Case Study 5

Shubh Shah

2024-11-19

## Clustering Analysis

For this portion, I will be random sampling 90% of the Iris database to perform K-means and Hierarchical clustering. Loaded in is the “fpc” library. I had first loaded in the iris data set, then had random sampled it. I had then displayed the statistics to understand the data I am working with.

data(iris)  
iris\_1 = iris[sample(nrow(iris), nrow(iris) \* 0.9), ]  
summary(iris\_1)

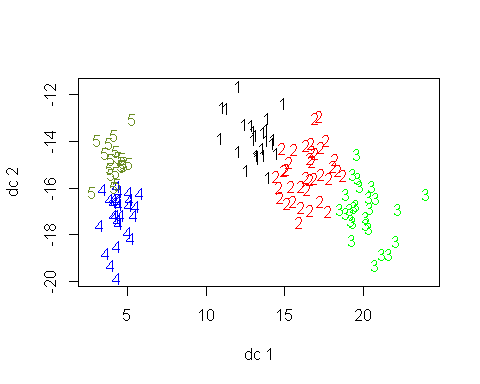
## Sepal.Length Sepal.Width Petal.Length Petal.Width   
## Min. :4.300 Min. :2.000 Min. :1.000 Min. :0.100   
## 1st Qu.:5.100 1st Qu.:2.800 1st Qu.:1.500 1st Qu.:0.300   
## Median :5.700 Median :3.000 Median :4.300 Median :1.300   
## Mean :5.821 Mean :3.061 Mean :3.699 Mean :1.173   
## 3rd Qu.:6.400 3rd Qu.:3.400 3rd Qu.:5.100 3rd Qu.:1.800   
## Max. :7.900 Max. :4.400 Max. :6.900 Max. :2.500   
## Species   
## setosa :47   
## versicolor:44   
## virginica :44   
##   
##   
##

I had then displayed a 5 cluster solution. To explain it better, a cluster would mean a group of observations with similar characterisitics based upon different metrics, example would include their distance or even hierarchial. In this scenario, we would have 5 different observations displayed.

fit <- kmeans(iris\_1[,1:4], 5) # This would represent a 5 cluster solution  
table(fit$cluster)

##   
## 1 2 3 4 5   
## 25 37 26 27 20

plotcluster(iris\_1[,1:4], fit$cluster)

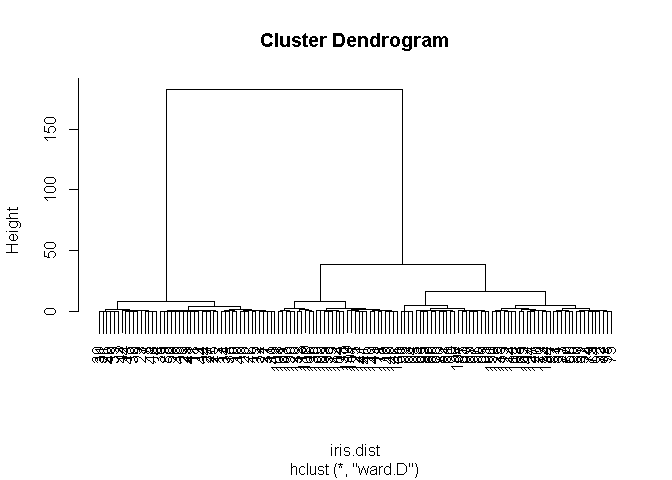


# After I had plotted it, I wanted to see what is in each group  
  
fit$centers # This would provide the cluster means

## Sepal.Length Sepal.Width Petal.Length Petal.Width  
## 1 5.512000 2.628000 3.944000 1.2120000  
## 2 6.251351 2.843243 4.859459 1.6243243  
## 3 6.969231 3.107692 5.857692 2.1576923  
## 4 5.248148 3.662963 1.496296 0.2851852  
## 5 4.690000 3.130000 1.410000 0.2100000

I had then performed Wards Method, where it would calculate the Euclidean distances between each observations from the first 4 columns of the data set. I then produced a dendrogram to visually show how it would be grouped.

iris.dist = dist(iris\_1[,1:4])  
iris.hclust = hclust(iris.dist, method = "ward.D")  
plot(iris.hclust)



From my dendrogram, I was confused into the cuts itself, I had then created multiple points to show different cuts established.

Ideally with 2, it would split it within 2 main groups.

groupIris.2 = cutree(iris.hclust, k = 2)  
table(groupIris.2)

## groupIris.2  
## 1 2   
## 88 47

When it is split within two groups, the values range between 40-50 and 80-90

groupIris.3 = cutree(iris.hclust, k = 3)  
table(groupIris.3)

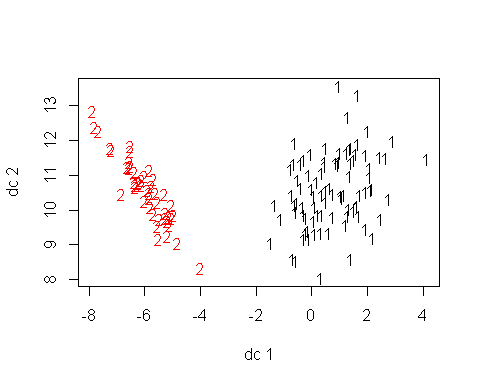
## groupIris.3  
## 1 2 3   
## 56 47 32

Here we had split between 3 and the values vary between 20-30-60 or 30-40-50. Occasionally a 30-40-60

aggregate(iris\_1[,1:4], by = list(groupIris.2), FUN = mean)

## Group.1 Sepal.Length Sepal.Width Petal.Length Petal.Width  
## 1 1 6.253409 2.860227 4.894318 1.6647727  
## 2 2 5.010638 3.436170 1.459574 0.2531915

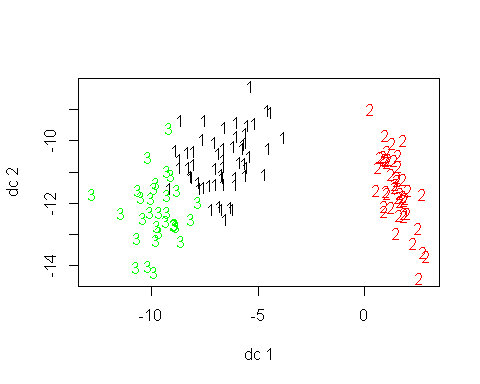
plotcluster(iris\_1[,1:4], groupIris.2)



aggregate(iris\_1[,1:4], by = list(groupIris.3), FUN = mean)

## Group.1 Sepal.Length Sepal.Width Petal.Length Petal.Width  
## 1 1 5.905357 2.739286 4.398214 1.4250000  
## 2 2 5.010638 3.436170 1.459574 0.2531915  
## 3 3 6.862500 3.071875 5.762500 2.0843750

plotcluster(iris\_1[,1:4], groupIris.3)

 We can now compare the plots between each other. From my previous runs, I had found that when split in two, certain categories such as Sepal.Length, Petal.Length, and Petal.Width would have a higher mean value which would represent a larger flower. Likewise, in previous run, I had a high Sepal.Width but a smaller Petal.Length and Petal.Width, which would represent a smaller flower. When I had made it into three clusters, we had a group that had corresponded with a median sized flower, one represented the largest, and one were the smallest. So it had done the split by size. This was seen by the means itself. Since my work would be random, it would be difficult to comment on these results.

## Association Rules

First, I had loaded in the preliminary data such as the library, and the Groceries data. The grocery dataset is the collection of items in a grocery store for a month. Understanding this data would provide us info in everyday shopping habits

library(arules)

## Warning: package 'arules' was built under R version 4.4.2

## Loading required package: Matrix

##   
## Attaching package: 'arules'

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

data("Groceries")

I had then performed data exploration. From my data exploration, I had found that the dimensions of the set were 9835 rows and 169 columns. The most frequently purchased item was milk, which was purchased a quarter of the time when shopping (or 2513 times within the month). The average transaction involved 4.4 items, whereas the largest was 32 items. This can be found within the summary or the itemFrequencyPlot

# Data Exploration   
  
# What are the dimensions of the grocery dataset  
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   
## 2513 1903 1809 1715   
## yogurt (Other)   
## 1372 34055   
##   
## element (itemset/transaction) length distribution:  
## sizes  
## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16   
## 2159 1643 1299 1005 855 645 545 438 350 246 182 117 78 77 55 46   
## 17 18 19 20 21 22 23 24 26 27 28 29 32   
## 29 14 14 9 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

# We are able to see that there are 9835 rows and 169 columns   
  
# What was the most frequently purchased item  
# This would be whole milk, look at either Frequency Item Plot or summary. Milk was purchased 1/4 of time when shopping.   
  
# Average Transaction involved how many items:  
# this would be 4.4 items   
  
#Largest transaction:   
# 32 items

Below would represent the first 10 transactions within the data. As we can see, just within the first 10 transaction, milk appears 5 times.

# Print out the first 10 transactions   
inspect(Groceries[1:10])

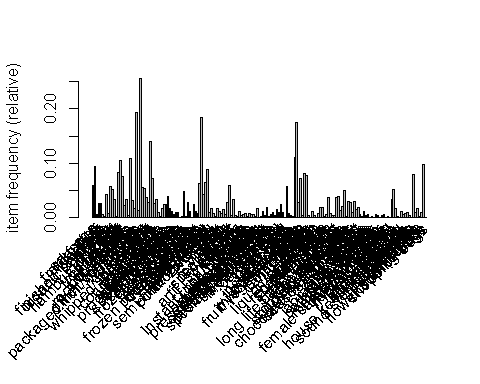
## items   
## [1] {citrus fruit,   
## semi-finished bread,   
## margarine,   
## ready soups}   
## [2] {tropical fruit,   
## yogurt,   
## coffee}   
## [3] {whole milk}   
## [4] {pip fruit,   
## yogurt,   
## cream cheese ,   
## meat spreads}   
## [5] {other vegetables,   
## whole milk,   
## condensed milk,   
## long life bakery product}  
## [6] {whole milk,   
## butter,   
## yogurt,   
## rice,   
## abrasive cleaner}   
## [7] {rolls/buns}   
## [8] {other vegetables,   
## UHT-milk,   
## rolls/buns,   
## bottled beer,   
## liquor (appetizer)}   
## [9] {pot plants}   
## [10] {whole milk,   
## cereals}

Below are going to be itemFrequency and itemFrequencyPlot, I had kept it default, which would show certain items with a value less than 0.1%. Ideally this would show all of the items. Once I had established a support and changed the size of the name, we would see that the chart drastically had cut down. We still see that milk is bought 25% of time when shopping, but also soda and rolls/buns are bought 20% shopping, fruits and vegetables are little bit above 10%, same with other vegetables.

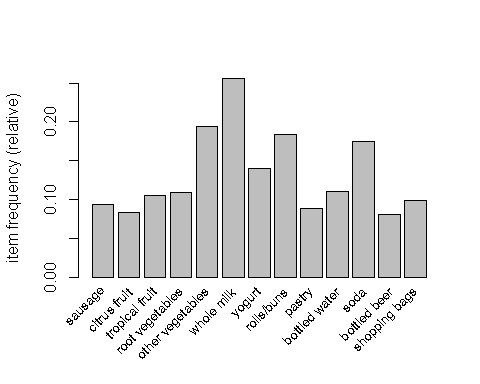
# Frequency of Each item  
itemFrequency(Groceries)

## frankfurter sausage liver loaf   
## 0.0589730554 0.0939501779 0.0050838841   
## ham meat finished products   
## 0.0260294865 0.0258261312 0.0065073716   
## organic sausage chicken turkey   
## 0.0022369090 0.0429079817 0.0081342145   
## pork beef hamburger meat   
## 0.0576512456 0.0524656838 0.0332486019   
## fish citrus fruit tropical fruit   
## 0.0029486528 0.0827656329 0.1049313676   
## pip fruit grapes berries   
## 0.0756481952 0.0223690900 0.0332486019   
## nuts/prunes root vegetables onions   
## 0.0033553635 0.1089984748 0.0310116929   
## herbs other vegetables packaged fruit/vegetables   
## 0.0162684291 0.1934926284 0.0130147433   
## whole milk butter curd   
## 0.2555160142 0.0554143366 0.0532791052   
## dessert butter milk yogurt   
## 0.0371123538 0.0279613625 0.1395017794   
## whipped/sour cream beverages UHT-milk   
## 0.0716827656 0.0260294865 0.0334519573   
## condensed milk cream soft cheese   
## 0.0102694459 0.0013218099 0.0170818505   
## sliced cheese hard cheese cream cheese   
## 0.0245043213 0.0245043213 0.0396542959   
## processed cheese spread cheese curd cheese   
## 0.0165734621 0.0111845450 0.0050838841   
## specialty cheese mayonnaise salad dressing   
## 0.0085409253 0.0091509914 0.0008134215   
## tidbits frozen vegetables frozen fruits   
## 0.0023385867 0.0480935435 0.0012201322   
## frozen meals frozen fish frozen chicken   
## 0.0283680732 0.0116929334 0.0006100661   
## ice cream frozen dessert frozen potato products   
## 0.0250127097 0.0107778343 0.0084392476   
## domestic eggs rolls/buns white bread   
## 0.0634468734 0.1839349263 0.0420945602   
## brown bread pastry roll products   
## 0.0648703610 0.0889679715 0.0102694459   
## semi-finished bread zwieback potato products   
## 0.0176919166 0.0069140824 0.0028469751   
## flour salt rice   
## 0.0173868836 0.0107778343 0.0076258261   
## pasta vinegar oil   
## 0.0150482969 0.0065073716 0.0280630402   
## margarine specialty fat sugar   
## 0.0585663447 0.0036603965 0.0338586680   
## artif. sweetener honey mustard   
## 0.0032536858 0.0015251652 0.0119979664   
## ketchup spices soups   
## 0.0042704626 0.0051855618 0.0068124047   
## ready soups Instant food products sauces   
## 0.0018301983 0.0080325369 0.0054905948   
## cereals organic products baking powder   
## 0.0056939502 0.0016268429 0.0176919166   
## preservation products pudding powder canned vegetables   
## 0.0002033554 0.0023385867 0.0107778343   
## canned fruit pickled vegetables specialty vegetables   
## 0.0032536858 0.0178952720 0.0017285206   
## jam sweet spreads meat spreads   
## 0.0053889171 0.0090493137 0.0042704626   
## canned fish dog food cat food   
## 0.0150482969 0.0085409253 0.0232841891   
## pet care baby food coffee   
## 0.0094560244 0.0001016777 0.0580579563   
## instant coffee tea cocoa drinks   
## 0.0074224708 0.0038637519 0.0022369090   
## bottled water soda misc. beverages   
## 0.1105236401 0.1743772242 0.0283680732   
## fruit/vegetable juice syrup bottled beer   
## 0.0722928317 0.0032536858 0.0805287239   
## canned beer brandy whisky   
## 0.0776817489 0.0041687850 0.0008134215   
## liquor rum liqueur   
## 0.0110828673 0.0044738180 0.0009150991   
## liquor (appetizer) white wine red/blush wine   
## 0.0079308592 0.0190137265 0.0192170819   
## prosecco sparkling wine salty snack   
## 0.0020335536 0.0055922725 0.0378240976   
## popcorn nut snack snack products   
## 0.0072191154 0.0031520081 0.0030503305   
## long life bakery product waffles cake bar   
## 0.0374173869 0.0384341637 0.0132180986   
## chewing gum chocolate cooking chocolate   
## 0.0210472801 0.0496187087 0.0025419420   
## specialty chocolate specialty bar chocolate marshmallow   
## 0.0304016268 0.0273512964 0.0090493137   
## candy seasonal products detergent   
## 0.0298932384 0.0142348754 0.0192170819   
## softener decalcifier dish cleaner   
## 0.0054905948 0.0015251652 0.0104728012   
## abrasive cleaner cleaner toilet cleaner   
## 0.0035587189 0.0050838841 0.0007117438   
## bathroom cleaner hair spray dental care   
## 0.0027452974 0.0011184545 0.0057956279   
## male cosmetics make up remover skin care   
## 0.0045754957 0.0008134215 0.0035587189   
## female sanitary products baby cosmetics soap   
## 0.0061006609 0.0006100661 0.0026436197   
## rubbing alcohol hygiene articles napkins   
## 0.0010167768 0.0329435689 0.0523640061   
## dishes cookware kitchen utensil   
## 0.0175902389 0.0027452974 0.0004067107   
## cling film/bags kitchen towels house keeping products   
## 0.0113879004 0.0059989832 0.0083375699   
## candles light bulbs sound storage medium   
## 0.0089476360 0.0041687850 0.0001016777   
## newspapers photo/film pot plants   
## 0.0798169802 0.0092526690 0.0172852059   
## flower soil/fertilizer flower (seeds) shopping bags   
## 0.0019318760 0.0103711235 0.0985256736   
## bags   
## 0.0004067107

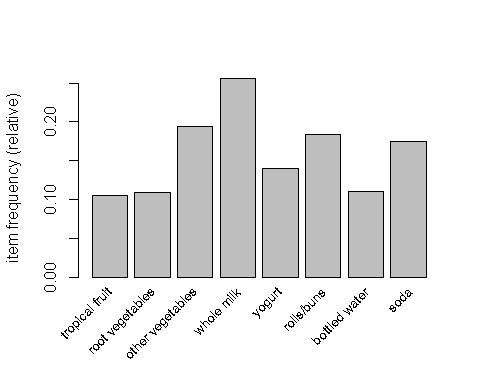
itemFrequencyPlot(Groceries)



# This will show for all items with no support, or possibly a very small support Everything is merged and we cannot see.   
  
  
itemFrequencyPlot(Groceries, support = 0.08, cex.names = 0.785)



# This will display the item frequency plot with a support of 8%, this drastically cut it down.   
  
  
itemFrequencyPlot(Groceries, support = 0.1, cex.names = 0.785)



# This will display the item frequency plot with a support of 10%, this drastically cut   
# it down. We are able to show that milk is bought 1/4 of the time each transaction

# This is an example on if I have a low support and a low confidence.  
# An item combination LHS and RHS appear atleast 0.1% of the total transaction to be considered - support  
# Confidence 1% where when RHS is bought, so is LHS.   
rules <- apriori(Groceries, parameter = list(support = 0.001, confidence = 0.01))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.01 0.1 1 none FALSE TRUE 5 0.001 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: 9   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].  
## sorting and recoding items ... [157 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 3 4 5 6 done [0.01s].  
## writing ... [40887 rule(s)] done [0.00s].  
## creating S4 object ... done [0.01s].

length(rules)

## [1] 40887

summary(rules)

## set of 40887 rules  
##   
## rule length distribution (lhs + rhs):sizes  
## 1 2 3 4 5 6   
## 88 5818 20493 12548 1880 60   
##   
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.000 3.000 3.000 3.257 4.000 6.000   
##   
## summary of quality measures:  
## support confidence coverage lift   
## Min. :0.001017 Min. :0.01012 Min. :0.001017 Min. : 0.3109   
## 1st Qu.:0.001118 1st Qu.:0.11794 1st Qu.:0.003965 1st Qu.: 1.7989   
## Median :0.001423 Median :0.21127 Median :0.007524 Median : 2.4086   
## Mean :0.002154 Mean :0.26188 Mean :0.018325 Mean : 2.6753   
## 3rd Qu.:0.002034 3rd Qu.:0.36667 3rd Qu.:0.016574 3rd Qu.: 3.2357   
## Max. :0.255516 Max. :1.00000 Max. :1.000000 Max. :35.7158   
## count   
## Min. : 10.00   
## 1st Qu.: 11.00   
## Median : 14.00   
## Mean : 21.18   
## 3rd Qu.: 20.00   
## Max. :2513.00   
##   
## mining info:  
## data ntransactions support confidence  
## Groceries 9835 0.001 0.01  
## call  
## apriori(data = Groceries, parameter = list(support = 0.001, confidence = 0.01))

With low parameters, we are able to see more than 10,000 rules are written to certain items. Within this, we are able to see that our lift is 2.6753. Now say we changed up those parameters. Below I had changed it to a support of 2.5% and a confidence of 10%. After running it, we are able to see that 75 rules have been written. 8 of the rules involve 1 item and 67 rules involve 2 items. We are able to see that the average of the lift is 1.4945

# As we can see, when we change it up to a support of 2.5% and a confidence of 10%,   
# only 75 rules were written   
rules <- apriori(Groceries, parameter = list(support = 0.025, confidence = 0.1))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.1 0.1 1 none FALSE TRUE 5 0.025 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: 245   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].  
## sorting and recoding items ... [54 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 3 done [0.00s].  
## writing ... [75 rule(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

length(rules)

## [1] 75

summary(rules)

## set of 75 rules  
##   
## rule length distribution (lhs + rhs):sizes  
## 1 2   
## 8 67   
##   
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.000 2.000 2.000 1.893 2.000 2.000   
##   
## summary of quality measures:  
## support confidence coverage lift   
## Min. :0.02522 Min. :0.1023 Min. :0.05328 Min. :0.8991   
## 1st Qu.:0.02888 1st Qu.:0.1530 1st Qu.:0.10696 1st Qu.:1.2050   
## Median :0.03274 Median :0.2198 Median :0.18393 Median :1.5036   
## Mean :0.04860 Mean :0.2442 Mean :0.25320 Mean :1.4945   
## 3rd Qu.:0.04342 3rd Qu.:0.3339 3rd Qu.:0.25552 3rd Qu.:1.7678   
## Max. :0.25552 Max. :0.4972 Max. :1.00000 Max. :2.2466   
## count   
## Min. : 248   
## 1st Qu.: 284   
## Median : 322   
## Mean : 478   
## 3rd Qu.: 427   
## Max. :2513   
##   
## mining info:  
## data ntransactions support confidence  
## Groceries 9835 0.025 0.1  
## call  
## apriori(data = Groceries, parameter = list(support = 0.025, confidence = 0.1))

# 8 rules involve 1 items, and 67 rules involve 2 items.   
# average of the lift is 1.4945

After running those parameters, we are able to see the details in regards to the associations rules. For example, to interpret these results, we see rule one regards to bottle water. Its support is 11.05% and confidence is 11.05%. Its lift is 1, since there is no LHS. Likewise if we go to rule 9 where the LHS is curd and RHS is Whole Milk, we are able to see that the support is 0.0261 or 2.61%. This means that 2.61% include both curd and whole milk. With a confidence of 0.4905, it means 49.05% of transactions with curd would also include milk. Our lift was 1.919, meaning people who buy curd are 1.92 times likely to buy milk. I had then printed 10 rules and had sorted by the lift values.

inspect(rules)

## lhs rhs support confidence  
## [1] {} => {bottled water} 0.11052364 0.1105236   
## [2] {} => {tropical fruit} 0.10493137 0.1049314   
## [3] {} => {root vegetables} 0.10899847 0.1089985   
## [4] {} => {soda} 0.17437722 0.1743772   
## [5] {} => {yogurt} 0.13950178 0.1395018   
## [6] {} => {rolls/buns} 0.18393493 0.1839349   
## [7] {} => {other vegetables} 0.19349263 0.1934926   
## [8] {} => {whole milk} 0.25551601 0.2555160   
## [9] {curd} => {whole milk} 0.02613116 0.4904580   
## [10] {whole milk} => {curd} 0.02613116 0.1022682   
## [11] {brown bread} => {whole milk} 0.02521607 0.3887147   
## [12] {butter} => {whole milk} 0.02755465 0.4972477   
## [13] {whole milk} => {butter} 0.02755465 0.1078392   
## [14] {newspapers} => {whole milk} 0.02735130 0.3426752   
## [15] {whole milk} => {newspapers} 0.02735130 0.1070434   
## [16] {domestic eggs} => {whole milk} 0.02999492 0.4727564   
## [17] {whole milk} => {domestic eggs} 0.02999492 0.1173896   
## [18] {fruit/vegetable juice} => {whole milk} 0.02663955 0.3684951   
## [19] {whole milk} => {fruit/vegetable juice} 0.02663955 0.1042579   
## [20] {whipped/sour cream} => {other vegetables} 0.02887646 0.4028369   
## [21] {other vegetables} => {whipped/sour cream} 0.02887646 0.1492380   
## [22] {whipped/sour cream} => {whole milk} 0.03223183 0.4496454   
## [23] {whole milk} => {whipped/sour cream} 0.03223183 0.1261441   
## [24] {pip fruit} => {other vegetables} 0.02613116 0.3454301   
## [25] {other vegetables} => {pip fruit} 0.02613116 0.1350499   
## [26] {pip fruit} => {whole milk} 0.03009659 0.3978495   
## [27] {whole milk} => {pip fruit} 0.03009659 0.1177875   
## [28] {pastry} => {whole milk} 0.03324860 0.3737143   
## [29] {whole milk} => {pastry} 0.03324860 0.1301234   
## [30] {citrus fruit} => {other vegetables} 0.02887646 0.3488943   
## [31] {other vegetables} => {citrus fruit} 0.02887646 0.1492380   
## [32] {citrus fruit} => {whole milk} 0.03050330 0.3685504   
## [33] {whole milk} => {citrus fruit} 0.03050330 0.1193792   
## [34] {sausage} => {rolls/buns} 0.03060498 0.3257576   
## [35] {rolls/buns} => {sausage} 0.03060498 0.1663903   
## [36] {sausage} => {other vegetables} 0.02694459 0.2867965   
## [37] {other vegetables} => {sausage} 0.02694459 0.1392538   
## [38] {sausage} => {whole milk} 0.02989324 0.3181818   
## [39] {whole milk} => {sausage} 0.02989324 0.1169916   
## [40] {bottled water} => {soda} 0.02897814 0.2621895   
## [41] {soda} => {bottled water} 0.02897814 0.1661808   
## [42] {bottled water} => {whole milk} 0.03436706 0.3109476   
## [43] {whole milk} => {bottled water} 0.03436706 0.1345006   
## [44] {tropical fruit} => {yogurt} 0.02928317 0.2790698   
## [45] {yogurt} => {tropical fruit} 0.02928317 0.2099125   
## [46] {tropical fruit} => {other vegetables} 0.03589222 0.3420543   
## [47] {other vegetables} => {tropical fruit} 0.03589222 0.1854966   
## [48] {tropical fruit} => {whole milk} 0.04229792 0.4031008   
## [49] {whole milk} => {tropical fruit} 0.04229792 0.1655392   
## [50] {root vegetables} => {yogurt} 0.02582613 0.2369403   
## [51] {yogurt} => {root vegetables} 0.02582613 0.1851312   
## [52] {root vegetables} => {other vegetables} 0.04738180 0.4347015   
## [53] {other vegetables} => {root vegetables} 0.04738180 0.2448765   
## [54] {root vegetables} => {whole milk} 0.04890696 0.4486940   
## [55] {whole milk} => {root vegetables} 0.04890696 0.1914047   
## [56] {soda} => {yogurt} 0.02735130 0.1568513   
## [57] {yogurt} => {soda} 0.02735130 0.1960641   
## [58] {soda} => {rolls/buns} 0.03833249 0.2198251   
## [59] {rolls/buns} => {soda} 0.03833249 0.2084024   
## [60] {soda} => {other vegetables} 0.03274021 0.1877551   
## [61] {other vegetables} => {soda} 0.03274021 0.1692065   
## [62] {soda} => {whole milk} 0.04006101 0.2297376   
## [63] {whole milk} => {soda} 0.04006101 0.1567847   
## [64] {yogurt} => {rolls/buns} 0.03436706 0.2463557   
## [65] {rolls/buns} => {yogurt} 0.03436706 0.1868436   
## [66] {yogurt} => {other vegetables} 0.04341637 0.3112245   
## [67] {other vegetables} => {yogurt} 0.04341637 0.2243826   
## [68] {yogurt} => {whole milk} 0.05602440 0.4016035   
## [69] {whole milk} => {yogurt} 0.05602440 0.2192598   
## [70] {rolls/buns} => {other vegetables} 0.04260295 0.2316197   
## [71] {other vegetables} => {rolls/buns} 0.04260295 0.2201787   
## [72] {rolls/buns} => {whole milk} 0.05663447 0.3079049   
## [73] {whole milk} => {rolls/buns} 0.05663447 0.2216474   
## [74] {other vegetables} => {whole milk} 0.07483477 0.3867578   
## [75] {whole milk} => {other vegetables} 0.07483477 0.2928770   
## coverage lift count  
## [1] 1.00000000 1.0000000 1087   
## [2] 1.00000000 1.0000000 1032   
## [3] 1.00000000 1.0000000 1072   
## [4] 1.00000000 1.0000000 1715   
## [5] 1.00000000 1.0000000 1372   
## [6] 1.00000000 1.0000000 1809   
## [7] 1.00000000 1.0000000 1903   
## [8] 1.00000000 1.0000000 2513   
## [9] 0.05327911 1.9194805 257   
## [10] 0.25551601 1.9194805 257   
## [11] 0.06487036 1.5212930 248   
## [12] 0.05541434 1.9460530 271   
## [13] 0.25551601 1.9460530 271   
## [14] 0.07981698 1.3411103 269   
## [15] 0.25551601 1.3411103 269   
## [16] 0.06344687 1.8502027 295   
## [17] 0.25551601 1.8502027 295   
## [18] 0.07229283 1.4421604 262   
## [19] 0.25551601 1.4421604 262   
## [20] 0.07168277 2.0819237 284   
## [21] 0.19349263 2.0819237 284   
## [22] 0.07168277 1.7597542 317   
## [23] 0.25551601 1.7597542 317   
## [24] 0.07564820 1.7852365 257   
## [25] 0.19349263 1.7852365 257   
## [26] 0.07564820 1.5570432 296   
## [27] 0.25551601 1.5570432 296   
## [28] 0.08896797 1.4625865 327   
## [29] 0.25551601 1.4625865 327   
## [30] 0.08276563 1.8031403 284   
## [31] 0.19349263 1.8031403 284   
## [32] 0.08276563 1.4423768 300   
## [33] 0.25551601 1.4423768 300   
## [34] 0.09395018 1.7710480 301   
## [35] 0.18393493 1.7710480 301   
## [36] 0.09395018 1.4822091 265   
## [37] 0.19349263 1.4822091 265   
## [38] 0.09395018 1.2452520 294   
## [39] 0.25551601 1.2452520 294   
## [40] 0.11052364 1.5035766 285   
## [41] 0.17437722 1.5035766 285   
## [42] 0.11052364 1.2169396 338   
## [43] 0.25551601 1.2169396 338   
## [44] 0.10493137 2.0004746 288   
## [45] 0.13950178 2.0004746 288   
## [46] 0.10493137 1.7677896 353   
## [47] 0.19349263 1.7677896 353   
## [48] 0.10493137 1.5775950 416   
## [49] 0.25551601 1.5775950 416   
## [50] 0.10899847 1.6984751 254   
## [51] 0.13950178 1.6984751 254   
## [52] 0.10899847 2.2466049 466   
## [53] 0.19349263 2.2466049 466   
## [54] 0.10899847 1.7560310 481   
## [55] 0.25551601 1.7560310 481   
## [56] 0.17437722 1.1243678 269   
## [57] 0.13950178 1.1243678 269   
## [58] 0.17437722 1.1951242 377   
## [59] 0.18393493 1.1951242 377   
## [60] 0.17437722 0.9703476 322   
## [61] 0.19349263 0.9703476 322   
## [62] 0.17437722 0.8991124 394   
## [63] 0.25551601 0.8991124 394   
## [64] 0.13950178 1.3393633 338   
## [65] 0.18393493 1.3393633 338   
## [66] 0.13950178 1.6084566 427   
## [67] 0.19349263 1.6084566 427   
## [68] 0.13950178 1.5717351 551   
## [69] 0.25551601 1.5717351 551   
## [70] 0.18393493 1.1970465 419   
## [71] 0.19349263 1.1970465 419   
## [72] 0.18393493 1.2050318 557   
## [73] 0.25551601 1.2050318 557   
## [74] 0.19349263 1.5136341 736   
## [75] 0.25551601 1.5136341 736

# using the inspect rules to see by lift   
inspect(head(sort(rules, by = "lift"), n = 10))

## lhs rhs support confidence  
## [1] {root vegetables} => {other vegetables} 0.04738180 0.4347015   
## [2] {other vegetables} => {root vegetables} 0.04738180 0.2448765   
## [3] {whipped/sour cream} => {other vegetables} 0.02887646 0.4028369   
## [4] {other vegetables} => {whipped/sour cream} 0.02887646 0.1492380   
## [5] {tropical fruit} => {yogurt} 0.02928317 0.2790698   
## [6] {yogurt} => {tropical fruit} 0.02928317 0.2099125   
## [7] {butter} => {whole milk} 0.02755465 0.4972477   
## [8] {whole milk} => {butter} 0.02755465 0.1078392   
## [9] {curd} => {whole milk} 0.02613116 0.4904580   
## [10] {whole milk} => {curd} 0.02613116 0.1022682   
## coverage lift count  
## [1] 0.10899847 2.246605 466   
## [2] 0.19349263 2.246605 466   
## [3] 0.07168277 2.081924 284   
## [4] 0.19349263 2.081924 284   
## [5] 0.10493137 2.000475 288   
## [6] 0.13950178 2.000475 288   
## [7] 0.05541434 1.946053 271   
## [8] 0.25551601 1.946053 271   
## [9] 0.05327911 1.919481 257   
## [10] 0.25551601 1.919481 257

Rule 1: {root vegetables} <=> {other vegetables} Customers who buy root vegetables are 2.25 times more likely to buy other vegetables, and vice versa. This association occurs in 4.74% of transactions.

Rule 2: {whipped/sour cream} <=> {other vegetables} Customers who buy whipped/sour cream are 2.08 times more likely to buy other vegetables, and vice versa. This relationship is present in 2.89% of transactions

Rule 3: {tropical fruit} <=> {yogurt} Customers who buy tropical fruit are 2.00 times more likely to buy yogurt, and vice versa. This association occurs in 2.93% of transactions

Rule 4: {butter} <=> {whole milk} Customers who buy butter are 1.95 times more likely to buy whole milk, and vice versa. This association occurs in 2.76% of transactions

Rule 5: {curd} <=> {whole milk} Customers who buy curd are 1.92 times more likely to buy whole milk, and vice versa. This relationship is present in 2.61% of transactions