Market Basket Insights

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"**.

First it's important to define the Apriori algorithm, including some statistical concepts (support, confidence, lift and conviction) to select interesting rules. Then we are going to use a data set containing more than 6.000 transactions from a bakery to apply the algorithm and find combinations of products that are bought together. Let's start!

Flow Chart/Process Flow Diagram

A Flow Chart (also known as a Process Flow Diagram or Process Map) is a diagram of the steps in a process and their sequence. Two types of flow charts are utilized in quality improvement. A high-level flowchart, outlining 6-10 major steps, gives a high-level view of a process. These flowcharts display the

major blocks of activity, or the major system components, in a process. These charts are especially useful in the early phases of a project and help to set priorities for improvement work. A detailed flowchart is a close-up view of the process, typically showing dozens of steps. These flowcharts make it easy to identify complexity, excessive steps, etc. in a process and should be used when you want to standardize or make changes in the process. There are many types of flow charts/process maps including swim lane, value stream, cross functional and workflow. When to Use a Flow Chart When you need to define or analyze an existing process. When you need to standardize or redesign a process. When you need to find areas for improvement in a process such as unnecessary steps, gaps, barriers, etc. How Flow Charts Are Constructed 1. Identify the goal for creating the flowchart and the level of detail required-high or detailed. 2. Assemble the people who know the process best and outline the process steps. 1 3. Define the first and last steps in the process. 4. Begin documenting the process steps in sequence. Some steps may be parallel-they happen at the same time. Describe the process as it really exists, not the ideal. Most flow charts are made up of three main types of symbol: Elongated circles, which signify the start or end of a process. Rectangles or squares which show instructions or actions. Diamonds which show decisions that must be made 5. Work through the entire process, showing actions and decisions appropriately in the order they occur. Link these together using arrows to show the flow of the process. (Tip: Self-adhesive notes are a flexible way to document steps, using one note for each step. This allows you to easily change the order or add new steps.) 6. At decision symbols, choose the most natural branch and continue to the end. 7. Use notes for unfamiliar steps and continue to the end. 8. When you reach the last step, go back to fill in any branches. 9. Follow up on unfamiliar steps and update chart. 10. Validate your flow chart. Work from step to step asking yourself and others if you have correctly represented the sequence of actions and decisions involved in the process. 11. Identify areas for improvement and redesign the process.

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:

3 **Loading Data**

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

```
## transactions in sparse format with
##  6614 transactions (rows) and
##  104 items (columns)
```

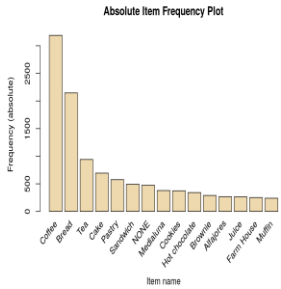
4 **Data Dictionary**

The data set contains 15.010 observations and the following columns,

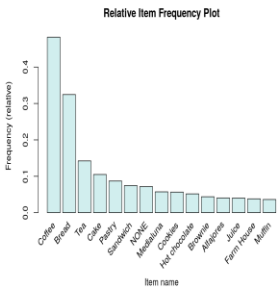
- `Date`. Categorical variable that tells us the date of the transactions (YYYY-MM-DD format). The column includes dates from 30/10/2016 to 09/04/2017.
- `Time`. Categorical variable that tells us the time of the transactions (HH:MM:SS format).
- `Transaction`. Quantitative variable that allows us to 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.

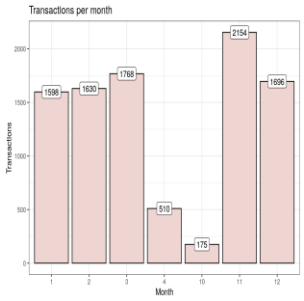


The `itemFrequencyPlot()` allows us to show the absolute or relative values. If absolute it will plot numeric frequencies of each item independently. If relative it will plot how many times these items have appeared as compared to others, as it's shown in the following plot.

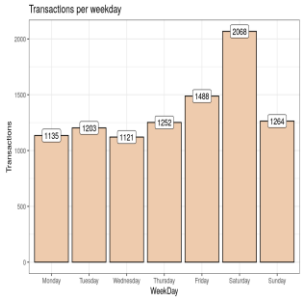


Coffee is the best-selling product by far, followed by bread and tea. Let's display some other visualizations describing the time distribution using the ggplot() function.

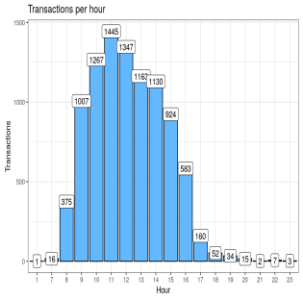
- Transactions per month
- Transactions per weekday
- Transactions per hour



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.



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



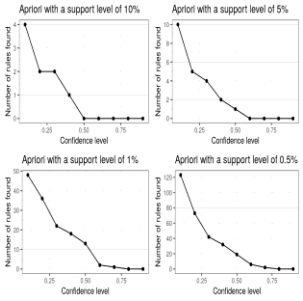
There's not much to discuss with this visualization. The results are logical and expected.

6 **Apriori algorithm**

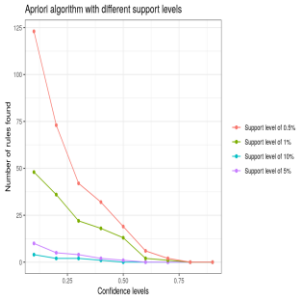
6.1 Choice of support and confidence

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.

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



We can join the four lines to improve the visualization.



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.

The generated association rules are the following,

```
##      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.

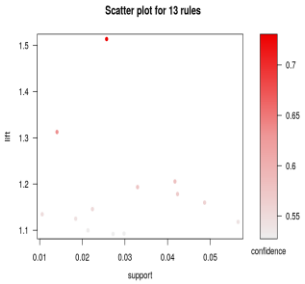
How do we interpret these rules?

- 52% of the customers who bought a hot chocolate algo 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.

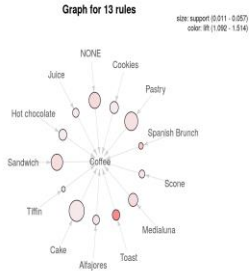
And so on. It seems that in this bakery there are many coffee lovers!

6.3 Visualize association rules

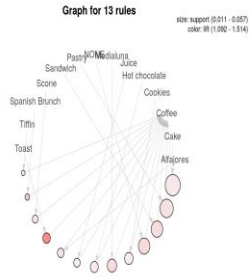
We are going to use the `arulesViz` package to create the visualizations. Let's begin with a simple scatter plot with different measures of interestingness on the axes (lift and support) and a third measure (confidence) represented by the color of the points.



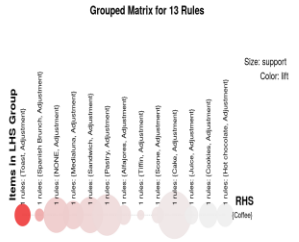
The following visualization represents the rules as a graph with items as labeled vertices, and rules represented as vertices connected to items using arrows.



We can also change the graph layout.



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.

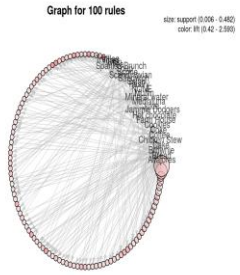


There's an awesome function called `ruleExplorer()` that explores association rules using interactive manipulations and visualization using shiny. Unfortunately, R Markdown still doesn't support shiny app objects.

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%.

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.



9 **Citations for used packages**

Hadley Wickham (2017). tidyverse: Easily Install and Load the 'Tidyverse'. R package version 1.2.1

Michael Hahsler, Christian Buchta, Bettina Gruen and Kurt Hornik (2018). arules: Mining Association Rules and Frequent Itemsets. R package version 1.6-1.

Michael Hahsler, Bettina Gruen and Kurt Hornik (2005), arules - A Computational Environment for Mining Association Rules and Frequent Item Sets. Journal of Statistical Software 14/15. URL:

Michael Hahsler, Sudheer Chelluboina, Kurt Hornik, and Christian Buchta (2011), The arules R-package ecosystem: Analyzing interesting patterns from large transaction datasets. Journal of Machine Learning Research, 12:1977–1981.

Michael Hahsler (2018). arulesViz: Visualizing Association Rules and Frequent Itemsets. R package version 1.3-1.

Yihui Xie (2018). knitr: A General-Purpose Package for Dynamic Report Generation in R. R package version 1.20.

Yihui Xie (2014) knitr: A Comprehensive Tool for Reproducible Research in R. In Victoria Stodden, Friedrich Leisch and Roger D. Peng, editors, Implementing Reproducible Computational Research. Chapman and Hall/CRC. ISBN 978-1466561595

Baptiste Auguie (2017). gridExtra: Miscellaneous Functions for “Grid” Graphics.