

Statistical Computing with R: Masters in Data Sciences 503 (S28) Fourth Batch, SMS, TU, 2025

Shital Bhandary

Associate Professor

Statistics/Bio-statistics, Demography and Public Health Informatics

Patan Academy of Health Sciences, Lalitpur, Nepal

Faculty, Masters in Medical Research, NHRC/Kathmandu University

Faculty, FAIMER Fellowship in Health Professions Education, India/USA

Review Preview: Unsupervised models

- Association rules learning
 - Market-Basket analysis
- Monte Carlo simulations
 - Good old days!
- Class imbalance problem
 - Statistical approach
 - Data science approach

Association rules learning/mining:

<https://towardsdatascience.com/association-rule-mining-in-r-ddf2d044ae50>

- Association Rule Mining (also called as Association Rule Learning) is a common technique used **to find associations (co-occurrence) between many variables.**
- It is often used by grocery stores, e-commerce websites, and anyone with large **transactional** databases.
- A most common example that we encounter in our daily lives — **Amazon knows what else you want to buy when you order something on their site.**
- The same idea extends to **Spotify too — They know what song you want to listen to next.**
- All of these incorporate, at some level, data mining concepts and association rule mining algorithms.

Association rules: example problem

<https://www.datacamp.com/community/tutorials/market-basket-analysis-r>

- You get a client who runs a retail store and gives you **data for all transactions that consists of items bought in the store by several customers over a period of time.**
- Your client then **asks you to use that data to help boost their business.**
- Your client will **use your findings to not only change/update/add items in inventory but also use them to change the layout of the physical store or rather an online store.**
- To find results that will help your client, you will use **Market Basket Analysis (MBA)** which uses **Association Rule Mining** on the given transaction data.

Use of association rules mining result:

<https://www.datacamp.com/community/tutorials/market-basket-analysis-r>

- Changing the store layout according to trends
- Customer behavior analysis
- Catalogue design
- Cross marketing on online stores
- What are the trending items customers buy
- Customized emails with add-on sales
- etc.

Association rule mining: If => Then analysis

<https://www.datacamp.com/community/tutorials/market-basket-analysis-r>

- Association Rule Mining is used when you want to **find an association between different objects** in a set, **find frequent patterns in a transaction database**, relational databases or any other information repository.
- The applications of Association Rule Mining are found in Marketing, Basket Data Analysis (or Market Basket Analysis) in retailing, clustering and classification.
- It can tell you what items do customers frequently buy together by generating a set of rules called **Association Rules**.
- In simple words, it gives you output as rules in form **if this then that**.

What is **apriori** algorithm and rule?

<http://r-statistics.co/Association-Mining-With-R.html>

- Association mining is usually done on transactions data from a retail market or from an online e-commerce store.
- Since most transactions data is **large**, the apriori algorithm makes it **easier to find these patterns or rules quickly**.
- A rule is a notation that represents which item/s is frequently bought with what item/s.
- It has an **LHS** and an **RHS** part and can be represented as follows:
$$\text{itemsetA} \Rightarrow \text{itemsetB}$$
- This means, the item/s on the **right** were frequently purchased along with items on the **left**.

How to measure the strength of a rule?

<http://r-statistics.co/Association-Mining-With-R.html>

- The **apriori algorithm** generates the most relevant set of rules from a given transaction data.
- **It also shows the support, confidence and lift of those rules.**
- These three measures can be used to decide the relative strength of the rules.
- How are they computed?

Lets consider the rule $A \Rightarrow B$ in order to compute these metrics.

Support=Number of transactions with both A and B/Total number of transactions
 $=P(A \cap B) = \text{frequency}(A,B)/N$

Confidence=Number of transactions with both A and B/Total number of transactions with A
 $=P(A \cap B)/P(A) = \text{frequency}(A,B)/\text{frequency}(A)$

ExpectedConfidence=Number of transactions with B/Total number of transactions
 $=P(B)=\text{frequency}(B)/N$

Lift=Confidence/Expected Confidence
 $=P(A \cap B)/P(A).P(B) = \text{Support}(A,B)/\text{Support}(A).\text{Support}(B)$

Association rule: Support and confidence

- Association rules are given in the form as below:

A=>B[Support,Confidence]

- The part **before =>** is referred to as **if (Antecedent)** and the part **after =>** is referred to as **then (Consequent)**.
- Where A and B are sets of items in the transaction data. A and B are disjoint sets.

Computer=>Anti-virusSoftware
[Support=20%,confidence=60%]

Above rule says:

- 20% transaction show Anti-virus software is bought with purchase of a Computer (**support**)
- 60% of customers who purchase Anti-virus software is bought with purchase of a Computer (**confidence**)

Lift:

- *Lift* is the factor by which, the co-occurrence of A and B exceeds the expected probability of A and B co-occurring, **had they been independent**.
- So, higher the lift, higher the chance of A and B occurring together.
- **lift = 1**: implies no association between items.
- **lift > 1**: greater than 1 means that item B is likely to be bought if item A is bought,
- **lift < 1**: less than 1 means that item B is unlikely to be bought if item A is bought.

Note:

- **Frequent Itemsets:**

Item-sets whose support is greater or equal than minimum support threshold (min_sup).

- **min_sup is set on user choice.**

- **Strong rules:**

If a rule $A \Rightarrow B[\text{Support, Confidence}]$ satisfies **min_sup** and **min_confidence** then it is a strong rule.

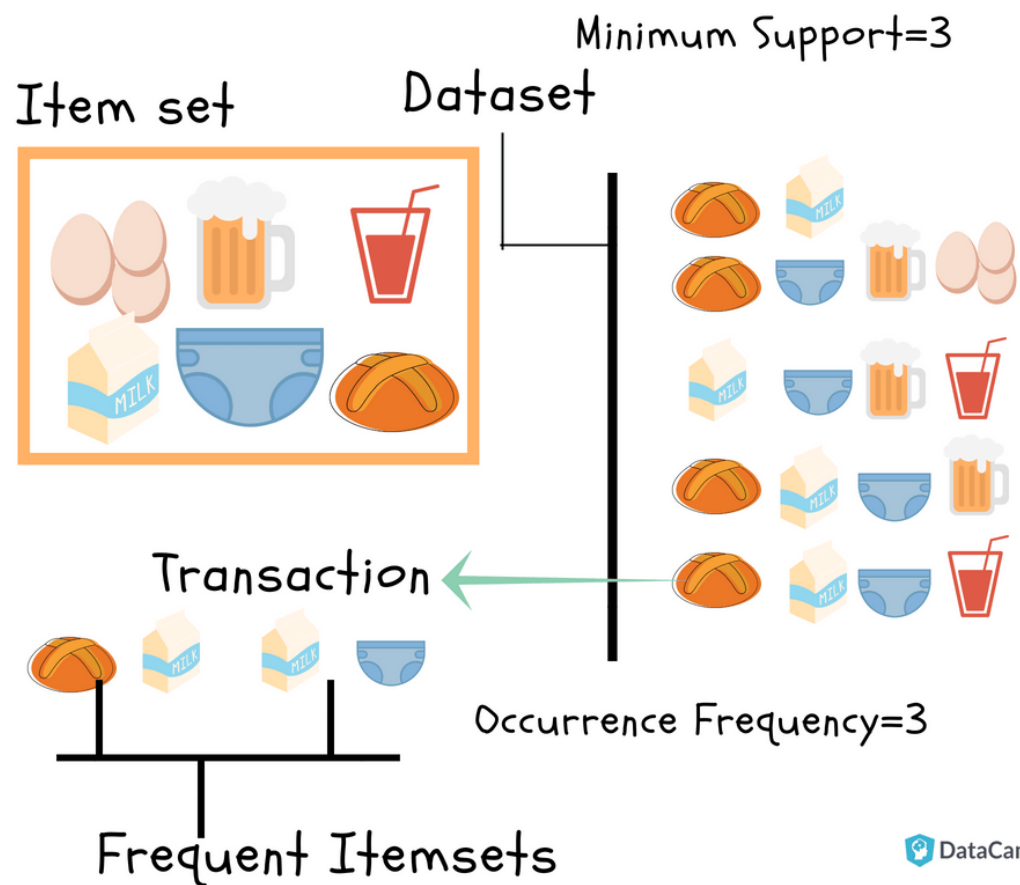
- **Coverage:**

Coverage (also called cover or LHS-support) is the support of the left-hand-side of the rule, i.e., $\text{supp}(X)$.

It represents a measure of “to how often the rule can be applied”.

Example:

<https://www.datacamp.com/community/tutorials/market-basket-analysis-r>



ID	Items
1	{Bread, Milk}
2	{Bread, Diapers, Beer, Eggs}
3	{Milk, Diapers, Beer, Cola}
4	{Bread, Milk, Diapers, Beer}
5	{Bread, Milk, Diapers, Cola}
...	...

market
basket
transactions

{Diapers, Beer}

Example of a frequent itemset

{Diapers} → {Beer}

Example of an association rule

Calculate the following for {Bread => Milk}:

- Support for (Bread)
 - Support for (Milk)
 - Support for (Bread, Milk)
 - Confidence (Bread => Milk)
 - ExpectedConfidence(Bread=>Milk)
 - Lift (Bread => Milk)
 - Coverage(Bread=>Milk) = support(lhs)
- Support for (Bread)= $4/5 = f(B)/N=0.8$
 - Support for (Milk)= $4/5=f(M)/N=0.8$
 - Support(B,M) = $f(B,M)/N=3/5=0.6$
 - Confidence (Bread => Milk) = $3/4=0.75$
 - ExpectedConfidence= $P(M)=4/5=0.8$
 - Lift (Bread => Milk)
= $\text{Confidence}/\text{ExpectedConfidence}=0.75/0.80$
= 0.9375
 - OR
= $\text{support}(A,B)/\text{support}(A).\text{support}(B)$
= $(0.6)/[(0.8).(0.8)] = 0.6/0.64 = 0.9375$

Let's do it in R!

create a list of baskets

```
market_basket <-  
list(  
  c("bread", "milk"),  
  c("bread", "diapers", "beer", "Eggs"),  
  c("milk", "diapers", "beer", "cola"),  
  c("bread", "milk", "diapers", "beer"),  
  c("bread", "milk", "diapers", "cola")  
)
```

set transaction names (T1 to T5)

```
names(market_basket) <- paste("T", c(1:5), sep  
= "")
```

> # create a list of baskets

```
> market_basket <-  
+ list(  
+   c("bread", "milk"),  
+   c("bread", "diapers", "beer", "Eggs"),  
+   c("milk", "diapers", "beer", "cola"),  
+   c("bread", "milk", "diapers", "beer"),  
+   c("bread", "milk", "diapers", "cola")  
+ )
```

>

> # set transaction names (T1 to T5)

```
> names(market_basket) <- paste("T", c(1:5),  
sep = "")
```

Let's use "arules" package and get some outputs:

- library(arules)
- #Transformation
- trans <- as(market_basket, "transactions")
- #Dimensions
- dim(trans)
- #Item labels
- itemLabels(trans)
- #Summary
- summary(trans)
- #Plot
- image(trans)

#Transformation to transactions data

```
trans <- as(market_basket, "transactions")
```

dim(trans)

- [1] 5 6 #5 transactions, 6 items

#Item labels

```
> itemLables(trans)
```

```
[1] "beer"  "bread" "cola"  "diapers"  
"Eggs"  "milk"
```

Let's use “arules” package and get some outputs:

#Summary

- summary(trans)

transactions as itemMatrix in sparse format with
5 rows (elements/itemsets/transactions) and
6 columns (items) and a **density of 0.6 (non-zero cells)**

most frequent items:

bread	diapers	milk	beer	cola	(Other)
4	4	4	3	2	1

element (itemset/transaction) length distribution:

sizes

2	4	(Itemset)
1	4	(transactions)

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
2.0	4.0	4.0	3.6	4.0	4.0

Let's inspect the “trans”

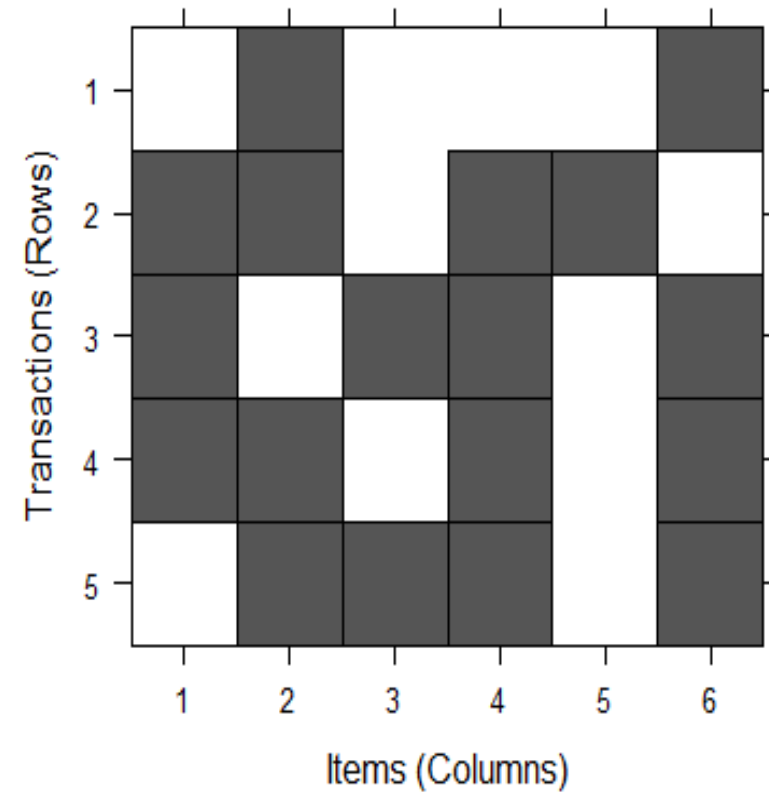
- inspect(trans)

•	items	transactionID
•	[1] {bread, milk}	T1
•	[2] {beer, bread, diapers, Eggs}	T2
•	[3] {beer, cola, diapers, milk}	T3
•	[4] {beer, bread, diapers, milk}	T4
•	[5] {bread, cola, diapers, milk}	T5

Plot of “trans”

#Plot

- image(trans)



Apriori algorithm: why?

- **Frequent Itemset Generation** is the most computationally expensive step because it requires a full database scan.
- In above example, we have seen the example of only 5 transactions, but in **real-world transaction data for retail can exceed up to GB s and TBs of data** for which an **optimized algorithm is needed to prune out Item-sets** that will not help in later steps.
- **For this APRIORI Algorithm is used to create new rules.**
- Since Support and Confidence measure how interesting the rule is, we will use them to create rules.
- **New rule is set by the minimum support and minimum confidence thresholds.**
- The closer to threshold the more the rule is of use to the client.
- These thresholds set by client help to compare the rule strength according to **your own or client's will.**

Apriori algorithm in “trans” with minimum support of 0.3 and min. confidence of 0.5:

#Min Support 0.3, confidence as 0.5.

```
rules <- apriori(trans,  
parameter = list(supp=0.3, conf=0.5,  
                  maxlen=10,  
                  target= "rules"))
```

Note: maxlen = maximum length of the transaction! We could have used maxlen = 4 here as we know it but this will not be known in real-life!

Apriori

- Parameter specification:

confidence	minval	smax	arem	aval	originalSupport	maxtime	support	minlen	maxlen
0.5	0.1	1	none	FALSE	TRUE	5	0.3	1	10

```
target ext  
rules TRUE
```

Algorithmic control:

```
filter tree heap memopt load sort verbose  
0.1 TRUE TRUE FALSE TRUE 2 TRUE
```

Summary of the “rules”:

summary(rules)

#summary of quality measures:

support confidence coverage lift

Min. : 0.4000	Min. : 0.5000	Min. : 0.4000	Min. : 0.8333
1st Qu.:0.4000	1st Qu.:0.6667	1st Qu.:0.6000	1st Qu.:0.8333
Median :0.4000	Median :0.7500	Median :0.6000	Median :1.0000
Mean :0.4938	Mean :0.7474	Mean :0.6813	Mean :1.0473
3rd Qu.:0.6000	3rd Qu.:0.8000	3rd Qu.:0.8000	3rd Qu.:1.2500
Max. : 0.8000	Max. : 1.0000	Max. : 1.0000	Max. : 1.6667

mining info:

- data ntransactions support confidence
- trans 5 0.3 0.5

Inspection of the “rules” with minlen:

Inspect (rules)

#Output from R:

	lhs	rhs	support	confidence	coverage	lift	count
• [1]	{}	=> {beer}	0.6	0.6000000	1.0	1.0000000	3
• [2]	{}	=> {milk}	0.8	0.8000000	1.0	1.0000000	4
• [3]	{}	=> {bread}	0.8	0.8000000	1.0	1.0000000	4
• [4]	{}	=> {diapers}	0.8	0.8000000	1.0	1.0000000	4
• [5]	{cola}	=> {milk}	0.4	1.0000000	0.4	1.2500000	2
• [6]	{milk}	=> {cola}	0.4	0.5000000	0.8	1.2500000	2
• [7]	{cola}	=> {diapers}	0.4	1.0000000	0.4	1.2500000	2
• [8]	{diapers}	=> {cola}	0.4	0.5000000	0.8	1.2500000	2
• [9]	{beer}	=> {milk}	0.4	0.6666667	0.6	0.8333333	2
• [10]	{milk}	=> {beer}	0.4	0.5000000	0.8	0.8333333	2
• [11]	{beer}	=> {bread}	0.4	0.6666667	0.6	0.8333333	2
• [12]	{bread}	=> {beer}	0.4	0.5000000	0.8	0.8333333	2
• [13]	{beer}	=> {diapers}	0.6	1.0000000	0.6	1.2500000	3
• [14]	{diapers}	=> {beer}	0.6	0.7500000	0.8	1.2500000	3
• [15]	{milk}	=> {bread}	0.6	0.7500000	0.8	0.9375000	3
• [16]	{bread}	=> {milk}	0.6	0.7500000	0.8	0.9375000	3
•							
• [32]							

We can remove the “empty” rules

```
rules <- apriori(trans,  
  parameter = list(supp=0.3,  
    conf=0.5,  
      maxlen=10,  
      minlen=2,  
      target= "rules"))
```

- **set of 28 rules**
- rule length distribution (lhs + rhs):
 sizes
- 2 3
- 16 12
-

	lhs	rhs	support	confidence	coverage	lift	count
• [1]	{cola}	=> {milk}	0.4	1.0000000	0.4	1.2500000	2
• [2]	{milk}	=> {cola}	0.4	0.5000000	0.8	1.2500000	2
• [3]	{cola}	=> {diapers}	0.4	1.0000000	0.4	1.2500000	2
•	...						
• [17]	{cola, milk}	=> {diapers}	0.4	1.0000000	0.4	1.2500000	2
• [18]	{cola, diapers}	=> {milk}	0.4	1.0000000	0.4	1.2500000	2
• [19]	{diapers, milk}	=> {cola}	0.4	0.6666667	0.6	1.6666667	2

Let's set RHS rule for "trans" data:

#For example, to analyze what items customers buy before buying {beer},

#we set **rhs=beer** and default=lhs:

```
beer_rules_rhs <- apriori(trans,  
                           parameter =  
list(supp=0.3, conf=0.5,  
      maxlen=10,  
      minlen=2),
```

```
appearance = list(default="lhs",  
rhs="beer"))
```

#Inspect

- inspect(beer_rules_rhs)

	lhs	rhs	support	confidence	coverage	lift	count
• [1]	{bread}	=> {beer}	0.4	0.5000000	0.8	0.8333333	2
• [2]	{milk}	=> {beer}	0.4	0.5000000	0.8	0.8333333	2
• [3]	{diapers}	=> {beer}	0.6	0.7500000	0.8	1.2500000	3
• [4]	{bread, diapers}	=> {beer}	0.4	0.6666667	0.6	1.1111111	2
• [5]	{diapers, milk}	=> {beer}	0.4	0.6666667	0.6	1.1111111	2

Let's set LHS rule for “trans” data:

#For example, to analyze what items customers buy before buying {beer},

#we set **lhs=beer** and default=rhs:

```
beer_rules_lhs <- apriori(trans,  
                           parameter =  
list(supp=0.3, conf=0.5,  
      maxlen=10,  
      minlen=2),
```

```
      appearance =  
list(lhs="beer", default="rhs"))
```

#Inspect the result:

```
inspect(beer_rules_lhs)
```

	lhs	rhs	support	confidence	coverage	lift	count
[1]	{beer}	=> {bread}	0.4	0.6666667	0.6	0.8333333	2
[2]	{beer}	=> {milk}	0.4	0.6666667	0.6	0.8333333	2
[3]	{beer}	=> {diapers}	0.6	1.0000000	0.6	1.2500000	3

Product recommendation rule:

#Product recommendation rule

- `rules_conf <- sort (rules,
by="confidence",
decreasing=TRUE)`

#inspect the rule

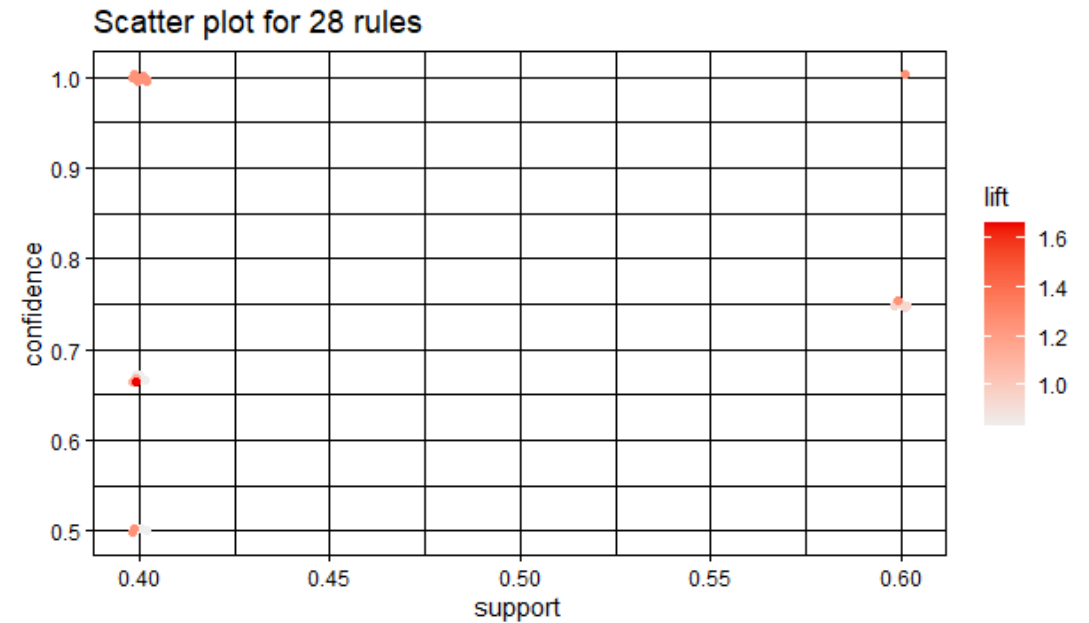
show the support, lift and
confidence for all rules

- `inspect(head(rules_conf))`

	lhs	rhs	support	confidence	coverage	lift	n
• [1]	{cola}	=> {milk}	0.4	1	0.4	1.25	2
• [2]	{cola}	=> {diapers}	0.4	1	0.4	1.25	2
• [3]	{beer}	=> {diapers}	0.6	1	0.6	1.25	3
• [4]	{cola, milk}	=> {diapers}	0.4	1	0.4	1.25	2
• [5]	{cola, diapers}	=> {milk}	0.4	1	0.4	1.25	2
• [6]	{beer, milk}	=> {diapers}	0.4	1	0.4	1.25	2

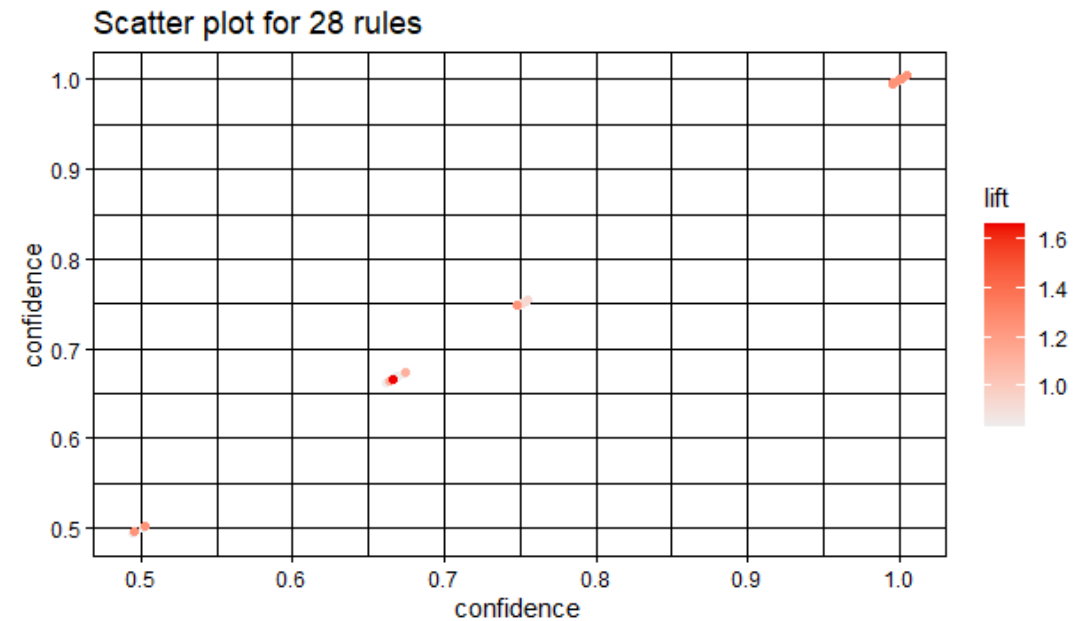
Plotting rules with “arulesViz” package:

- `library(arulesViz)`
- `plot(rules)`



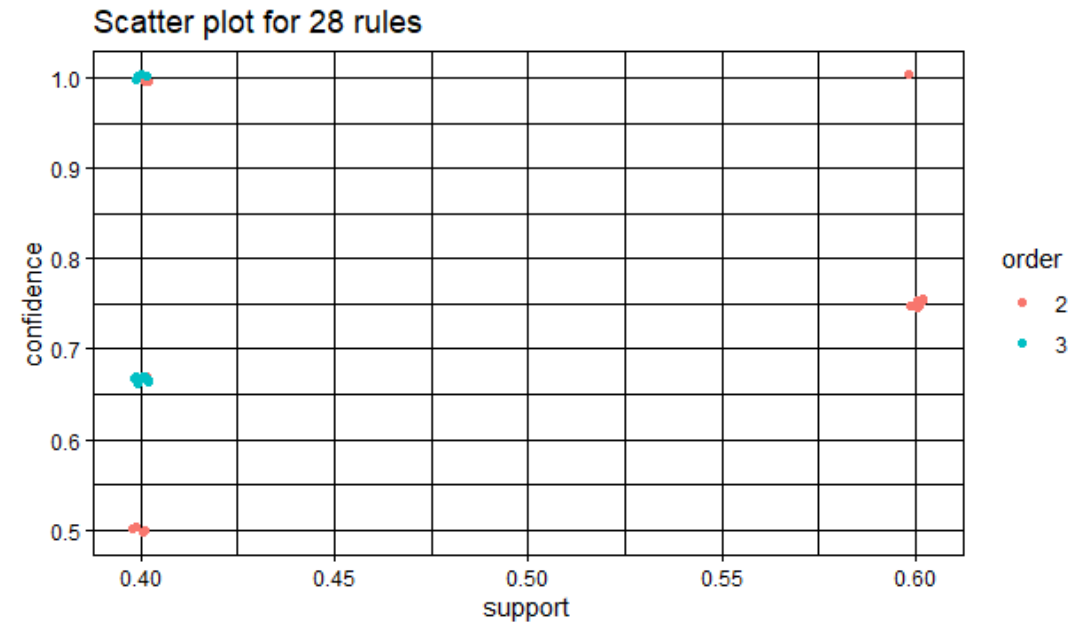
Plotting rules with “arulesViz” package:

- `plot(rules, measure = "confidence")`



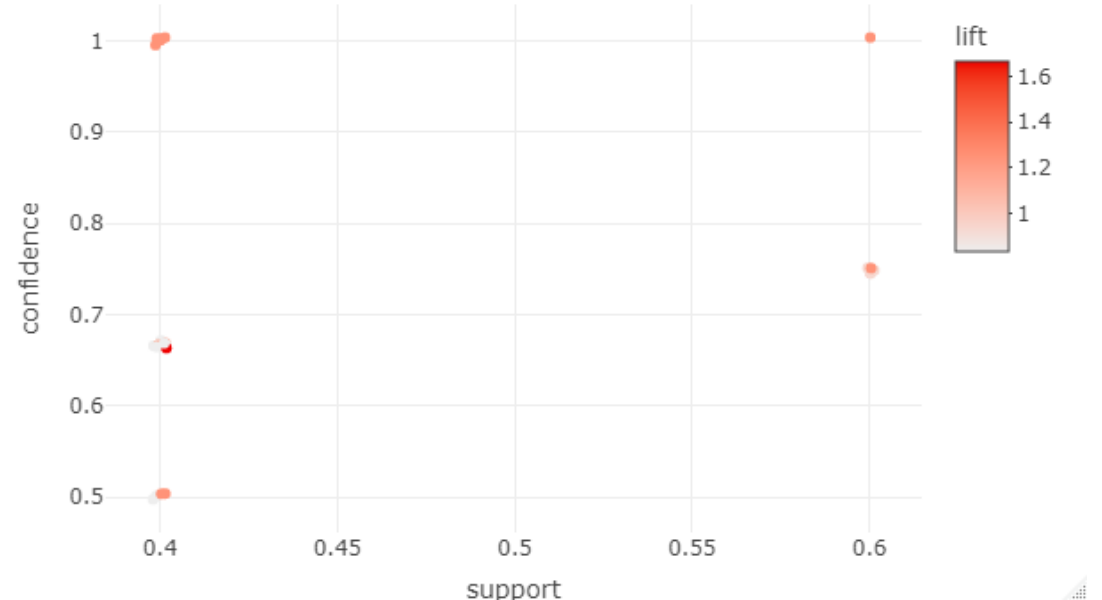
Plotting rules with “arulesViz” package:

- `plot(rules, method = "two-key plot")`



Interactive plot with “plotly” engine:

- #Interactive plot
- `plot(rules, engine = "plotly")`

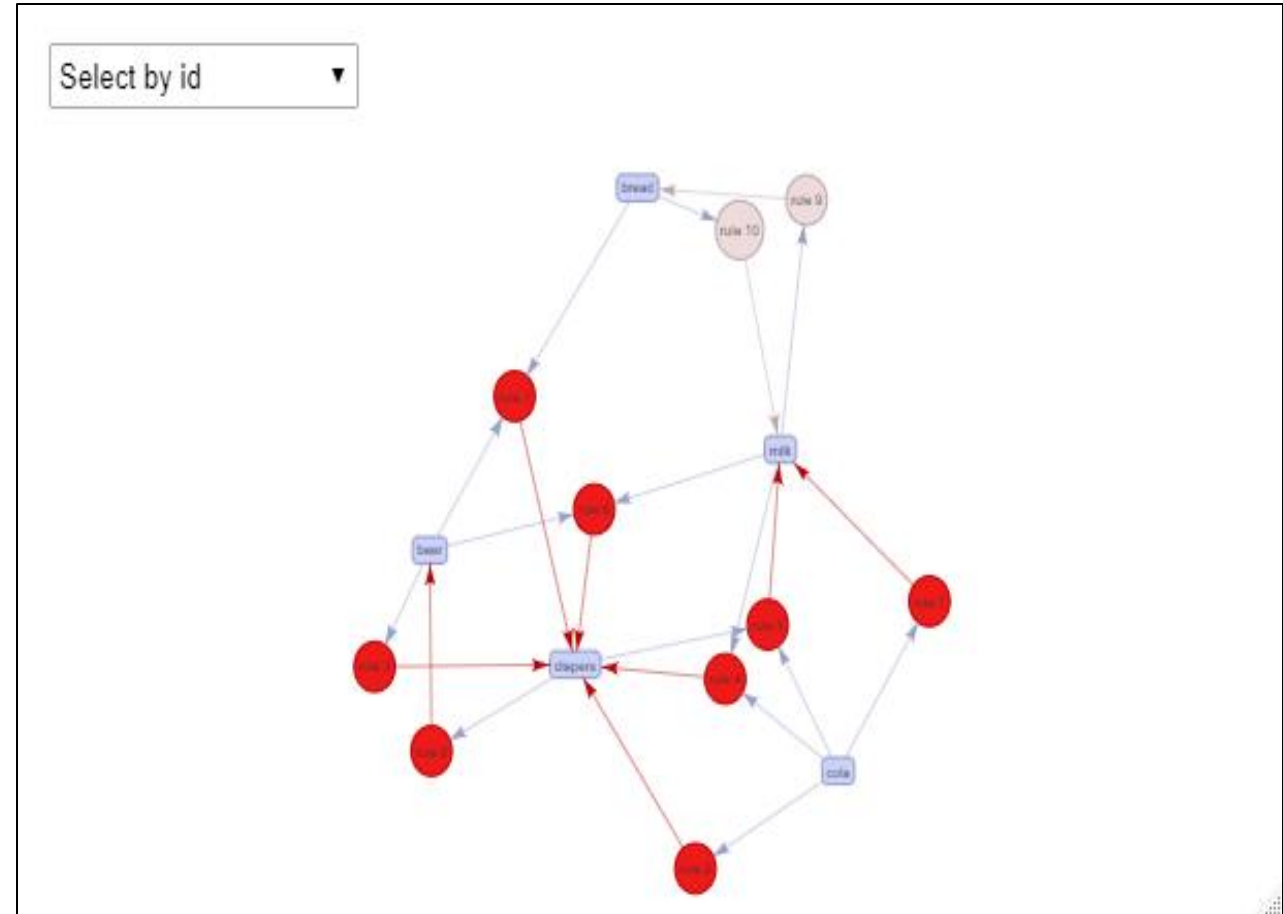


Graph based visualization:

```
#Graph based visualization
```

```
subrules <- head(rules, n = 10, by  
= "confidence")
```

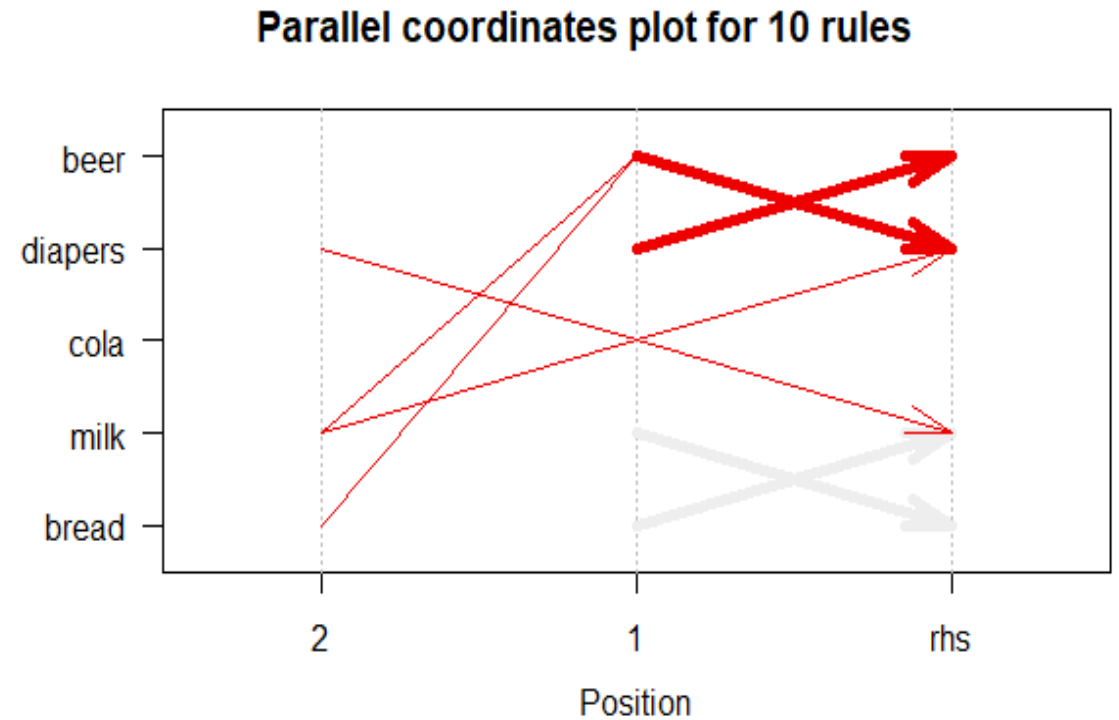
```
plot(subrules, method = "graph",  
engine = "htmlwidget")
```



Parallel coordinate plot for 10 rules:

#Paraller coordinate plot

- `plot(subrules, method="paracoord")`



More here:

- Like the one we did before:
- <https://www.kirenz.com/post/2020-05-14-r-association-rule-mining/>
- Real life example:
- <https://www.youtube.com/watch?v=91CmrpD-4Fw>

Question/queries?

- Next class
- Monte Carlo Simulations
- Class imbalance problem
 - Statistical approach
 - Data sciences approach

Thank you!

@shitalbhandary