# CSCI446/946 Big Data Analytics

## Week 5 Advanced Analytical Theory and Methods: Association Rules

School of Computing and Information Technology
University of Wollongong Australia

# Advanced Analytical Theory and Methods: Association Rules

- Overview of Association Rules
- Apriori Algorithm
- Evaluation of Candidate Rules
- An example of rule generation in R
- Validation and Testing
- Diagnostics

All the figures, tables and codes are from the book "Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data" unless indicated otherwise.

# Advanced Analytical Theory and Methods: Association Rules

- An unsupervised learning method
- Descriptive, not predictive
- Discover interesting, hidden relationship
  - Represented as rules or frequent itemsets
- Commonly used for mining transactions in databases

All the figures, tables and codes are from the book "Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data" unless indicated otherwise.

# Advanced Analytical Theory and Methods: Association Rules

- It can usually answer the questions like
  - Which products tend to be purchased together?
  - Of those customers who are similar to this person, what products do they tend to buy?
  - Of those customers who have purchased this product, what other products do they tend to view or purchase?

# Overview of Association Rules

- Each transaction consists of one or more items.
- What items are frequently purchased together?
- Goal: discover "interesting" relationships among the items.

# Overview of Association Rules

- Uncovered rule is in the form X → Y
  - meaning that when item X is observed, item Y is also observed
  - X: left-hand side (lhs); Y: right-hand side (rhs)
  - What does "Cereal → Milk (90%)" mean?
    - When cereal is purchased, 90% of the time milk is also purchased.

# Overview of Association Rules

- Also known as “market basket analysis”
  - Each transaction – shopping basket
- Itemset
  - A collection of items or individual entities that contain some kind of relationship
- k-itemset
  - An itemset containing k items
  - {item1, item2, …, item k}

# Overview of Association Rules

- Exhaustively check all possible itemsets?
  - No! The size is exponentially large…
- Apriori algorithm
  - One of the earliest and the most fundamental algorithms for generating association rules.
- Key concept: support
  - For pruning itemsets and controlling the exponential growth of candidate itemsets.

# Overview of Association Rules

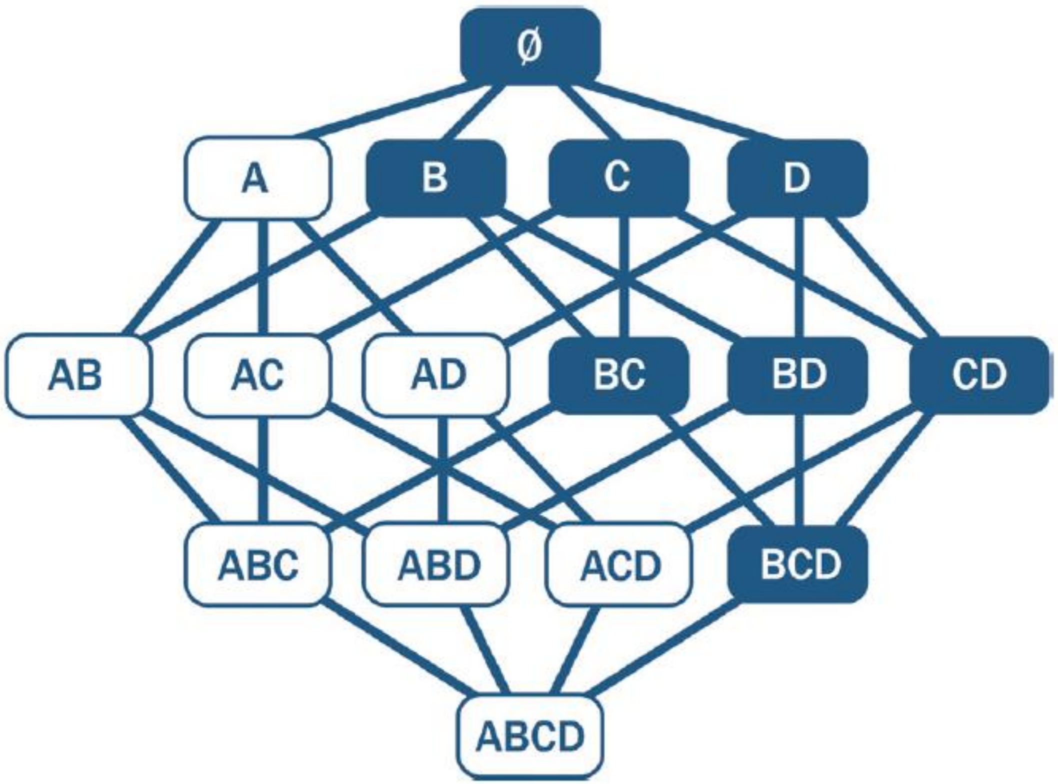
- Support
  - Given an item X, the support of X is the percentage of transactions that contain X
  - Denoted by support(X)
- Frequent itemset
  - Contains items that appear together often enough
  - Formally, its support >= a minimum support

# Overview of Association Rules

- Apriori property (downward closure property)
  - If an itemset is frequent, then any subset of this itemset must also be frequent
  - It provides the basis for the Apriori algorithm
- An example: If support({bread, jam}) = 0.6 ➔ Support({bread}}) >= 0.6 and Support({jam}}) >= 0.6
- Therefore, if X is infrequent then all supersets that contain X must also be infrequent.

# Overview of Association Rules

- Apriori property (downward closure property)




Itemset {A, B, C, D} and its subsets

# Apriori Algorithm

- It takes a bottom-up iterative approach to uncovering frequent itemsets
  - First, identify all frequent items (or 1-itemsets)
  - The identified frequent 1-itemsets are paired into 2-itemsets to identify frequent 2-itemsets
  - Grow the size of identified frequent itemsets and identify again
  - Repeat this process until 1) it runs out of support or 2) the itemsets reach a predefined length

# Apriori Algorithm

## Input

- A transaction database D
- A minimum support threshold δ
- An optional parameter N indicating the maximum length an itemset could reach

```
Apriori (D, δ, N)
   k ← 1
   L_k ← {1-itemsets that satisfy minimum support δ}
   while L_k ≠ ∅
      if ∄N ∨ (∃N ∧ k < N)
         C_{k+1} ← candidate itemsets generated from L_k
         for each transaction t in database D do
            increment the counts of C_{k+1} contained in t
         L_{k+1} ← candidates in C_{k+1} that satisfy minimum support δ
         k ← k+1
   return ⋃_k L_k
```

# Apriori Algorithm

- Output of the Apriori algorithm
  - The collection of all the frequent k-itemsets
- A collection of candidate rules is formed based on the frequent itemsets uncovered
  - {milk, eggs} may suggest candidate rules
    - {milk} → {eggs} and {eggs} → {milk}
- Implemented by `apriori()` function in R

```
itemsets <- apriori(Groceries, parameter=list(minlen=1, support=0.02,
                                               target="frequent itemsets"))
```

# Evaluation of Candidate Rules

- How to evaluate the appropriateness of these candidate rules?
  - Many measures!
  - Confidence, lift, leverage are among the most common.
- Confidence
  - The measure of certainty or trustworthiness associated with each rule

$$Confidence(X \rightarrow Y) = \frac{Support(X \wedge Y)}{Support(X)}$$

# Evaluation of Candidate Rules

- Minimum Confidence
  - A predefined threshold to indicate a relationship is "interesting".
  - A higher confidence could indicates that the rule (X→Y) is more interesting (be careful…).
  - All the rules can be ranked based on support or confidence

$$Confidence(X \rightarrow Y) = \frac{Support(X \wedge Y)}{Support(X)}$$

# Evaluation of Candidate Rules

- Issue with "Confidence"
  - In what cases will we obtain a high confidence?
  - It cannot tell
    - if a rule contains true implication of the relationship
    - If the rule is purely coincidental

$$Confidence(X \rightarrow Y) = \frac{Support(X \wedge Y)}{Support(X)}$$

# Evaluation of Candidate Rules

- Lift
  - Measures how many times more often X and Y occur together than expected if they are statistically independent of each other.
  - Measures how X and Y are really related rather than coincidentally happening together:

$$Lift(X \rightarrow Y) = \frac{Support(X \wedge Y)}{Support(X) * Support(Y)}$$

# Evaluation of Candidate Rules

- Lift
  - Lift is 1 if X and Y are statistically independent of each other.
  - A lift of X → Y greater than 1 indicates some usefulness of the rule.
  - A larger lift suggests a greater strength of the association between X and Y.

$$Lift\left(X \to Y\right) = \frac{Support(X \wedge Y)}{Support(X) * Support(Y)}$$

# Evaluation of Candidate Rules

- Leverage (Pitetsky-Shapiro's)
  - Measures the difference in the probability of X and Y appearing together compared to what would be expected if X and Y were statistically independent of each other

$$Lift(X \rightarrow Y) = \frac{Support(X \wedge Y)}{Support(X) * Support(Y)}$$

$$Leverage(X \rightarrow Y) = Support(X \wedge Y) - Support(X) * Support(Y)$$

# Evaluation of Candidate Rules

- Leverage
  - Its value will be zero when X and Y are statistically independent of each other.
  - If X and Y have some kind of relationship, the leverage would be greater than zero.

$$Lift(X \rightarrow Y) = \frac{Support(X \wedge Y)}{Support(X) * Support(Y)}$$

$$Leverage(X \rightarrow Y) = Support(X \wedge Y) - Support(X) * Support(Y)$$

# Evaluation of Candidate Rules

- Four measures
  - Support, Confidence, Lift, and Leverage
  - A high-confidence rule can sometimes be misleading.
  - Lift and leverage not only ensure interesting rules but also filter out coincidental rules.

$$Confidence(X \rightarrow Y) = \frac{Support(X \wedge Y)}{Support(X)} \qquad Lift(X \rightarrow Y) = \frac{Support(X \wedge Y)}{Support(X) * Support(Y)}$$

$$Leverage(X \rightarrow Y) = Support(X \wedge Y) - Support(X) * Support(Y)$$

# Evaluation of Candidate Rules

- Combination of Measures
  - Measures are often used in combination.
  - Example: Find all rules with a minimum level of confidence then, of those rules, sort rules in descending order by lift or leverage.
- Problem: These measures do not reflect novelty of rules i.e. differentiate between known rules and rules that are new to an observer.
  - Novelty and value of rules need to be evaluated by a human observer.

# Applications of Association Rules

- Market basket analysis
  - Better merchandising, Placement of products, and Promotion plan
- Recommender system
  - Discover related products or similar customers
- Clickstream analysis
  - Analyse data of web browsing and use clicks
- Much more…

# An Example:
# Transactions in a Grocery Store

- Employ Apriori algorithm to
  - Generate frequent itemsets and rules
  - Visualise the generated rules
- Use R and `arules` and `arulesViz` packages

```
install.packages('arules')
install.packages('arulesViz')

library('arules')
library('arulesViz')
```

# An Example: Transactions in a Grocery Store

- The Groceries dataset
  - 30 days of real-world sale transactions of a store

```
data(Groceries)
Groceries
transactions in sparse format with
 9835 transactions (rows) and
 169 items (columns)

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

# An Example: Transactions in a Grocery Store

- Class of "`transactions`" (in `arules` package)
  - `itemsetInfo`: a data frame with vectors of the same length as the number of transactions
    - Say, store Customer ID
  - `itemInfo`: A data frame to store item labels
  - `data`: A binary incidence matrix that indicates which item labels appear in every transaction

```
class(Groceries)
[1] "transactions"
attr(,"package")
[1] "arules"
```

# An Example: Transactions in a Grocery Store

- `itemInfo`: A data frame to store item labels

```
Example: List the first 10 item labels

Groceries@itemInfo[1:10, “labels”]
[1] "frankfurter" "sausage" "liver loaf" "ham“ "meat"
[6] "finished products" "organic sausage" "chicken"
"turkey" "pork"
```

# An Example: Transactions in a Grocery Store

- Example: list higher order descriptions

```
Groceries@itemInfo[1:20,]
               labels    level2            level1
1         frankfurter   sausage  meet and sausage
2             sausage   sausage  meet and sausage
3          liver loaf   sausage  meet and sausage
4                 ham   sausage  meet and sausage
5                meat   sausage  meet and sausage
6   finished products   sausage  meet and sausage
7     organic sausage   sausage  meet and sausage
8             chicken   poultry  meet and sausage
9              turkey   poultry  meet and sausage
10               pork      pork  meet and sausage
```

# An Example: Transactions in a Grocery Store

- data: A binary incidence matrix that indicates which item labels appear in every transaction
- Display the 10th to 20th transactions of the dataset
- Try: Groceries@data[1,] (returns sparse matrix)
- Better:

```
apply(Groceries@data[,10:20], 2,
      function(r) paste(Groceries@itemInfo[r,"labels"], collapse=", ")
      )
```

```
 [1] "whole milk, cereals"
 [2] "tropical fruit, other vegetables, white bread, bottled water,
chocolate"
 [3] "citrus fruit, tropical fruit, whole milk, butter, curd, yogurt,
flour, bottled water, dishes"
 [4] "beef"
```

# An Example: Transactions in a Grocery Store

- Frequent Itemset Generation
  - Use `apriori()` function in the `arules` package
  - The `apriori()` function executes all the steps (What are they?) once
  - Specific the minimum support threshold
  - Until it runs out of support or until k (in *k*-itemset) reaches the default `maxlen = 10`

```
itemsets <- apriori(Groceries, parameter=list(minlen=1, support=0.02,
                                              target="frequent itemsets"))
```

# An Example: Transactions in a Grocery Store

```
itemsets <- apriori(Groceries, parameter=list(minlen=1, support=0.02,
                                           target="frequent itemsets"))
parameter specification:
 confidence minval smax arem  aval originalSupport support minlen
        0.8    0.1    1 none FALSE            TRUE    0.02      1
 maxlen            target   ext
     10 frequent itemsets FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [59 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [122 set(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

# An Example: Transactions in a Grocery Store

- **Rule** Generation and Visualization
  - Use `apriori()` function in the `arules` package

```
rules <- apriori(Groceries, parameter=list(support=0.001,
                              confidence=0.6, target = "rules"))
```

# An Example: Transactions in a Grocery Store

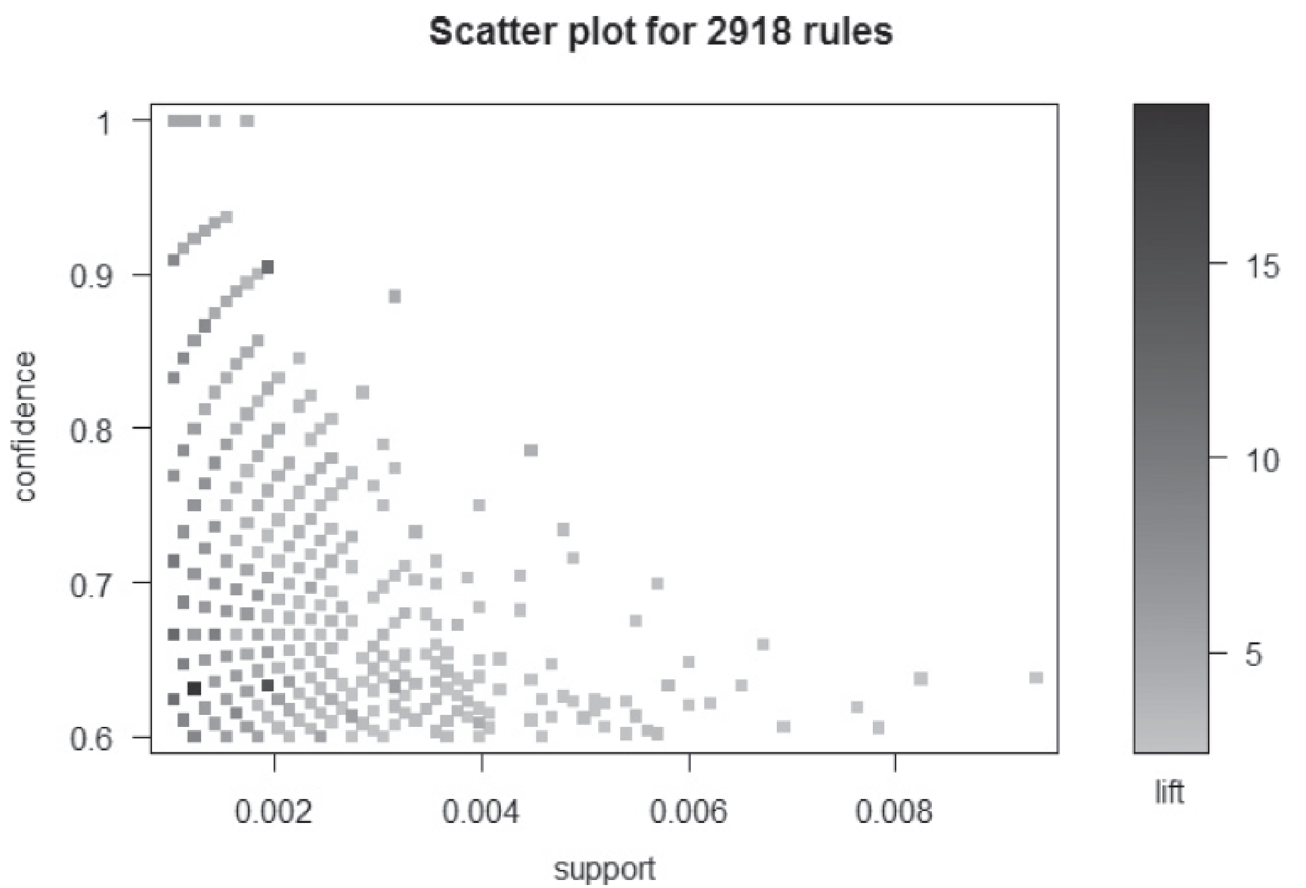
```
rules <- apriori(Groceries, parameter=list(support=0.001,
                        confidence=0.6, target = "rules"))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen
        0.6    0.1    1 none FALSE            TRUE   0.001      1
 maxlen target   ext
     10  rules FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
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 ... [2918 rule(s)] done [0.00s].
creating S4 object  ... done [0.01s].
```

# An Example: Transactions in a Grocery Store

```
summary(rules)
set of 2918 rules

rule length distribution (lhs + rhs):sizes
   2    3    4    5    6 
   3  490 1765  626   34 

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  2.000   4.000   4.000   4.068   4.000   6.000 

summary of quality measures:
    support           confidence          lift       
 Min.   :0.001017   Min.   :0.6000   Min.   : 2.348  
 1st Qu.:0.001118   1st Qu.:0.6316   1st Qu.: 2.668  
 Median :0.001220   Median :0.6818   Median : 3.168  
 Mean   :0.001480   Mean   :0.7028   Mean   : 3.450  
 3rd Qu.:0.001525   3rd Qu.:0.7500   3rd Qu.: 3.692  
 Max.   :0.009354   Max.   :1.0000   Max.   :18.996  

mining info:
      data ntransactions support confidence
 Groceries          9835   0.001        0.6
```

# An Example: Transactions in a Grocery Store

- Visualization: `plot(rules)` function




*Scatterplot of the 2,918 rules with minimum support 0.001 and minimum confidence 0.6*

# An Example: Transactions in a Grocery Store

- Visualization: `plot(rules@quality)`

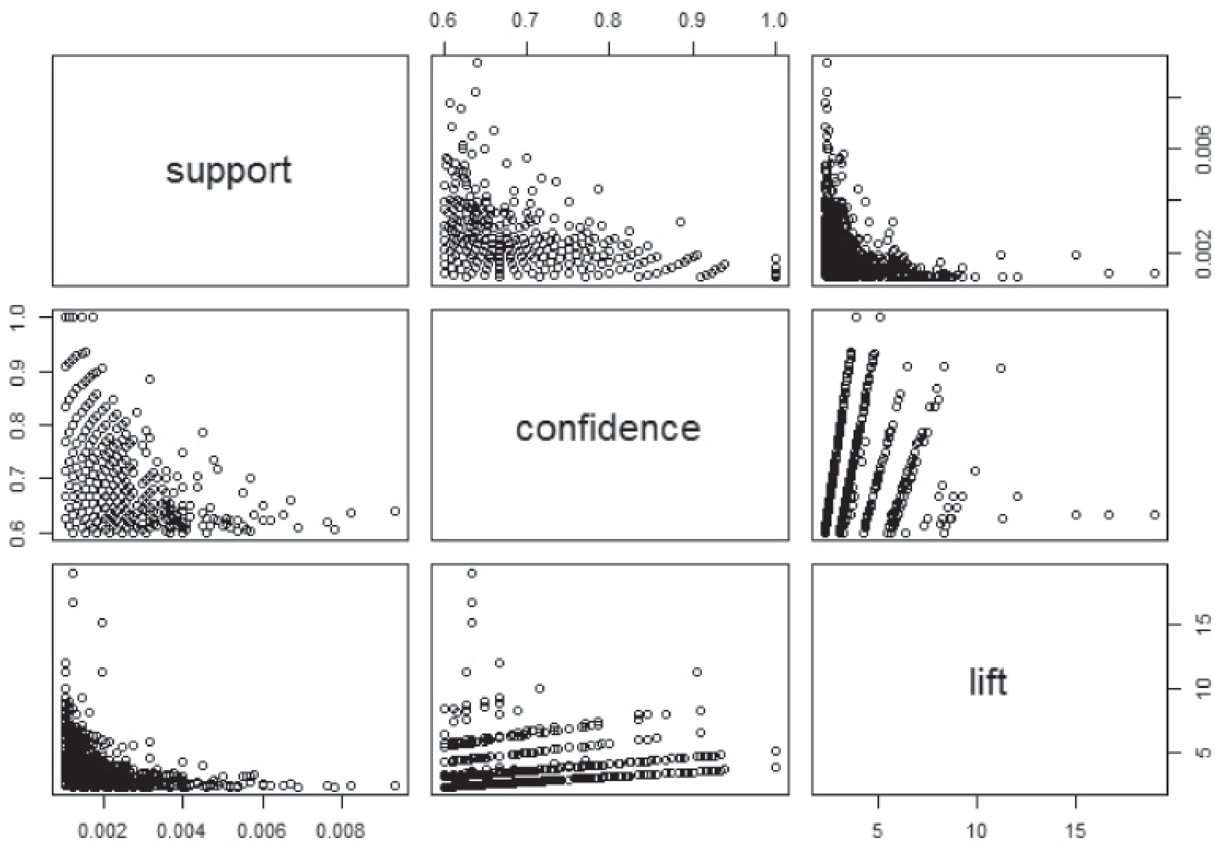
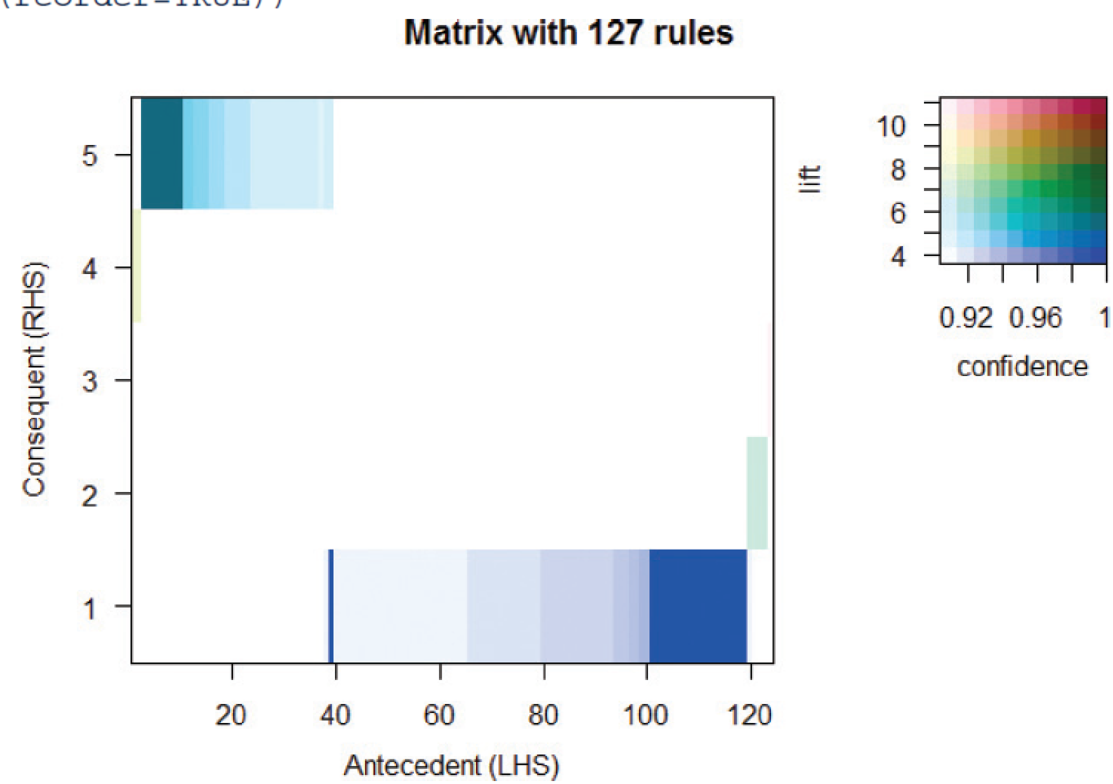



FIGURE 5-4 *Scatterplot matrix on the support, confidence, and lift of the 2,918 rules*

# An Example: Transactions in a Grocery Store

- Visualization: `plot()`

```
confidentRules <- rules[quality(rules)$confidence > 0.9]
plot(confidentRules, method="matrix", measure=c("lift", "confidence"),
     control=list(reorder=TRUE))
```

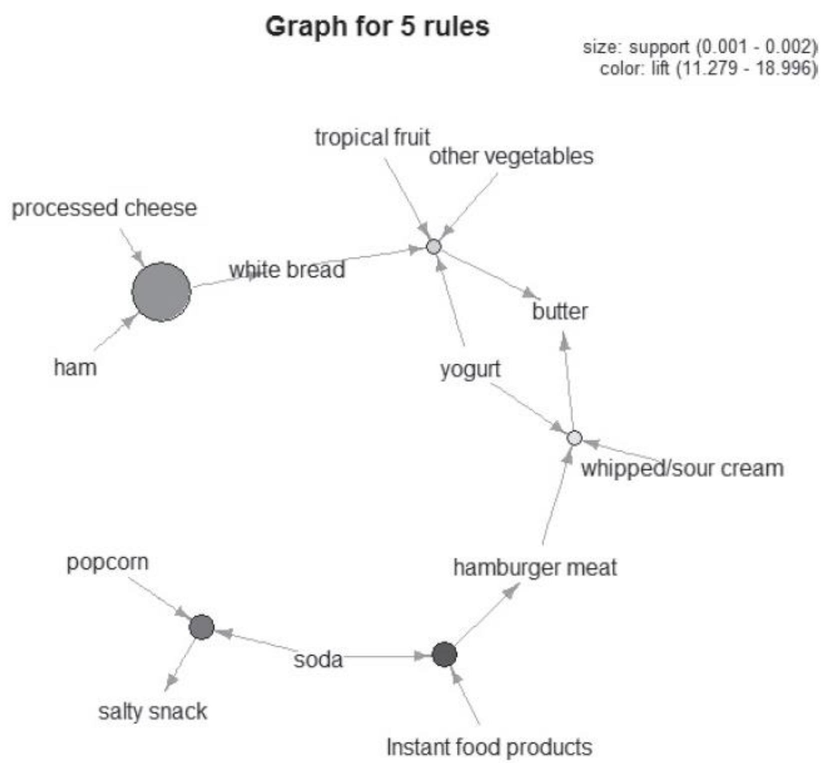



*Matrix-based visualization of LHS and RHS, colored by lift and confidence*

# An Example:
# Transactions in a Grocery Store

- Visualization: `plot()`

```
highLiftRules <- head(sort(rules, by="lift"), 5)
plot(highLiftRules, method="graph", control=list(type="items"))
```




*Graph visualization of the top five rules sorted by lift*

# An Example: Transactions in a Grocery Store

- Display rule content: `inspect()`

```
inspect(head(sort(rules, by="lift"), 10))
   lhs                                      rhs
support   confidence   lift
1   {Instant food products,
     soda}                            => {hamburger meat}
0.001220132  0.6315789 18.995654
2   {soda,
     popcorn}                         => {salty snack}
0.001220132  0.6315789 16.697793
3   {ham,
     processed cheese}                => {white bread}
0.001931876  0.6333333 15.045491
```

# Validation and Testing

- Uninteresting rules
  - Involve mutually independent items
  - Cover few transactions
- Some rules could be purely coincidental
  - If 95% of customers buy X and 90% of them buy Y, then X and Y would occur together at least 85% of the time, even if there is no relationship between them
- Subjective criteria
  - Rules don't reveal unexpected profitable actions

# Diagnostics

- Measures like confidence, lift, and leverage shall be used along with human insights
- Properly specify the minimum support
- Apriori algorithm can be computationally expensive!
  - Various methods to improve Apriori's efficiency

# Recap: Advanced Analytical Theory and Methods: Association Rules

- Apriori Algorithm
  - Unsupervised analysis technique
  - Uncovers relationships among items
- A wide range of applications
- Several measures to help validation
- Interesting rules
  - Do not seem obvious
  - Provide valuable insights

# Q&A


Images Courtesy of Google Image