representation, an alternate code option would be:

```
plot(rules[1:20],
   method = "matrix",
   control = list(reorder = TRUE))
```

3.5 Scatterplot and Interactive Scatterplot

These plots show us each and every rule visualised into a form of a scatterplot. The confidence levels are plotted on the Y axis and Support levels on the X axis for each rule. We can hover over them in our interactive plot to see the rule.

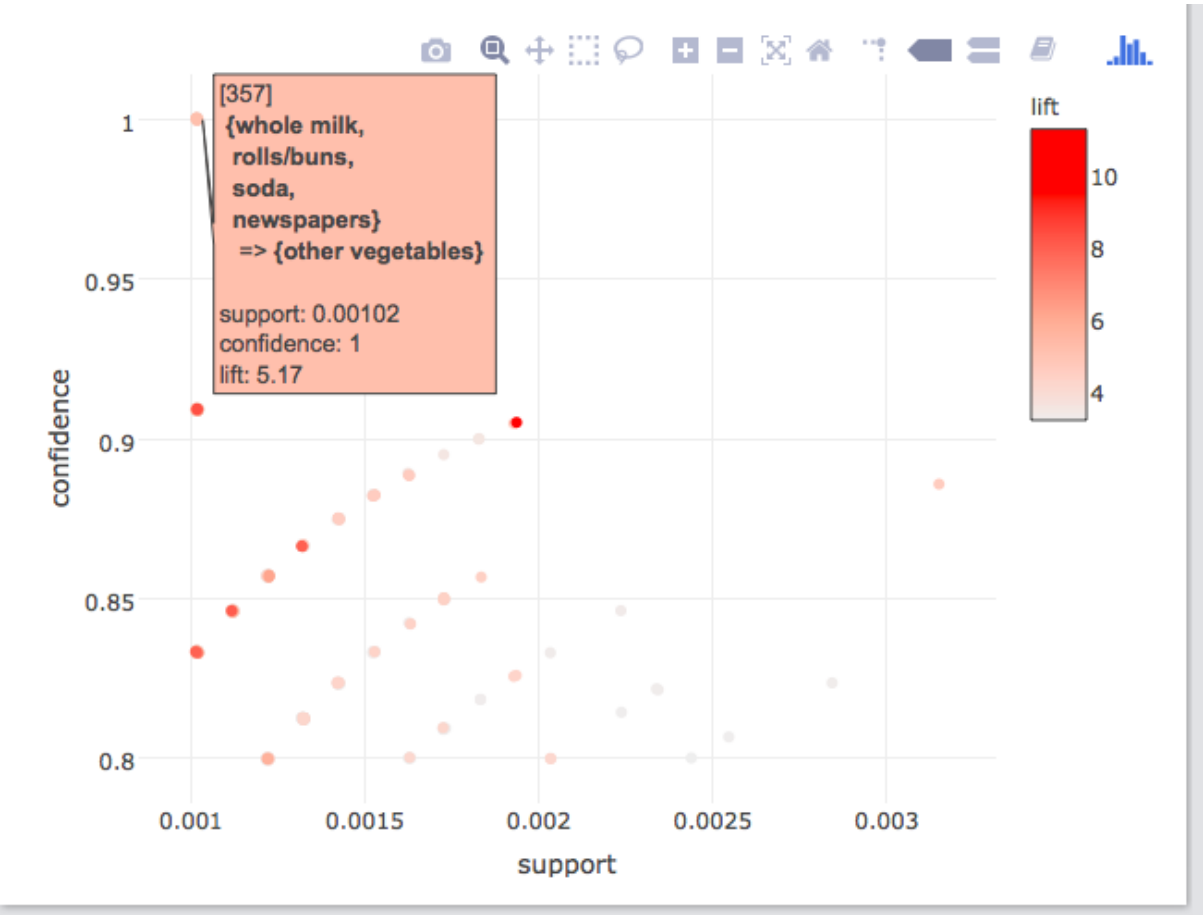


We can use two forms of plotting in this case.

Plot 1: plot(rules) #Generic R plot

Plot 2: arulesViz::plotly_arules(rules)

Plot 2 uses the arulesViz package and plotly to generate an interactive plot.



As the interactive plot suggests, one rule that has a confidence of 1 is the one above. It has an exceptionally high lift as well, at 5.17.

4. End Notes and Summary

By visualising these rules and plots, we can come up with a more detailed explanation of how to make business decisions in retail environments.

Now, we would place “Whole Milk” and “Vegetables” beside each other; “Wine” and “Bottled Beer” alongside too.

I can make some specific aisles now in my store to help customers pick products easily from one place and also boost the store sales simultaneously.

Aisles Proposed:

1. **Groceries Aisle** – Milk, Eggs and Vegetables
2. **Liquor Aisle** – Liquor, Red/Blush Wine, Bottled Beer, Soda
3. **Eateries Aisle** – Herbs, Tropical Fruits, Rolls/Buns, Fruit Juices, Jams
4. **Breakfast Aisle** - Cereals, Yogurt, Rice, Curd

This analysis would help us improve our store sales and make calculated business decisions for people both in a hurry and the ones leisurely shopping.