Cartesian Item Pairings

Colby Fry

2025-04-25

## Introduction

This report explores item pairings within transaction data using a combination of custom item pairing logic and market basket analysis techniques. The goal is to uncover frequently co-purchased products and generate insights that can inform suggestive selling strategies and product bundling.

## Package Setup

The following R packages are used to perform rule mining, manipulate data efficiently, and handle transaction formats.

library(arules)

## Warning: package 'arules' was built under R version 4.3.3

## Loading required package: Matrix

##   
## Attaching package: 'arules'

## The following objects are masked from 'package:base':  
##   
## abbreviate, write

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:arules':  
##   
## intersect, recode, setdiff, setequal, union

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(data.table)

## Warning: package 'data.table' was built under R version 4.3.3

##   
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':  
##   
## between, first, last

library(purrr)

## Warning: package 'purrr' was built under R version 4.3.3

##   
## Attaching package: 'purrr'

## The following object is masked from 'package:data.table':  
##   
## transpose

The transaction data is first read from a CSV file. Each line represents a single item from a transaction. We ensure proper column naming and group the data by transaction ID to form item lists.

data <- fread("YearRoundFinal.csv", stringsAsFactors = FALSE)  
  
setnames(data, old = "Lineitem name", new = "Lineitem.name")  
  
transaction\_items <- data[!is.na(Lineitem.name), .(  
 ItemsList = list(unique(Lineitem.name))  
), by = Name]

The grouped item lists are converted into a format compatible with the arules package to enable association rule mining.

trans\_list <- transaction\_items$ItemsList  
names(trans\_list) <- transaction\_items$Name  
trans <- as(trans\_list, "transactions")

For each transaction containing two or more unique items, all possible unordered item pairs are generated. This creates a dataset of co-occurring item pairs for further analysis or visualization.

cartesian\_pairs <- map\_dfr(trans\_list, function(items) {  
 items <- unique(items)  
 if (length(items) >= 2) {  
 combinations <- combn(items, 2)  
 pair\_data <- data.frame(  
 Item1 = combinations[1, ],  
 Item2 = combinations[2, ],  
 stringsAsFactors = FALSE  
 )  
 return(pair\_data)  
 } else {  
 return(NULL)  
 }  
})

Next, we apply the Apriori algorithm to find item combinations that occur frequently and are strongly associated. Rules with minimum support of 1% and confidence of 50% are returned and previewed.

rules <- apriori(trans, parameter = list(supp = 0.01, conf = 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: 371   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[1352 item(s), 37115 transaction(s)] done [0.01s].  
## sorting and recoding items ... [68 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 done [0.00s].  
## writing ... [0 rule(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

inspect(head(rules))  
  
fwrite(cartesian\_pairs, "cartesian\_pairs.csv")

Finally, the item pairings generated earlier are exported to a CSV file for further analysis or visualization outside R.

fwrite(cartesian\_pairs, "cartesian\_pairs.csv")