Problem 8: Association Rule Mining

2023-08-12

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(igraph)
```

```
##
## Attaching package: 'igraph'
##
## The following objects are masked from 'package:lubridate':
##
##     %--%, union
##
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
##
## The following objects are masked from 'package:purrr':
##
##     compose, simplify
##
## The following object is masked from 'package:tidyr':
##
##     crossing
##
## The following object is masked from 'package:tibble':
##
##     as_data_frame
##
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
##
## The following object is masked from 'package:base':
```

```
##
##     union
```

```
library(arules)  # has a big ecosystem of packages built around it
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
##
## Attaching package: 'arules'
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
library(arulesViz)
library(igraph)
```

```
grocery_raw = read.csv("groceries.txt", sep = " ", header = FALSE)
```

```
groceryrules = apriori(grocery_raw,
    parameter=list(support=.005, confidence=.1, maxlen=4))
```

```
## Warning: Column(s) 1, 2, 3, 4, 5, 6, 7 not logical or factor. Applying default
## discretization (see '? discretizeDF').
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.1    0.1    1 none FALSE            TRUE       5   0.005      1
##  maxlen target  ext
##       4  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 53
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[10321 item(s), 10636 transaction(s)] done [0.03s].
```

```
## sorting and recoding items ... [85 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4
```

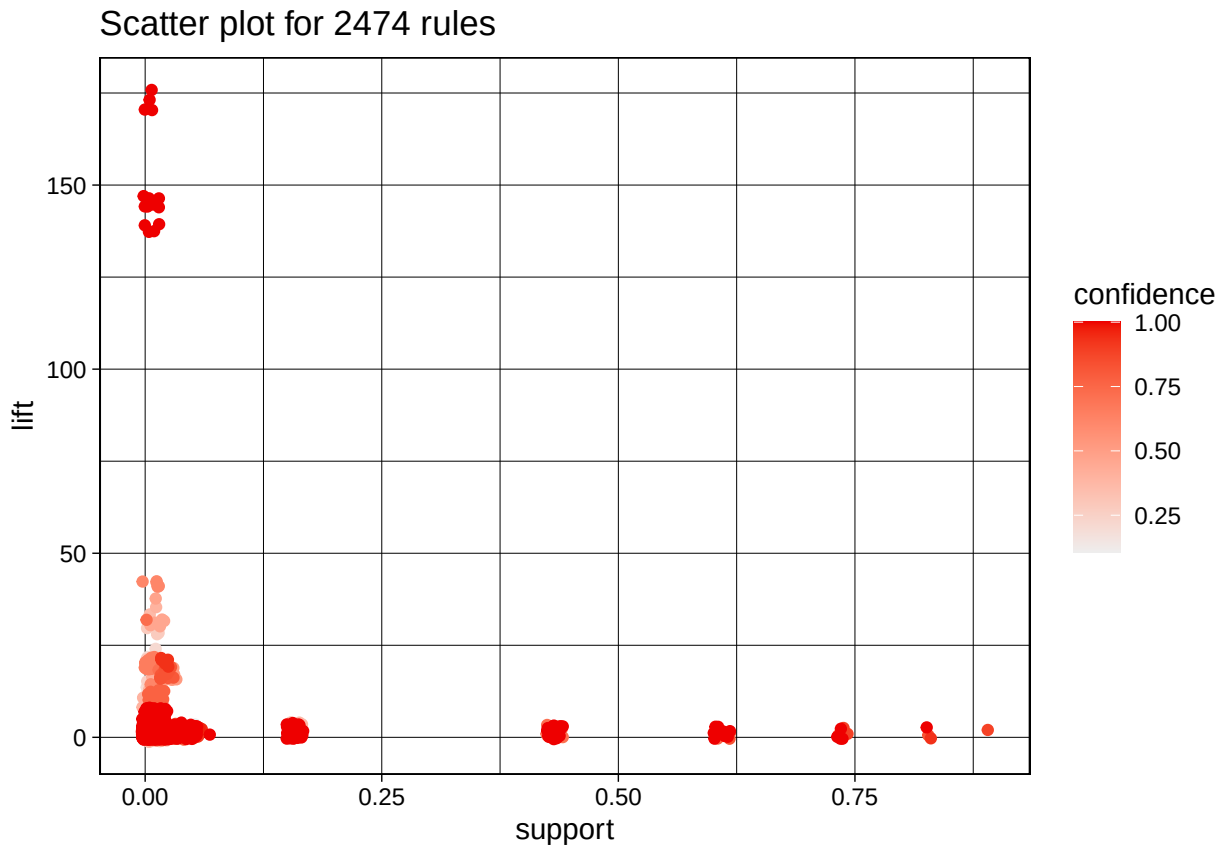
```
## Warning in apriori(grocery_raw, parameter = list(support = 0.005, confidence =
## 0.1, : Mining stopped (maxlen reached). Only patterns up to a length of 4
## returned!
```

```
##  done [0.00s].
## writing ... [2474 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

Support vs Confidence Scatter Plot

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

```
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```



Support gives us a measure of an association rule, telling us the fraction of grocery carts that contain both parts of the rule. For example, if your rule is that getting milk and eggs implies you also bought beer and cheese, then the support of the rule is the fraction of grocery carts that people have purchased with eggs, milk, beer, and cheese in them. The Lift is a different measure of the quality of the rule. The lift also takes into account likely it is your shopping cart contains a certain number of items in the first place. Sticking

with the same example, lift is ratio between the probability that you bought beer and cheese given the probability that you bought eggs and milk and the baseline probability that you bought eggs and milk. What the scatter plot tells us, then, is that most of our high lift rules as low support, which makes sense: if you punish having a high baseline probability of purchase, then your most informative rules will be the ones that associate rare pruchases.

```
transactions <- as(grocery_raw, "transactions")
```

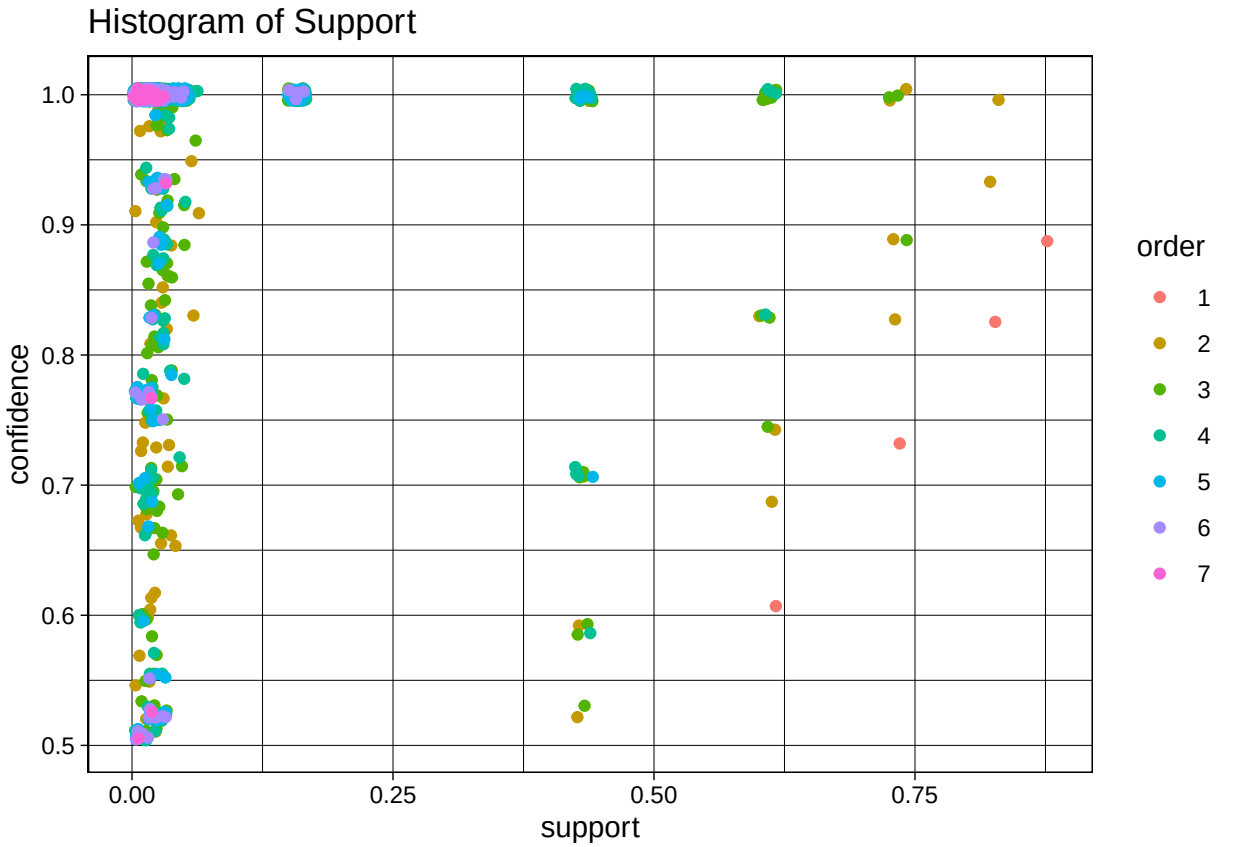
```
## Warning: Column(s) 1, 2, 3, 4, 5, 6, 7 not logical or factor. Applying default
## discretization (see '? discretizeDF').
```

```
# Mine association rules using apriori
rules <- apriori(transactions, parameter = list(support = 0.01, confidence = 0.5))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.5    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 106
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[10321 item(s), 10636 transaction(s)] done [0.02s].
## sorting and recoding items ... [38 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 7 done [0.00s].
## writing ... [1699 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
# Plotting a histogram of support
plot(rules, method = "two-key plot", measure = "support", main = "Histogram of Support")
```

```
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```

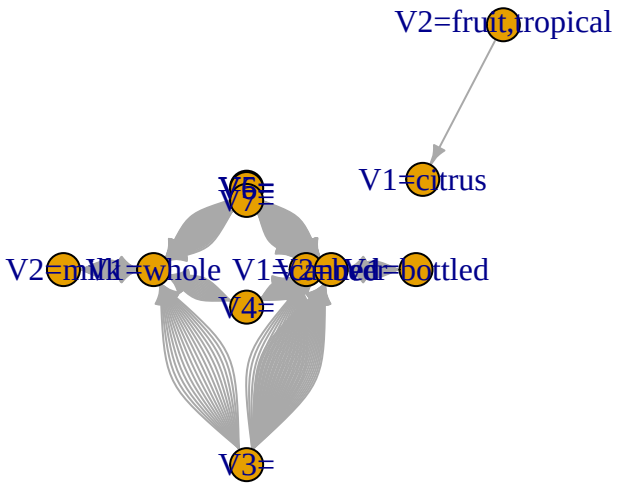


In the example we gave earlier, confidence is the probability that you bought beer and cheese given that you bought milk and eggs. This plot tells us a couple of things. First, for the rules that only associate one item, confidence is in direct correlation with support. This is because the rule is associating one item you bought with the empty set, which we think of as inherently existing in every cart. So for order 1, confidence and support are exactly the same. For some of the higher order rules we can see a similar problem: if all the actual items are on one side of the rule (and the empty set on the other), we get this linear effect where confidence = support. We can see another anomaly in the data points that appear at the very top: where confidence =1. These correspond to unique carts where one part of the rule is, say, the full cart of grocery items and the other is a unique subset of those items. That is, if only one person bought ham, sour cream, and eggs and then also bought milk, then the rule that associates ham, sour cream, and eggs with that full list of item will have perfect confidence (the three items perfectly predict the four). The final observation we should make is that higher order rules tend to have higher confidence. This is likely because the more items you use to predict the cart, the fewer times that cart appears in the data. That is, these rules pick out the larger carts in the data sets, because the more items you have, the more likely it is that no one else has bought that combination of items.

Association Rule Graph (Lift > 8)

```
grocery_graph = associations2igraph(subset(rules, lift>8), associationsAsNodes = FALSE)

layout <- layout_with_mds(grocery_graph)

# Save the layout coordinates to the graph
V(grocery_graph)$x <- layout[, 1]
V(grocery_graph)$y <- layout[, 2]

# Save the graph in GraphML format
```

```r
write_graph(grocery_graph, file = 'grocery.graphml', format = 'graphml')

# Read the graph with layout from the GraphML file
tograph <- read_graph('grocery.graphml', format = 'graphml')


# Plot the graph using the saved layout and modified labels
plot(tograph, edge.arrow.size = 0.5)
```




```r
# Print the full vertex labels as tooltips when hovering over nodes
tkplot(tograph, vertex.label = V(tograph)$name)
```

```
## [1] 1
```

```r
# Print the vertex numbers and their corresponding names
for (i in 1:vcount(tograph)) {
  vertex_number <- i
  vertex_name <- V(tograph)[i]$name
  vertex_label <- V(tograph)[i]$label
  cat(sprintf("%d = %s (%s)\n", vertex_number, vertex_name, vertex_label))
}
```

```
## 1 = 226 (V1=bottled)
## 2 = 315 (V1=canned)
```

```
## 3 = 454 (V1=citrus)
## 4 = 2123 (V1=whole)
## 5 = 2291 (V2=beer)
## 6 = 3151 (V2=fruit,tropical)
## 7 = 3456 (V2=milk)
## 8 = 4431 (V3=)
## 9 = 6264 (V4=)
## 10 = 7752 (V5=)
## 11 = 8879 (V6=)
## 12 = 9726 (V7=)
```

The graph above visualizes some of the association rules that have lift greater than 8. Running the code in rmd will generate an interactive plot of the graph which can help to visualize some of the more clustered rules. Additionally, a key is printed out that associates the tkplot vertex number with the original vertex name. We can gleen a couple of insights from this graph. First, we see that there is a strong association rule that says whenever you are buying "tropical fruit" you must be buying "citrus", indicating that potentially all of the tropical fruit items you can buy are considered citrus as well. Another insight as that the empty set (5 of the above vertices are unnamed, indicating the X in the rule is the empty itemset) can generate rules with high lift in our data set. This could happen if certain itemsets are common enough that they can be predicted with no information. For example, if 90% of shoppers buy milk and eggs, then the rule that predicts {} -> {milk, eggs} may have high lift.