

Tidy data, Association rules & K-nn classification model

Data & Web Mining



A picture containing text, map

Description automatically generated[[1]](#footnote-1)

teresa ventaja

Higher Diploma in Science in Data Analytics

INDEX

[**PART 1 - apply the principles of Tidy Data & run the Association Rules Algorithm** 2](#_Toc46578382)

[1. Loading the data 2](#_Toc46578383)

[2. Splitting the first column 2](#_Toc46578384)

[3. Association Rules & Report of Results 3](#_Toc46578385)

[Visuals 3](#_Toc46578386)

[*Apriori* 5](#_Toc46578387)

[**PART 2 - Classification Model** 6](#_Toc46578388)

[1. Pre-processing 7](#_Toc46578389)

[2. Normalizing the data 8](#_Toc46578390)

[3. k-NN Algorithm & Report of Results 8](#_Toc46578391)

[**Bibliography** 10](#_Toc46578392)

# **PART 1 - apply the principles of Tidy Data & run the Association Rules Algorithm**

In this section, based on the definitions and techniques outlined in the article of Hadley Wickham (Wickham, 2014), I am going to develop a series of pre-processing activities to get the data ready to apply Association Rules algorithm. I am going to explain every step thoroughly, clearly identifying the techniques used and their sources.

Once the data is clear enough to run the algorithm in RStudio, I will explain how I created the algorithm, and the meaning of the outcome of the code. I will also explain the meaning of the statistical parameters given by the result of running the code. I will support my explanations with relevant graphical representations, and I will copy the result of the code when it is relevant enough to deduce some conclusions.

The R script, whose file is named “*Part1.R*”, is divided into 3 sections clearly differentiated: “*STEP 1 – Loading the data*”, “*STEP 2 - Splitting the first column*” and “*STEP 3 - Association rules and results*”. It is fully commented and labelled, with the purpose of facilitating readiness.

## Loading the data

For a start, I installed and loaded packages "*readxl", “knitr", "lubridate", "dplyr", "ggplot2" “tidyverse*” and "*plyr".* Those 5 last packages where created by Hadley Wickham and are used for tidy data.

I then proceeded to load the data into R. I used *read\_excel* function from “*readxl*” instead of read.csv because *read.csv* requires a fixed length of rows per column. In this case, *groceries.csv* file contained one transaction per row, but each transaction had a different length. That was resulting in a faulty reading of the data, transferring columns number 33 and 34 into a new row. Using the data in that format would have impacted negatively the result of Association Rules algorithm, returning a wrong outcome of the code.

## Splitting the first column

Next, I executed the most essential conversion needed in this data set, i.e., splitting the first column into two columns. The first column resulting on the date, and the other one containing the item that I am going to use as part of the transaction list. I isolated the column that I needed to work on and, based on Section 3.2 of our reference article, called “*Multiple variables stored in one column*”, split the column using the function *separate().* As a separator, I used the regular expression “*(?<=000)|(?<=001)|(?<=002)”.*

Then, I replaced the first column in my original data set (containing the date and the item) with the "*item*" column just created (containing only the item). As a result, my original data frame “groceries” was clean and ready to apply *apriori* algorithm. I saved the result into a .csv file on my working directory, naming it as “*groceries2.csv*”.

## Association Rules & Report of Results

I started this section by reading my saved data as transactions (“*read.transactions()”* function), removing duplicates, because the *apriori* algorithm only recognizes if an item is present. Therefore, how many times the item is present is not relevant in this case. I also made sure the column headers were not present, because that would have resulted in a faulty outcome of the code.

Using the *summary()* and *inspect()* functions, I got those results:

vegetables poultry

1089 613

waffles dishwashing liquid/detergent

587 585

ice cream (Other)

584 18394

The most frequent items are vegetables, poultry, waffles, dishwashing liquid/detergent and ice cream. The most frequent length distributions are transactions containing 20 items, 17 items, 18 items, 19 items and 14 items, followed by those containing 15, 16, 11, 21 and 9:

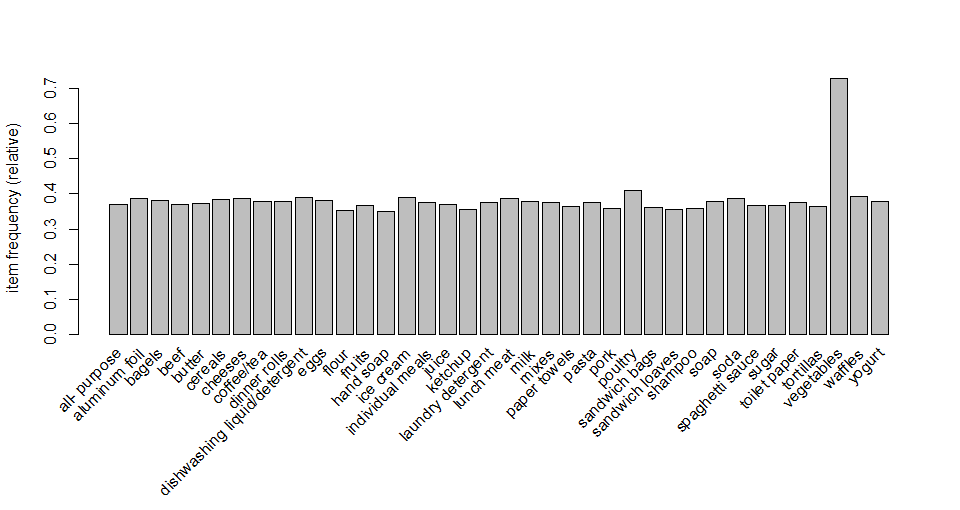
A screenshot of a cell phone

Description automatically generated

The minimum frequency list of items is 4, and the maximum is 27, whereas the median value of transaction length is 15, the first quartile being 10, and the third quartile 19.

### Visuals

1. **Plot for items meeting criteria "support equal to or greater than 0.1"**

[[2]](#footnote-2)

1. **Plot of Top 20 items (highest frequency)**

A screenshot of a cell phone

Description automatically generated[[3]](#footnote-3)

1. **Matrix containing items contained in each transaction**

y axis 🡪 each transaction

x axis 🡪 what items are included in each transaction

A close up of a logo

Description automatically generated[[4]](#footnote-4)

1. **Matrix (items contained in each transaction) - Sample of 50**

y axis 🡪 each transaction

x axis 🡪 what items are included in each transaction

A picture containing fruit

Description automatically generated[[5]](#footnote-5)

### *Apriori*

I started applying the algorithm with default parameters, getting a result of 443 rules created. Using *summary()* function, I could see that there are 5 rules containing 2 items, and 438 rules containing 3 items.

Then I experimented further with the algorithm, defining different parameters other than defaults. I tested different combinations by changing support level and confidence, and found a useful result with *support = 0.14*, *confidence = 0.8* and *minlen = 2*. In this case I got 5 rules containing 2 items, and 36 rules containing 3 items. To control the result, I used *inspect()* and *sort()* functions, as well as *subset()* by item, selecting “*vegetables*” because we could see that it is the most frequent item in absolute terms.

I can conclude that in this case the result obtained using the default parameters is probably more useful than the one we manipulated, because it creates a greater number of rules without being too many. However, that would depend on specific business needs. To finalise, as a visual reference, the code outcome looks as follows:

A screenshot of text

Description automatically generated

# **PART 2 - Classification Model**

In this section, I selected the “*Occupancy Detection Data Set*”, from the University of California Irvine (Candanedo and Feldheim, 2016). The data set contains 20560 observations and 7 attributes. The purpose of building a classification model is to detect room occupancy of an office room by using light, temperature, CO2 and humidity data collected from sensors. To build this model, I followed the tutorials of *Jalayer Academy* (2015)[[6]](#footnote-6):

## Pre-processing

The original data set was divided on 3 different files. One file containing the test data, and 2 files with the training data. The first thing I did is I combining the three files to do my own selection of test and training data. The reason why I decided to do so is because it was not a random selection. The test data was selected by date in a specific order, not randomly picked from the entire set of data. I believe the result is more representative if proper randomization is done.

Now we are going to look at the description of my resulting “room\_oc.csv” file data[[7]](#footnote-7):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Column description** | **Data type** | **Range** | **% Null ratios** |
| **Date** | Date & time in year-month-day hour:minute:second format | date | 02/02/2015 14:19:00 - 2/18/2015 9:19 | 0 |
| **Temperature** | Temperature in degree Celcius | numeric - ordinal - continuous | 19 - 24.40833333 | 0 |
| **Humidity** | Relative humidity in percentage | numeric - ordinal - continuous | 16.745 - 39.5 | 0 |
| **Light** | Illuminance measurement in unit Lux | numeric - ordinal - continuous | 0 - 1697.25 | 0 |
| **CO2** | CO2 in parts per million (ppm) | numeric - ordinal - continuous | 412.75 - 2076.5 | 0 |
| **HumidityRatio** | Humadity ratio: Derived quantity from temperature and relative humidity, in kgwater-vapor/kg-air | numeric - ordinal - continuous | 0.002674127 - 0.006476013 | 0 |
| **Occupancy** | Occupied or not: 1 for occupied and 0 for not occupied | numeric - ordinal - discrete | 0, 1 | 0 |

[[8]](#footnote-8)

I loaded my unified file using *read.csv()* function. Then I looked at the structure of the data with *str()* and *head()* functions, getting something similar to the table above and a sample of the top 6 rows. Since the aim of the classification is the detection of room occupancy, it is important to check how many observations we have for occupied and how many for not occupied. I checked that using *table()* function for “*Occupancy*” column, getting a result of 15810 for not occupied and 4750 for occupied.

## Normalizing the data

Now I am going to change the order of the data. Now the data is ordered by date, and I want it to be mixed, not following a chronological order but a random order instead. To do that, I first use the function *set.seed()* to allow reproducibility. Then I generate as many random numbers as the number of rows is, using *runif()* function. Using the order of those random numbers, I relocate the original rows to a new random position[[9]](#footnote-9). Once the order is randomized, I newly look at the structure and the top 6 rows. As expected, the structure has not changed, but the top 6 rows are now different than when we run *head()* before.

Before I start normalization, I use *summary()* function to check descriptive statistics for the attributes that I am going to use for the classification (Temperature, Humidity, Light and CO2). Clearly, the data is not normally distributed, and that would prevent the algorithm from getting an accurate result.

To fix that, I am using a function that it is going to be passed through parameter "*x*" and convert it into normally distributed data:

normalize <- function(x) {

return( (x - min(x)) / (max(x) - min(x)) )

}

To test if the "normalize" function works, I pass it to numbers 1:5, and I can see the result is accurate. Therefore, I proceed and use lapply() function to pass attributes Temperature, Humidity, Light and CO2 through the function "*normalize*" in a loop. What this does is converting every observation into normalized data which I can operate with on my kNN algorithm. Newly checking the structure and the summary of the data, I can see that the 4 variables that I am going to apply k-NN to are isolated, and the data is normally distributed. The data is ready for the next step.

## k-NN Algorithm & Report of Results

My data set is large enough to select a test sample of approximately 12% of the observations, so I used 2462 of the 20560 observations for the test data, and the rest of them for the training data. To prepare the target data, I noted that “*Occupancy”* column is not in not in my data frame containing the normalised data, but instead it is in the data frame that I created when I altered the order of the rows. It is called "room2", and I am using the 7th column to prepare the test and training target data.

I loaded the library for classification rules algorithm (“*class*”) and then I chose a “*k*” value. Since the rule of thumb is generally using and odd number close to the square root of the number of observations, I chose k=143 in this case. The data is now prepared to build the model with *knn()* function:

m1 <- knn(train=room\_n\_train, test=room\_n\_test,

cl=room\_train\_target, k=143)

m1

Finally, I created a table to check how well the model predicted the different classes:

m1

room\_test\_target 0 1

0 1834 27

1 9 592

1834 office rooms that were not occupied have been predicted correctly and only 9 incorrectly. The accuracy rate is, therefore, greater than 99.5% for rooms that were not occupied. When it comes to office rooms that were occupied, 592 were predicted correctly and 27 were not. The accuracy rate in this case is significantly lower, predicting correctly occupied office rooms 95.6% of the times.

# **Bibliography**

* Annalyn Ng (2016): “Association Rules and the Apriori Algorithm: A Tutorial”, *KDnuggets* [online]. Available at: <https://www.kdnuggets.com/2016/04/association-rules-apriori-algorithm-tutorial.html> (Accessed 24/07/2020)
* Candanedo, L.M. & Feldheim, V. (2016): “Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models” *Energy and Buildings*. Volume 112, 15 January 2016, Pages 28-39 [online]. Available at: [https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+#](https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+) (Accessed 24/07/2020)
* Jalayer Academy (2015): “R - kNN - k nearest neighbor (part 1)”, *YouTube* [online]. Available at: <https://www.youtube.com/watch?v=GtgJEVxl7DY> (Accessed 24/07/2020)
* Jalayer Academy (2015): “R - kNN - k nearest neighbor (part 2)” *YouTube* [online]. Available at: <https://www.youtube.com/watch?v=DkLNb0CXw84> (Accessed 24/07/2020)
* Merrett, R. (2020): “Occupancy Detection”, *Data Science Dojo* [online]. Available at: <https://code.datasciencedojo.com/datasciencedojo/datasets/tree/master/Occupancy%20Detection> (Accessed 24/07/2020)
* Wickham, H. (2014): “Tidy Data”, *Journal of Statistical Software*, Volume 59 (Issue 10), 1-21 [online]. Available at: <https://www.jstatsoft.org/article/view/v059i10> (Accessed 24/07/2020)

1. Image source: <https://www.kdnuggets.com/2016/04/association-rules-apriori-algorithm-tutorial.html> [↑](#footnote-ref-1)
2. Plot 1: Items with support level equal or greater than 0.1 [↑](#footnote-ref-2)
3. Plot 2: Top 20 frequency items [↑](#footnote-ref-3)
4. Plot 3: Matrix – Items bought per transaction (Top 5) [↑](#footnote-ref-4)
5. Plot 4: Matrix – Items bought per transaction (Top 50) [↑](#footnote-ref-5)
6. Source: <https://www.youtube.com/watch?v=GtgJEVxl7DY> & <https://www.youtube.com/watch?v=DkLNb0CXw84&t=692s> [↑](#footnote-ref-6)
7. <https://code.datasciencedojo.com/datasciencedojo/datasets/tree/master/Occupancy%20Detection> [↑](#footnote-ref-7)
8. Table 1: Description of “*Occupancy Detection Data Set*” data [↑](#footnote-ref-8)
9. *order()* function in R [↑](#footnote-ref-9)