R Exercise: Association Rule Mining

Pilsung Kang
School of Industrial Management Engineering
Korea University

Association Rule Mining: Packages

- Package "arules" & "arulesViz

Package 'arules'

December 3, 2018

Version 1.6-2

Date 2018-12-02

Title Mining Association Rules and Frequent Itemsets

Description Provides the infrastructure for representing, manipulating and analyzing transaction data and patterns (frequent itemsets and association rules). Also provides C implementations of the association mining algorithms Apriori and Eclat.

Classification/ACM G.4, H.2.8, I.5.1

URL https://github.com/mhahsler/arules, http://lyle.smu.edu/IDA/arules

BugReports https://github.com/mhahsler/arules

Package 'arulesViz'

December 5, 2018

Version 1.3-2

Date 2018-12-04

Title Visualizing Association Rules and Frequent Itemsets

Depends arules (>= 1.4.1), grid

Imports scatterplot3d, vcd, seriation, igraph (>= 1.0.0), graphics, methods, utils, grDevices, stats, colorspace, DT, plotly, visNetwork

Suggests graph, Rgraphviz, iplots, shiny, htmlwidgets

Description Extends package 'arules' with various visualization techniques for association rules and itemsets. The package also includes several interactive visualizations for rule exploration.

License GPL-3

URL https://github.com/mhahsler/arulesViz, http://lyle.smu.edu/IDA/arules/

BugReports https://github.com/mhahsler/arulesViz

R Exercise: Load Dataset

- Load dataset

```
# Part 1: Transform a data file into transaction format
# Basket type
tmp_basket <- read.transactions("Transaction_Sample_Basket.csv",
         format = "basket", sep = ",", rm.duplicates=TRUE)
inspect(tmp_basket)
# Single type
tmp_single <- read.transactions("Transaction_Sample_Single.csv",
         format = "single", cols = c(1,2), rm.duplicates=TRUE)
inspect(tmp_single)
```

✓ read.basket( ) function can convert two types of dataset into a transaction format

- basket format: row is associated with transaction id and column is associated with the items in the corresponding id
- single format: each row consists of transaction id and one item in the corresponding id

R Exercise: Load Dataset

- Load dataset
  - ✓ Basket format and after conversion

	A	B	C	D	E
1	A	B	C		
2	A	C	D	E	
3	A	E	B		
4	B	C	D		
5	F	A	B		
6	A	D	F	G	
7	G	F	B	C	E
8	A	B			
9	C	D			
10	C	F	G		


```
> inspect(tmp_basket)
      items
[1]   {A,B,C}
[2]   {A,C,D,E}
[3]   {A,B,E}
[4]   {B,C,D}
[5]   {A,B,F}
[6]   {A,D,F,G}
[7]   {B,C,E,F,G}
[8]   {A,B}
[9]   {C,D}
[10]  {C,F,G}
```

R Exercise: Load Dataset

- Load dataset
  - ✓ Single format and after conversion

	A	B
1	Tr1 A	
2	Tr1 B	
3	Tr1 C	
4	Tr2 A	
5	Tr2 C	
6	Tr2 D	
7	Tr2 E	
8	Tr3 A	
9	Tr3 E	
10	Tr3 B	
11	Tr4 B	
12	Tr4 C	
13	Tr4 D	
14	Tr5 F	
15	Tr5 A	
16	Tr5 B	
17	Tr6 A	
18	Tr6 D	
19	Tr6 F	
20	Tr6 G	

...


```
> inspect(tmp_single)
     items       transactionID
[1]  {A,B,C}     Tr1
[2]  {C,G}       Tr10
[3]  {A,C,D,E}   Tr2
[4]  {A,B,E}     Tr3
[5]  {B,C,D}     Tr4
[6]  {A,B,F}     Tr5
[7]  {A,D,F,G}   Tr6
[8]  {B,C,E,F,G} Tr7
[9]  {A,B}       Tr8
[10] {C}         Tr9
```

R Exercise: Market Basket Analysis

- Load the dataset

```
# Part 2: Association Rule Mining without sequence information
data("Groceries")
summary(Groceries)
str(Groceries)
inspect(Groceries)
```

✓ Use the "Groceries" dataset (it is already installed if you have "arules" package installed

- Transaction data format, sparse matrix, provide some useful summary information

```
> 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           yogurt
            2513             1903             1809             1715             1372
          (Other)
            34055

element (itemset/transaction) length distribution:
sizes
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17
2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55   46   29
  18   19   20   21   22   23   24   26   27   28   29   32
  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
```

R Exercise: Market Basket Analysis

- Draw Wordcloud using the items

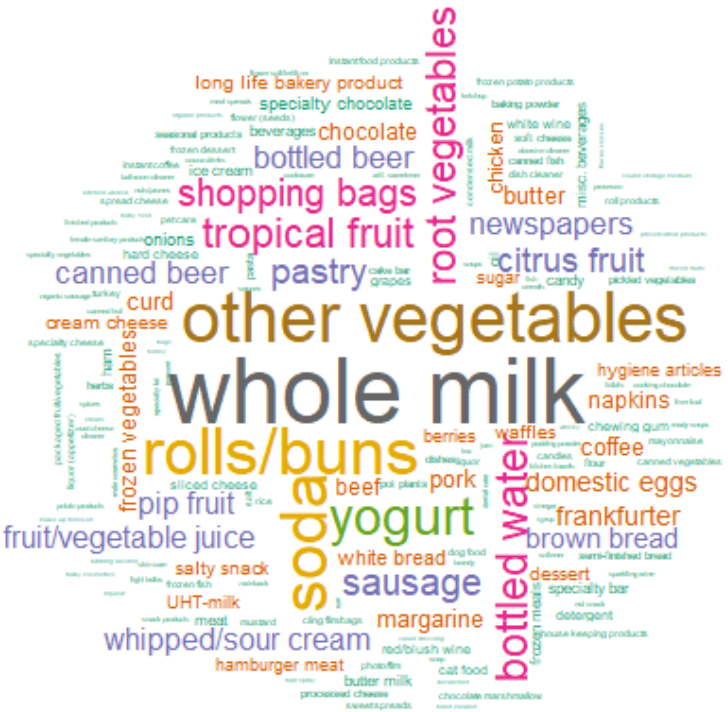
```
# Item inspection
itemName <- itemLabels(Groceries)
itemCount <- itemFrequency(Groceries)*nrow(Groceries)
col <- brewer.pal(8, "Dark2")
wordcloud(words = itemName, freq = itemCount, min.freq = 1,
          scale = c(3, 0.2), col = col, random.order = FALSE)
```

- ✓ itemName: item names used in the Wordcloud
- ✓ itemCount: item occurrence count used in the Wordcloud
- ✓ brewer.pal( ): color palette (usually choose one from predefined sets)
- ✓ wordcloud( ): Wordcloud generation function
  - words: used words, freq: item occurrence count, min.freq: minimum number of occurrence to be displayed, scale: relative scale between the most frequently bought item and the least frequently bought item

R Exercise: Market Basket Analysis

- Draw Wordcloud using the items

```
# Item inspection
itemName <- itemLabels(Groceries)
itemCount <- itemFrequency(Groceries)*nrow(Groceries)
col <- brewer.pal(8, "Dark2")
wordcloud(words = itemName, freq = itemCount, min.freq = 1,
          scale = c(3, 0.2), col = col, random.order = FALSE)
```

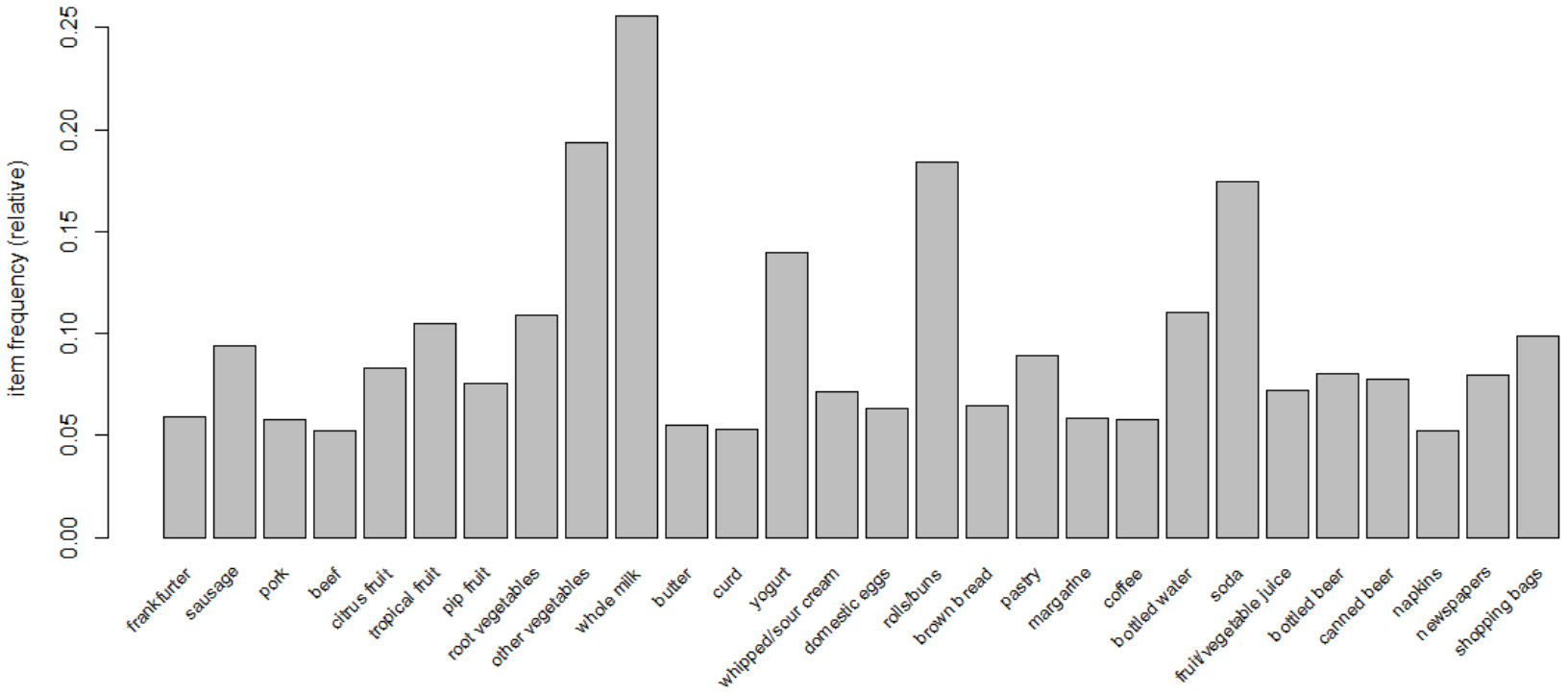


R Exercise: Market Basket Analysis

- Item frequency plot

```
itemFrequencyPlot(Groceries, support = 0.05, cex.names=0.8)
```

✓ Items with the frequency greater than 0.05 are displayed



R Exercise: Market Basket Analysis

- Data Preprocessing
  - ✓ Categorize a numeric variable, remove NA, etc.

```
15  # Remove "Name" column and group "Age" column
16  titanic_ar <- titanic[,2:5]
17  titanic_ar$Age = as.character(titanic_ar$Age)
18  c_idx <- which(as.numeric(titanic_ar$Age) < 20)
19  a_idx <- which(as.numeric(titanic_ar$Age) >= 20)
20  na_idx <- which(is.na(titanic_ar$Age))
21
22  titanic_ar$Age[c_idx] <- "Child"
23  titanic_ar$Age[a_idx] <- "Adult"
24  titanic_ar$Age[na_idx] <- "Unknown"
25
26  # Convert the attribues to factor
27  titanic_ar$Age <- as.factor(titanic_ar$Age)
28  titanic_ar$Survived <- as.factor(titanic_ar$Survived)
```

	PClass	Age	Sex	Survived
1	1st	Adult	female	1
2	1st	Child	female	0
3	1st	Adult	male	0
4	1st	Adult	female	0
5	1st	Child	male	1
6	1st	Adult	male	1
7	1st	Adult	female	1
8	1st	Adult	male	0
9	1st	Adult	female	1
10	1st	Adult	male	0

R Exercise: Market Basket Analysis

- Association rule generation

```
# Rule generation by Apriori
rules <- apriori(Groceries, parameter=list(support=0.01, confidence=0.35))

# Check the generated rules
inspect(rules)

# List the first three rules with the highest lift values
inspect(sort(rules, by="lift"))
```

✓ Directions insider the inspect( ) function means that the rules are displayed in an descending order

```
> inspect(sort(rules, by="lift"))
     lhs                                          rhs                  support    confidence lift     count
[1]  {citrus fruit,other vegetables}           => {root vegetables}    0.01037112 0.3591549  3.295045 102
[2]  {citrus fruit,root vegetables}            => {other vegetables}   0.01037112 0.5862069  3.029608 102
[3]  {tropical fruit,root vegetables}          => {other vegetables}   0.01230300 0.5845411  3.020999 121
[4]  {whole milk,curd}                         => {yogurt}             0.01006609 0.3852140  2.761356  99
[5]  {root vegetables,rolls/buns}              => {other vegetables}   0.01220132 0.5020921  2.594890 120
[6]  {root vegetables,yogurt}                  => {other vegetables}   0.01291307 0.5000000  2.584078 127
[7]  {tropical fruit,whole milk}               => {yogurt}             0.01514997 0.3581731  2.567516 149
[8]  {yogurt,whipped/sour cream}               => {other vegetables}   0.01016777 0.4901961  2.533410 100
[9]  {other vegetables,whipped/sour cream}     => {yogurt}             0.01016777 0.3521127  2.524073 100
[10] {root vegetables,whole milk}              => {other vegetables}   0.02318251 0.4740125  2.449770 228
```

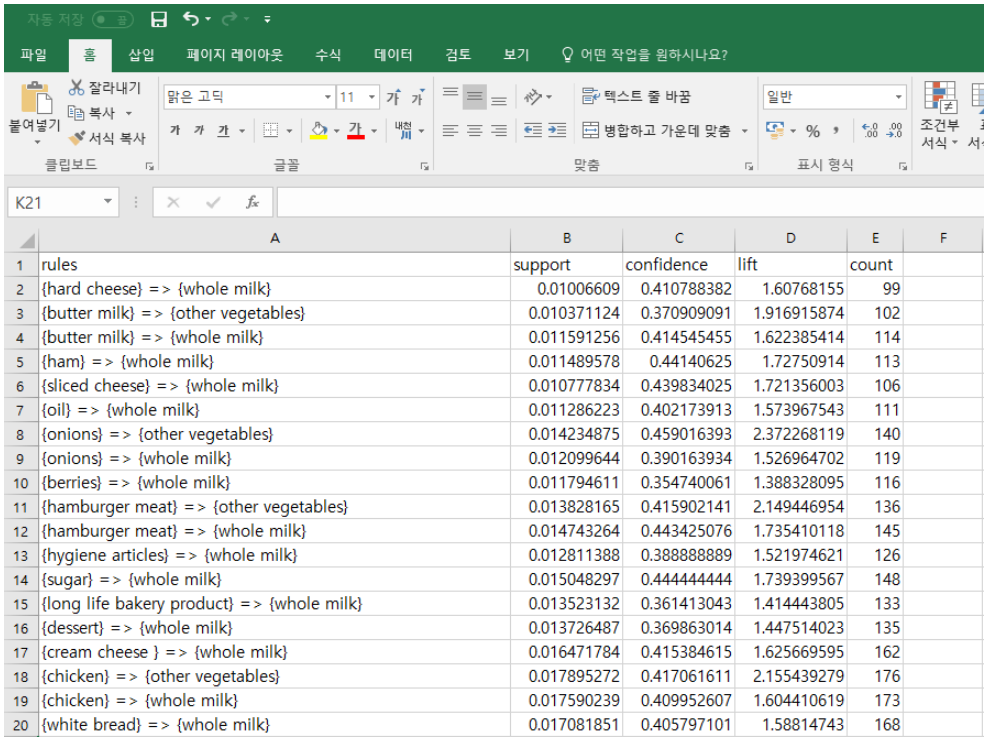
R Exercise: Market Basket Analysis

- Export the generated rules

```
# Save the rules in a text file
write.csv(as(rules, "data.frame"), "Groceries_rules.csv", row.names = FALSE)
```

✓ Convert the generated rules to data.frame format and export it as a csv file

- MS Excel file format (ex: xlsx) is not recommended (too slow!)



	A	B	C	D	E	F
1	rules	support	confidence	lift	count	
2	{hard cheese} => {whole milk}	0.01006609	0.410788382	1.60768155	99	
3	{butter milk} => {other vegetables}	0.010371124	0.370909091	1.916915874	102	
4	{butter milk} => {whole milk}	0.011591256	0.414545455	1.622385414	114	
5	{ham} => {whole milk}	0.011489578	0.44140625	1.72750914	113	
6	{sliced cheese} => {whole milk}	0.010777834	0.439834025	1.721356003	106	
7	{oil} => {whole milk}	0.011286223	0.402173913	1.573967543	111	
8	{onions} => {other vegetables}	0.014234875	0.459016393	2.372268119	140	
9	{onions} => {whole milk}	0.012099644	0.390163934	1.526964702	119	
10	{berries} => {whole milk}	0.011794611	0.354740061	1.388328095	116	
11	{hamburger meat} => {other vegetables}	0.013828165	0.415902141	2.149446954	136	
12	{hamburger meat} => {whole milk}	0.014743264	0.443425076	1.735410118	145	
13	{hygiene articles} => {whole milk}	0.012811388	0.388888889	1.521974621	126	
14	{sugar} => {whole milk}	0.015048297	0.444444444	1.739399567	148	
15	{long life bakery product} => {whole milk}	0.013523132	0.361413043	1.414443805	133	
16	{dessert} => {whole milk}	0.013726487	0.369863014	1.447514023	135	
17	{cream cheese } => {whole milk}	0.016471784	0.415384615	1.625669595	162	
18	{chicken} => {other vegetables}	0.017895272	0.417061611	2.155439279	176	
19	{chicken} => {whole milk}	0.017590239	0.409952607	1.604410619	173	
20	{white bread} => {whole milk}	0.017081851	0.405797101	1.58814743	168	

R Exercise: Market Basket Analysis

- Draw plots for the generated rules

```
# Plot the rules
plot(rules, method = "scatterplot")
plotly_arules(rules, method = "scatterplot", measure = c("support", "confidence"),
          shading = "lift")
```

✓ plot( ) function generates a fixed plot

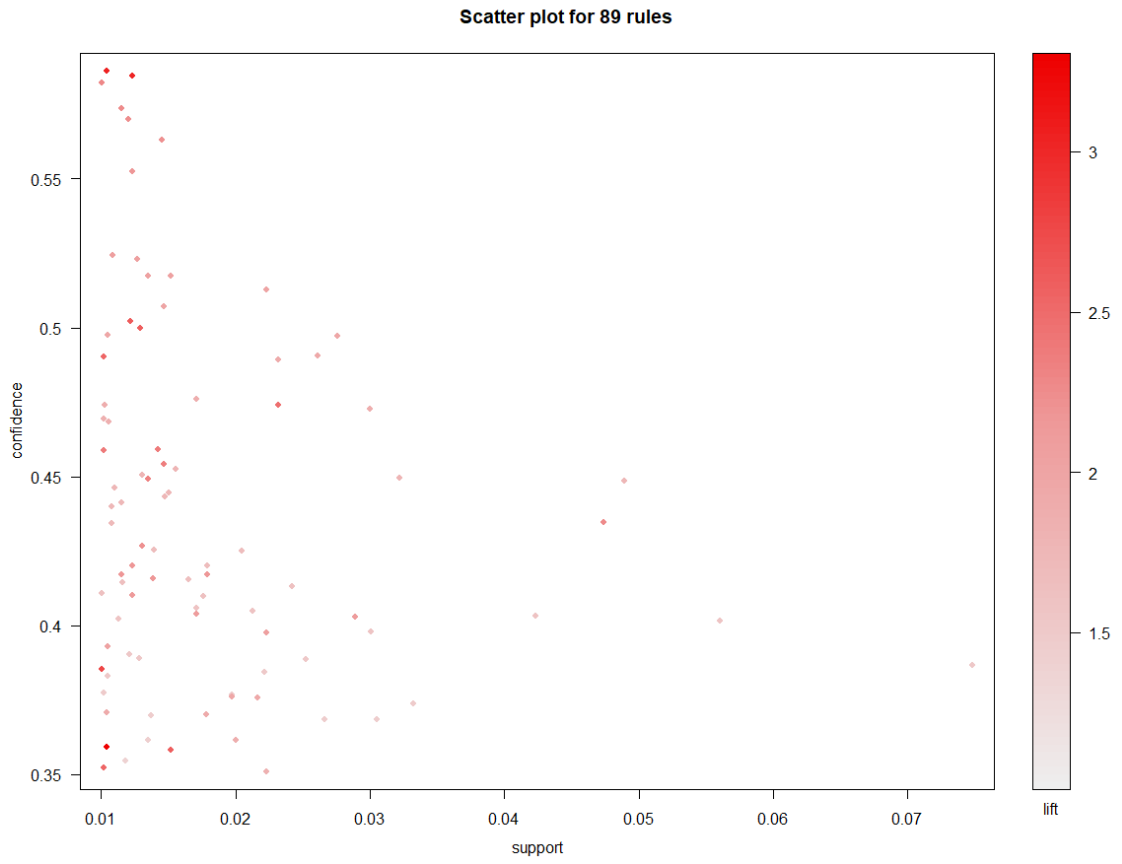
- Different formats are available (ex: scatterplot, matrix, graph) using the "method" option

✓ ploty_arules( ) function generates an interactive plot

- Users can adjust the axis, zoom in/out, etc.
- This function is now deprecated but still can be used

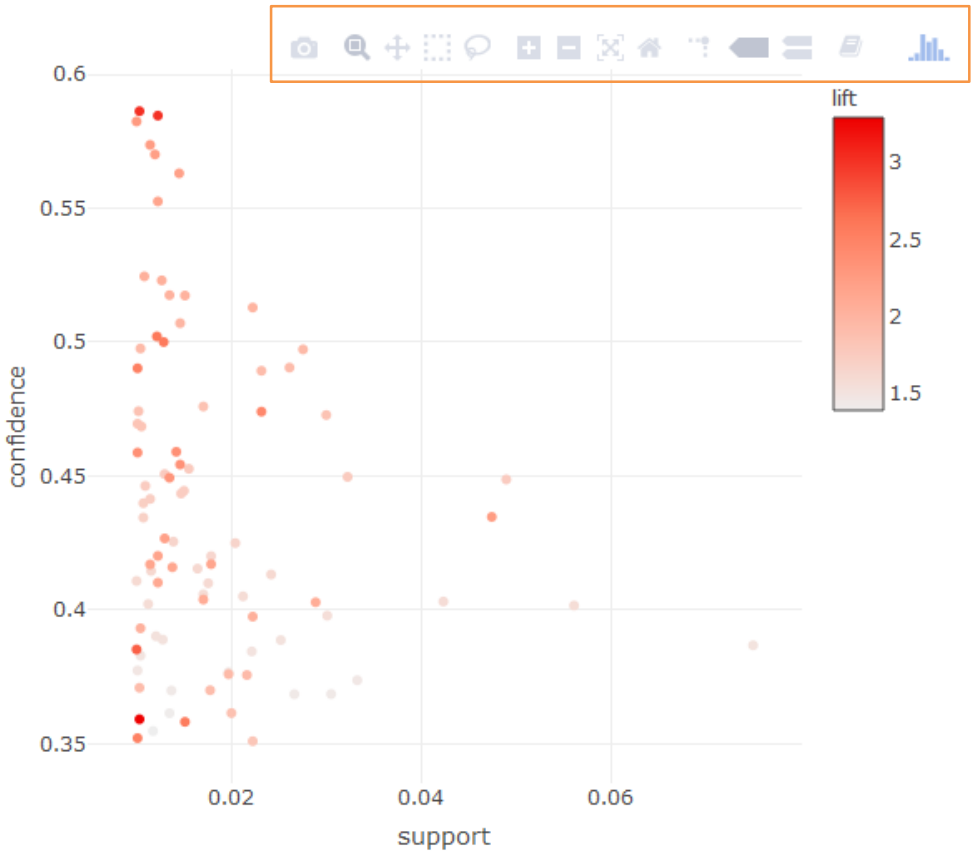
R Exercise: Market Basket Analysis

- Draw plots for the generated rules
  - ✓ plot( ) function (method = "scatterplot")
  - ✓ Used to understand the distribution of generated rules (not for interpreting individual rules)



R Exercise: Market Basket Analysis

- Draw plots for the generated rules
  - ✓ plotly_arules( ) function (method = "scatterplot")
  - ✓ You can adjusts some options

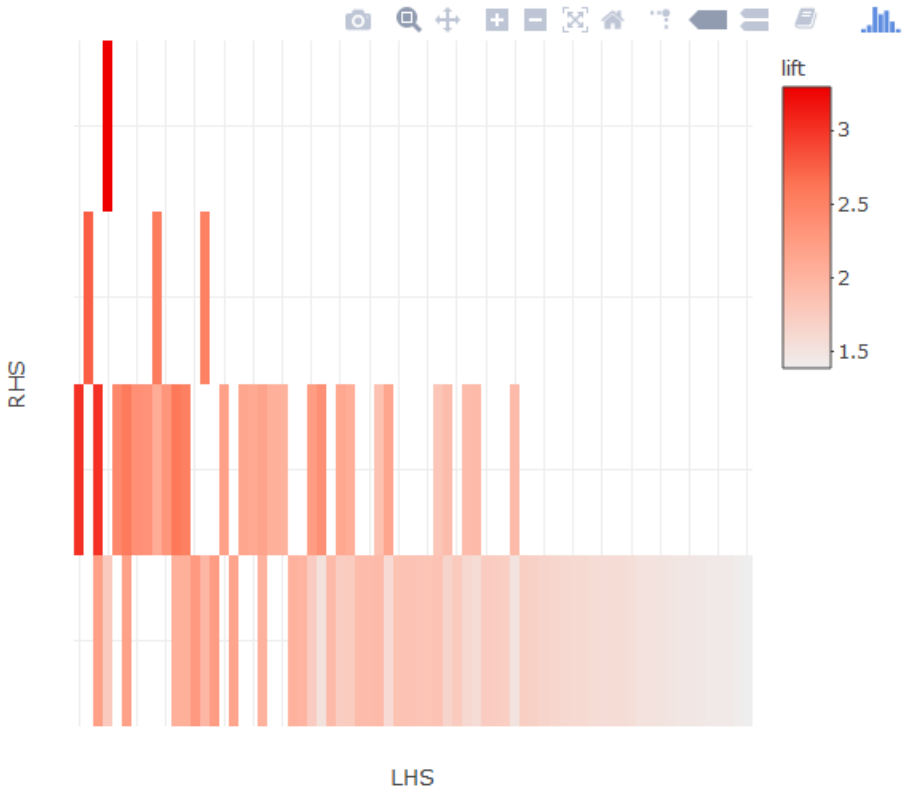


R Exercise: Market Basket Analysis

- Draw plots for the generated rules

```
plot(rules, method="matrix")
plotly_arules(rules, method = "matrix", measure = c("support", "confidence"),
          shading = "lift")
```

✓ method = "matrix"



R Exercise: Market Basket Analysis

- Change options to generate fewer rules

```
# Rule generation by Apriori with another parameters
rules <- apriori(Groceries, parameter=list(support=0.01, confidence=0.5))
plot(rules, method="graph")
plot(rules, method="paracoord")
```

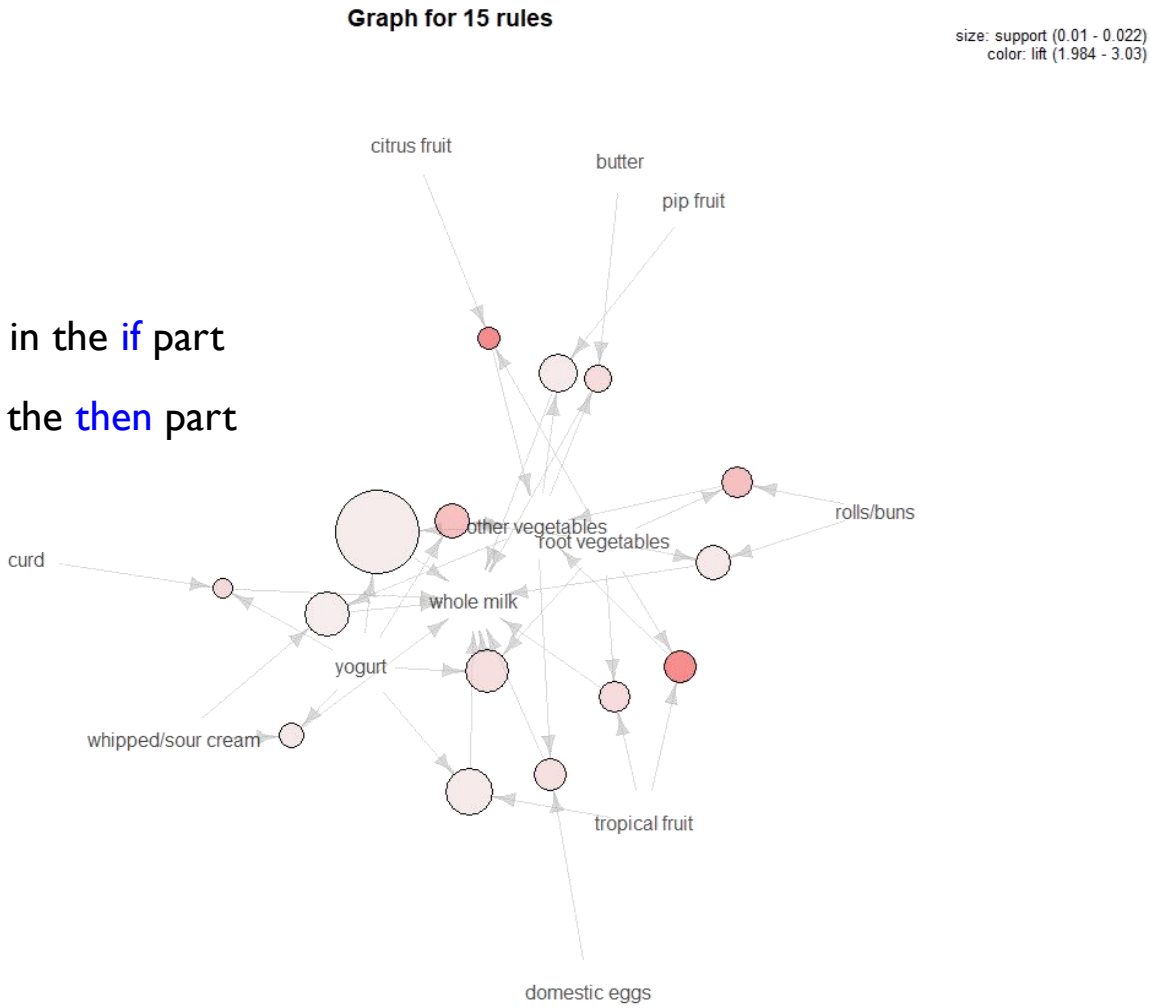
✓ Increase the confidence cut-off from 0.35 to 0.5

- 89 rules are reduced to 15 rules

✓ "graph" method and "paracoord" method can display the rules focusing on the relationship between the items in the generated rules

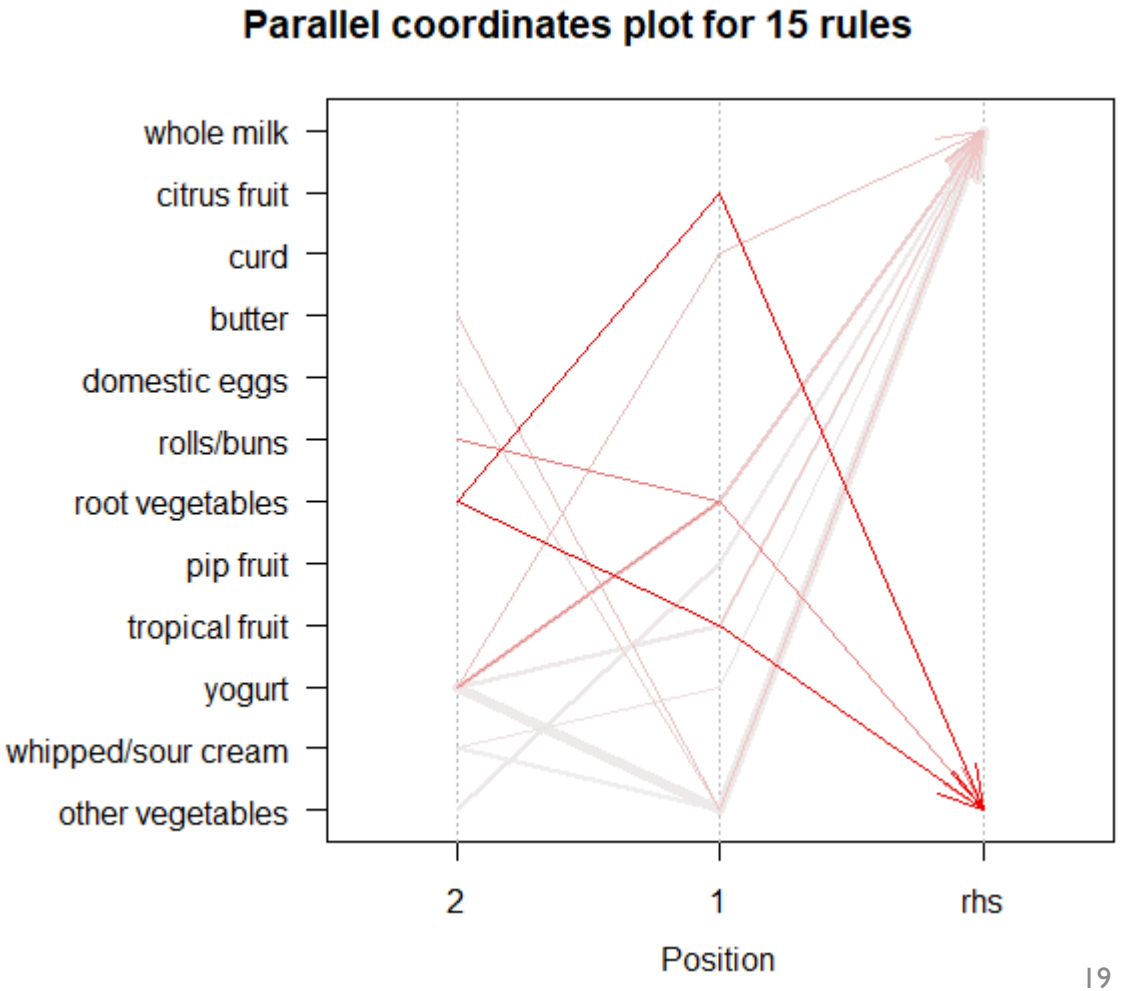
R Exercise: Market Basket Analysis

- Change options to generate fewer rules
  - ✓ "graph" method
    - Circle: rule
    - Circle size: support
    - Circle color: lift
    - Arrow from the circle: Item in the if part
    - Arrow to the circle: Item in the then part



R Exercise: Market Basket Analysis

- Change options to generate fewer rules
  - ✓ "paracoord" method
    - Line: rule
    - x-axis: item sequence
    - y-axsis: item name




ANY
questions?