Transactional data

MARKET BASKET ANALYSIS IN R

Christopher Bruffaerts
Statistician

What is a transaction?

Transaction: Activity of buying or selling something.

Transactional data: List of all items bought by a *customer* in a *single purchase*.

Example of one transaction:

```
  TID Product
1   1   Bread
2   1  Cheese
3   1  Cheese
4   1  Cheese
```

The transactional class in R

Transactions-class: represents transaction data used for mining itemsets or rules.

Coercion from:

- lists
- matrices
- dataframes

However, you will need to prepare your data first.

Important when considering transactional data

- Field/column used to identify a product
- Field/column used to identify a transaction

Back to the grocery store (1)

Transactional data from the store

```
my_transactions = data.frame(
  "TID" = c(1,1,1,1, 2,2,2, 3,3, 4,4,4, 5,5, 6,6, 7,7),
  "Product" = c("Bread", "Cheese", "Cheese", "Cheese",
              "Bread", "Butter", "Wine",
              "Butter", "Butter",
              "Butter", "Wine", "Wine",
              "Butter", "Cheese",
              "Cheese", "Wine",
              "Wine", "Wine")
)
```

Transaction glimpse

```
head(my_transactions, 10)
```

```
   TID Product
1    1   Bread
2    1  Butter
3    1  Cheese
4    1    Wine
5    2   Bread
6    2  Butter
7    2    Wine
8    3   Bread
9    3  Butter
10   4  Butter
```

Back to the grocery store (2)

Create lists with the split function

```
# Transform TID into a factor
my_transactions$TID =
  factor(my_transactions$TID)

# Split into groups
data_list = split(my_transactions$Product,
                  my_transactions$TID)
```

```
data_list
```

```
$`1`
[1] Bread  Butter Cheese Wine
Levels: Bread Butter Cheese Wine

$`2`
[1] Bread  Butter Wine
Levels: Bread Butter Cheese Wine

$`3`
[1] Bread  Butter
Levels: Bread Butter Cheese Wine
```

Back to the grocery store (3)

Transforming to transaction class

```
# Transform to transactional dataset
data_trx = as(data_list,"transactions")

# Inspect transactions
inspect(data_trx)
```

Inspection of the transactional data

```
    items                      transactionID
[1] {Bread,Butter,Cheese,Wine} 1
[2] {Bread,Butter,Wine}        2
[3] {Bread,Butter}             3
[4] {Butter,Cheese,Wine}       4
[5] {Butter,Cheese}            5
[6] {Cheese,Wine}              6
[7] {Butter,Wine}              7
```

More inspections of transactions

Overview of transactions

```
inspect(head(data_trx))
```

```
    items                        transactionID
[1] {Bread,Butter,Cheese,Wine} 1
[2] {Bread,Butter,Wine}        2
[3] {Bread,Butter}             3
[4] {Butter,Cheese,Wine}       4
[5] {Butter,Cheese}            5
[6] {Cheese,Wine}              6
```

Accessing specific transactions

```
inspect(data_trx[1])
inspect(data_trx[1:3])
```

Summary of the transactional object

```
summary(data_trx)
```

Overview of transactions

Plotting the ItemMatrix

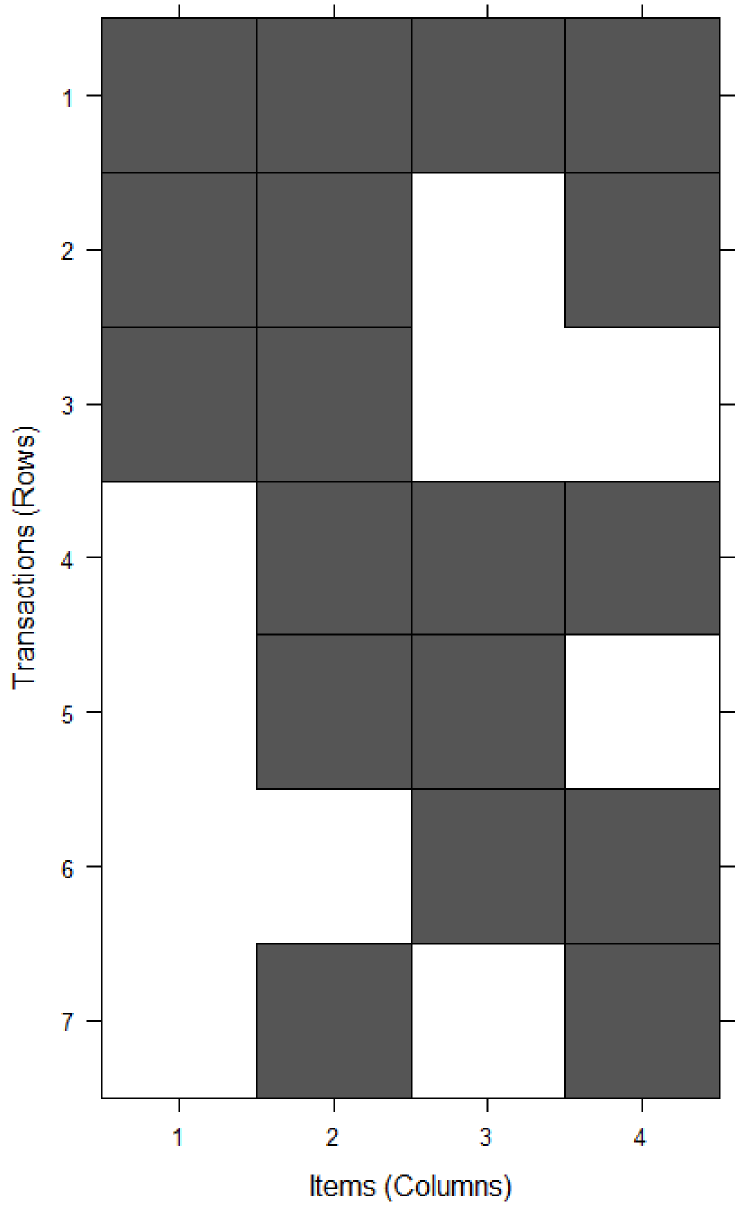
```
image(data_trx)
```

Warning: use the function on a limited number of transactions

Useful to identify:

- Patterns in the transactions
- Sparsity in the data

Density = 18/28 = 0.64



Let's inspect transactions!

MARKET BASKET ANALYSIS IN R

Metrics in market basket analysis

MARKET BASKET ANALYSIS IN R

Christopher Bruffaerts
Statistician

Metrics used for rule extraction

TID	Transaction
1	{Bread, Butter, Cheese, Wine}
2	{Bread, Butter, Wine}
3	{Bread, Butter}
4	{Butter, Cheese, Wine}
5	{Butter, Cheese}
6	{Cheese, Wine}
7	{Butter, Wine}

Goal: Extract association rules

Examples:

- {Bread} $\rightarrow$ {Butter}
  - *Bread* = "Antecedent"
  - *Butter* = "Consequent"
- {Butter, Cheese} $\rightarrow$ {Wine}

Metrics: Support, confidence, lift,...

Support measure

TID	Transaction
1	{**Bread**, Butter, *Cheese*, Wine}
2	{**Bread**, Butter, Wine}
3	{**Bread**, Butter}
4	{Butter, Cheese, Wine}
5	{Butter, Cheese}
6	{Cheese, Wine}
7	{Butter, Wine}

Support : "popularity of an itemset"

- **supp(X)** = Fraction of transactions that contain itemset X.
- **supp(X ∪ Y)** = Fraction of transactions with both X and Y.

Examples:

- supp({Bread}) = 3/7 = 42%
- supp({Bread} ∪ {Butter}) = 3/7 = 42%

Confidence measure

TID	Transaction
1	{**Bread, Butter**, Cheese, Wine}
2	{**Bread, Butter**, Wine}
3	{**Bread, Butter**}
4	{Butter, Cheese, Wine}
5	{Butter, Cheese}
6	{Cheese, Wine}
7	{Butter, Wine}

Confidence : "how often the rule is true"

$\text{conf}(X \rightarrow Y) = \text{supp}(X \cup Y) \,/\, \text{supp}(X)$

Confidence shows the percentage in which Y is bought with X.

Example:

$X = \{\text{Bread}\}$

$Y = \{\text{Butter}\}$

$\text{conf}(X \rightarrow Y) = \frac{3/7}{3/7} = 100\%$

Lift measure

TID	Transaction
1	{**Bread, Butter**, Cheese, Wine}
2	{**Bread, Butter**, Wine}
3	{**Bread, Butter**}
4	{Butter, Cheese, Wine}
5	{Butter, Cheese}
6	{Cheese, Wine}
7	{Butter, Wine}

Lift : "how strong is the association"

lift(X $\rightarrow$ Y) = $\dfrac{supp(X \cup Y)}{supp(X) \times supp(Y)}$

- Lift > 1: Y is likely to be bought with X
- Lift < 1: Y is unlikely to be bought if X is bought

Example:

X = {Bread}; Y = {Butter}

lift(X $\rightarrow$ Y) = $\frac{3/7}{(3/7)*(6/7)} = \frac{7}{6}$ ~ 1.16

The apriori function for frequent itemsets

```
library(arules)
# Frequent itemsets
supp.cw = apriori(trans, # the transactional dataset
                  # Parameter list
                  parameter=list(
                    # Minimum Support
                    supp=0.2,
                    # Minimum Confidence
                    conf=0.4,
                    # Minimum length
                    minlen=2,
                    # Target
                    target="frequent itemsets"),
                  # Appearence argument
                  appearance = list(
                    items = c("Cheese","Wine"))
                 )
```

The apriori function for rules

```
library(arules)
# Rules
rules.b.rhs = apriori(trans, # the transactional dataset
                  # Parameter list
                  parameter=list(
                    # Minimum Support
                    supp=0.2,
                    # Minimum Confidence
                    conf=0.4,
                    # Minimum length
                    minlen=2,
                    # Target
                    target="rules"),
                  # Appearence argument
                   appearance = list(
                     rhs = "Butter",
                    default = "lhs")
                 )
```

Frequent itemsets with the apriori

TID	Transaction
1	{Bread, Butter, Cheese, Wine}
2	{Bread, Butter, Wine}
3	{Bread, Butter}
4	{Butter, Cheese, Wine}
5	{Butter, Cheese}
6	{Cheese, Wine}
7	{Butter, Wine}

Retrieve the **frequent itemsets**

```
supp.all = apriori(trans,
                   parameter=list(supp=3/7,
                   target="frequent itemsets"))
inspect(head(sort(supp.all,by="support"),3))
```

```
    items          support   count
[1] {Butter}       0.8571429 6
[2] {Wine}         0.7142857 5
[3] {Cheese}       0.5714286 4
```

Inspect confidence and lift measures

TID	Transaction
1	{Bread, **Butter**, Cheese, Wine}
2	{Bread, **Butter**, Wine}
3	{Bread, **Butter**}
4	{**Butter**, Cheese, Wine}
5	{**Butter**, Cheese}
6	{Cheese, Wine}
7	{**Butter**, Wine}

Retrieve the **rules**

```
# Rules with "Butter" on rhs
rules.b.rhs = apriori(trans,
                 parameter=list(
                    minlen=2,
                    target="rules"),
                 appearance = list(
                     rhs="Butter",
                    default = "lhs")
           )
inspect(head(sort(rules.b.rhs,by="lift")), 5)
```

Inspect confidence and lift measures

TID	Transaction
1	{Bread, **Butter**, Cheese, Wine}
2	{Bread, **Butter**, Wine}
3	{Bread, **Butter**}
4	{**Butter**, Cheese, Wine}
5	{**Butter**, Cheese}
6	{Cheese, Wine}
7	{**Butter**, Wine}

Retrieve the **rules**

```
    lhs                        rhs      support  confidence lift  count
[1] {Bread}                 => {Butter} 0.42       1.0          1.16      3
[2] {Bread,Cheese}          => {Butter} 0.14       1.0          1.16      1
[3] {Bread,Wine}            => {Butter} 0.28       1.0          1.16      2
[4] {Bread,Cheese,Wine}     => {Butter} 0.14       1.0          1.16      1
[5] {Wine}                  => {Butter} 0.57       0.8          0.93      4
```

Let's practice!

MARKET BASKET ANALYSIS IN R

The apriori algorithm

MARKET BASKET ANALYSIS IN R

Christopher Bruffaerts
Statistician

Association rule mining

Association rule mining allows to discover interesting relationships between items in a large transactional database.

This mining task can be divided into two subtasks:

- **Frequent itemset generation**: determine all frequent itemsets of a potentially large database of transactions. An itemset is said to be frequent if it satisfies a *minimum support threshold*.
- **Rule generation**: from the above frequent itemsets, generate association rules with confidence above a *minimum confidence threshold*.

The **apriori algorithm** is a classic and fast mining algorithm belonging to the class of association rule mining algorithms.

Idea behind the apriori algorithm

The apriori algorithm:

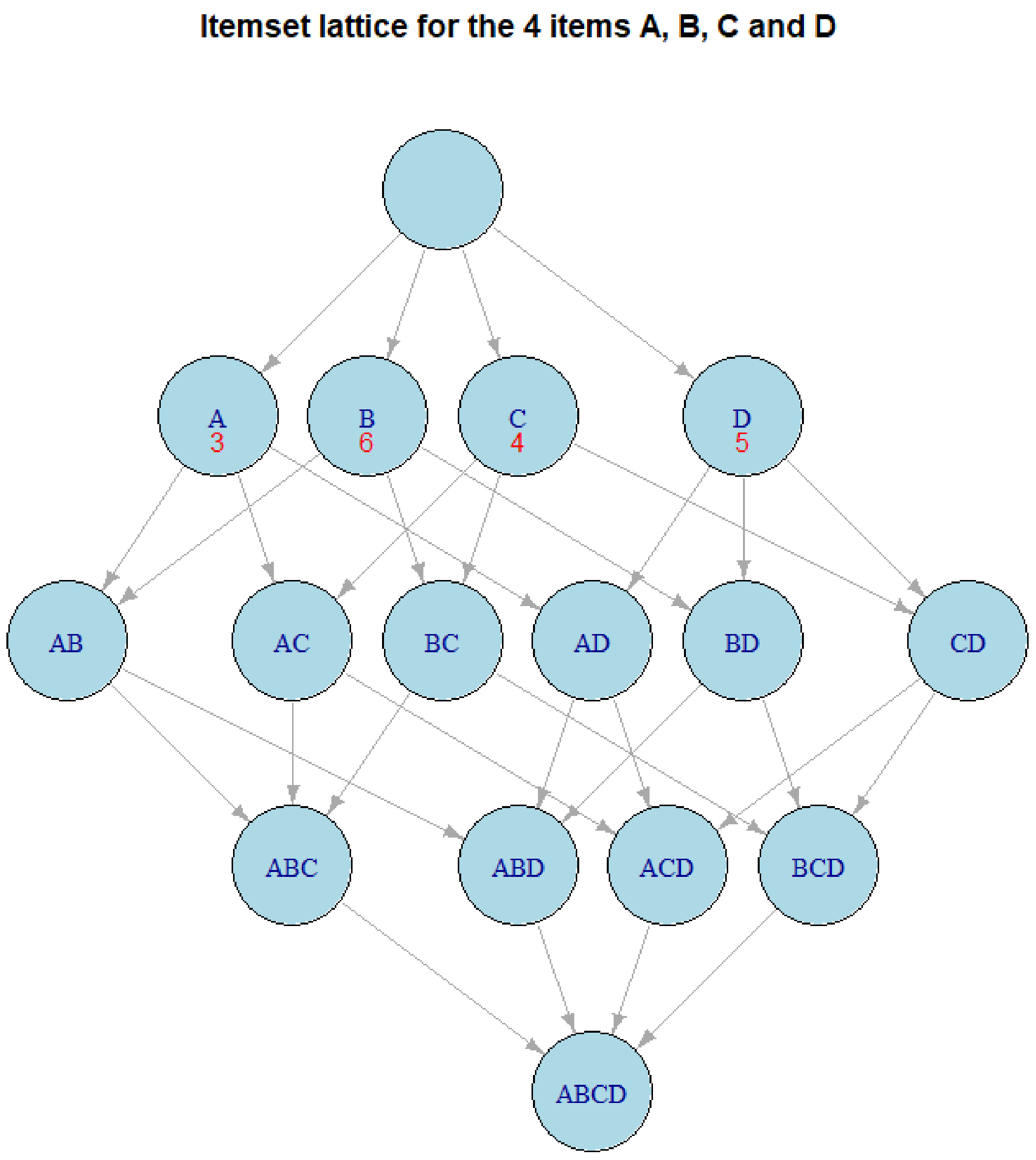
- Bottom-up approach
- Generates candidate itemsets by exploiting the apriori principle

Apriori principle:

- If an itemset is frequent, then all of its subsets must also be frequent.
  - *e.g.* if {A,B} is frequent, then both {A} and {B} are frequent
- For an infrequent itemset, all its super-sets are infrequent.
  - *e.g.* if {A} is infrequent, then {A,B}, {A,C} and {A,B,C} are infrequent.

[1] Agrawal and Srikant (1994)

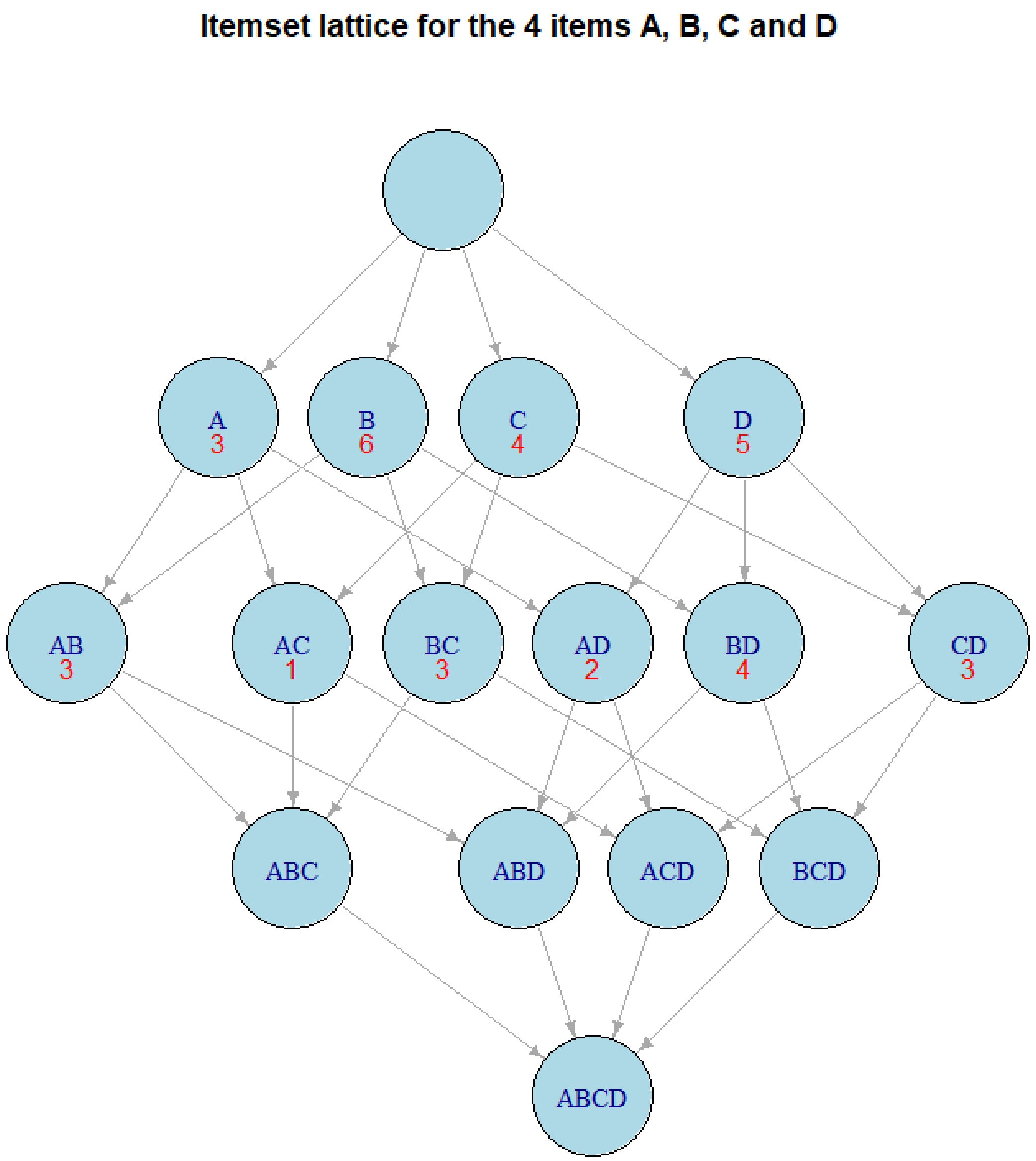
Example: 1-itemset



TID	Transaction
1	{A, B, C, D}
2	{A, B, D}
3	{A, B}
4	{B, C, D}
5	{B, C}
6	{C, D}
7	{B, D}

[1] Minimum support threshold = 3/7 = 0.42

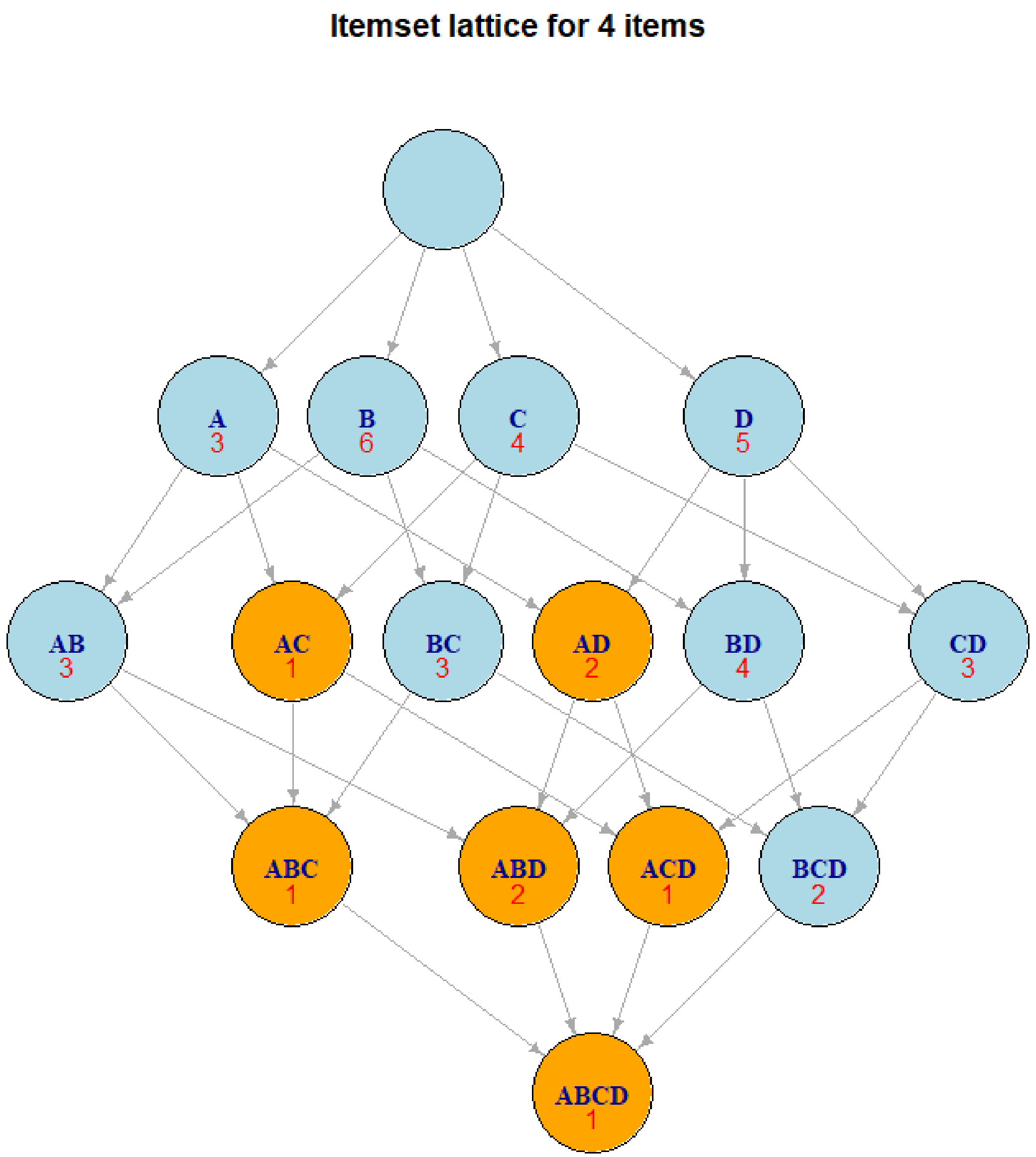
Example: 2-itemsets



TID	Transaction
1	{A, B, C, D}
2	{A, B, D}
3	{A, B}
4	{B, C, D}
5	{B, C}
6	{C, D}
7	{B, D}

[1] Minimum support threshold = 3/7 = 0.42

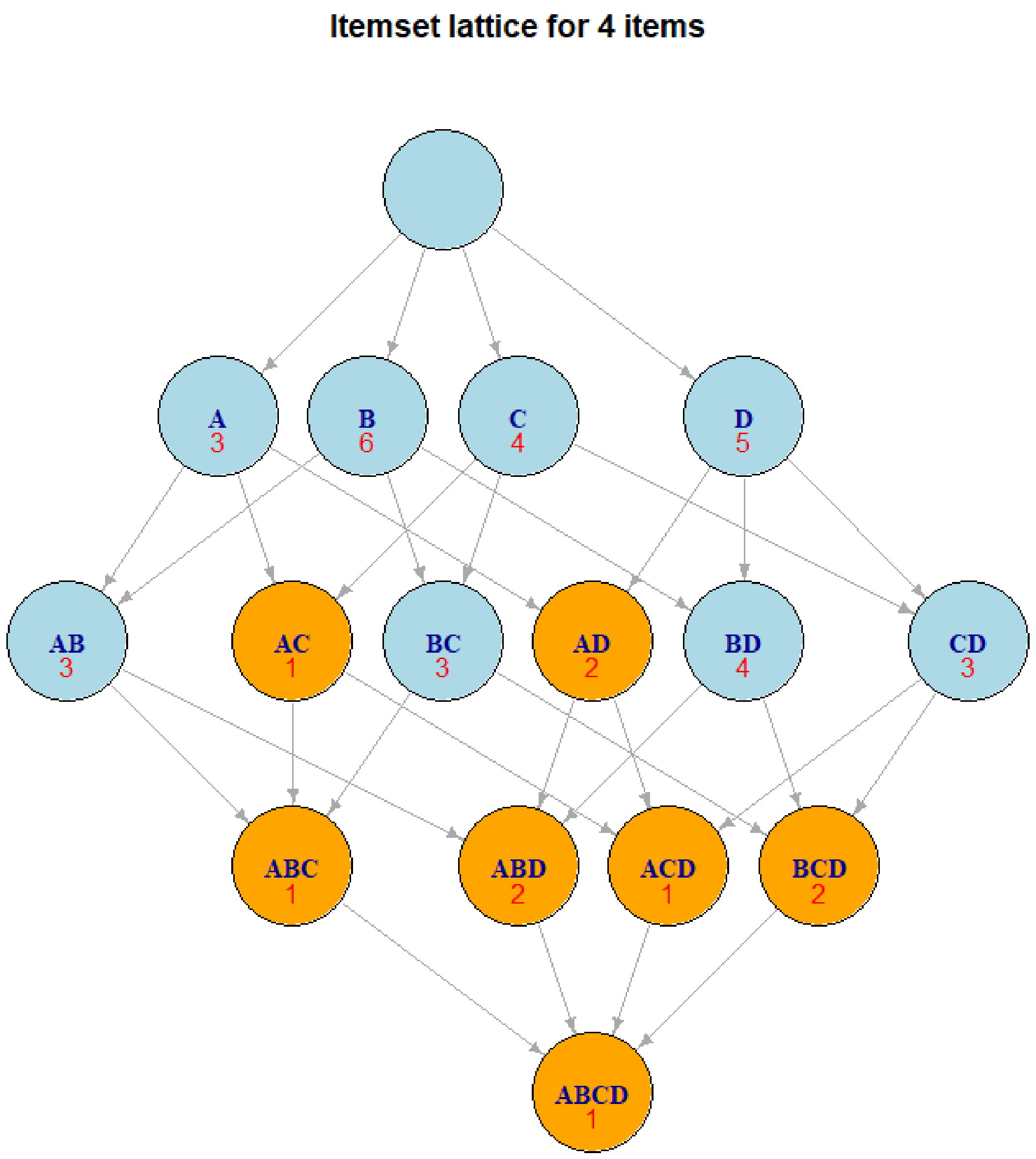
Example: 3-itemsets



TID	Transaction
1	{A, B, C, D}
2	{A, B, D}
3	{A, B}
4	{B, C, D}
5	{B, C}
6	{C, D}
7	{B, D}

[1] Minimum support threshold = 3/7 = 0.42

Example: frequent itemsets



Itemset	Count	Support
{A}	3	0.42
{B}	6	0.85
{C}	4	0.57
{D}	5	0.71
{A,B}	3	0.42
{B,C}	3	0.42
{B,D}	4	0.57

[1] Minimum support threshold = 3/7 = 0.42

Apriori: rule generation

After the computationally expensive frequent itemset generation, apriori generates rules:

- Start with high-confidence rules with single precedent
  - *e.g.* $\{A,C\} \rightarrow \{B\}$
- Build more complex rules, with more items on the right hand side
  - *e.g.* $\{A,C\} \rightarrow \{B, D\}$

Trick: pruning of association rule

e.g.: if the rule $\{B,C,D\} \rightarrow \{A\}$ has low confidence, all rules containing item A in its consequent can be discarded (such as the rule $\{B,D\} \rightarrow \{A, C\}$ or $\{D\} \rightarrow \{A,B, C\}$).

A first try with the apriori

Transactional data

```
inspect(head(trans,2))
```

```
    items     transactionID
[1] {A,B,C,D} 1
[2] {A,B,D}   2
```

First call to the apriori function - frequent itemsets

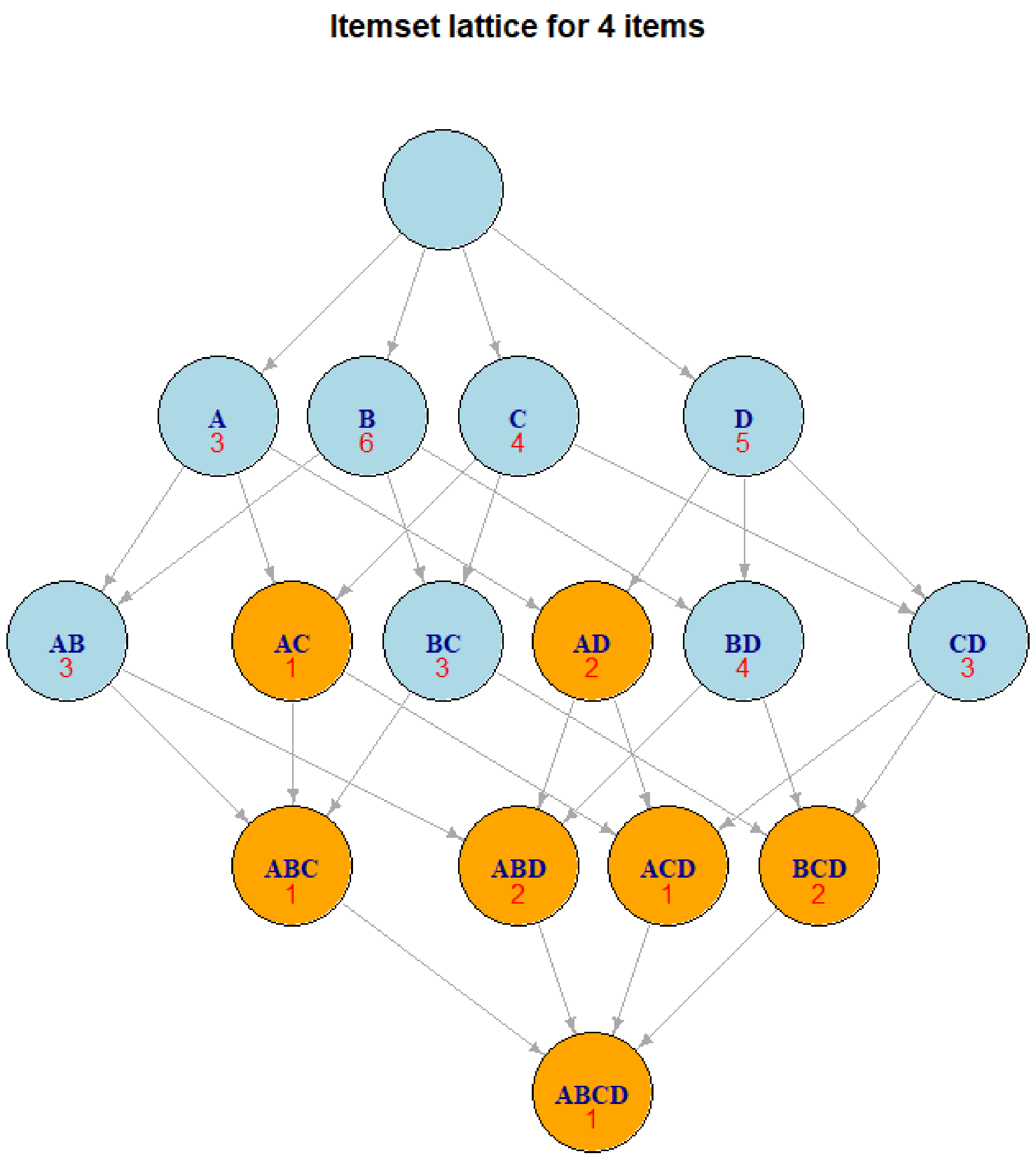
```
support.all = apriori(trans,
                      parameter = list(supp = 3/7, target="frequent itemsets"))
```

Output of the apriori - frequent itemsets

Frequent itemsets

```
inspect(support.all)
```

```
    items support   count
[1] {A}   0.4285714 3
[2] {C}   0.5714286 4
[3] {D}   0.7142857 5
[4] {B}   0.8571429 6
[5] {A,B} 0.4285714 3
[6] {C,D} 0.4285714 3
[7] {B,C} 0.4285714 3
[8] {B,D} 0.5714286 4
```



Extracting rules with the apriori function

Parameter: the mining parameters change the characteristics of the mined itemsets or rules.

- Support = 3/7
- Confidence = 60%
- Minimum length of rule = 2

Call to the apriori function for rule generation with specific arguments

```
rules.all = apriori(trans,
                parameter = list(supp=3/7, conf=0.6, minlen=2),
                control = list(verbose=F)
                   )
```

Extracting rules: output

Inspecting the rules

```
inspect(rules.all)
```

```
    lhs     rhs support   confidence lift      count
[1] {A} => {B} 0.4285714 1.0000000  1.1666667 3
[2] {C} => {D} 0.4285714 0.7500000  1.0500000 3
[3] {D} => {C} 0.4285714 0.6000000  1.0500000 3
[4] {C} => {B} 0.4285714 0.7500000  0.8750000 3
[5] {D} => {B} 0.5714286 0.8000000  0.9333333 4
[6] {B} => {D} 0.5714286 0.6666667  0.9333333 4
```

Let's practice!

MARKET BASKET ANALYSIS IN R

“If this then that” with the apriori

MARKET BASKET ANALYSIS IN R

Christopher Bruffaerts
Statistician

Recap of extracted rules (1)

TID	Transaction
1	{Bread, Butter, Cheese, Wine}
2	{Bread, Butter, Wine}
3	{Bread, Butter}
4	{Butter, Cheese, Wine}
5	{Butter, Cheese}
6	{Cheese, Wine}
7	{Butter, Wine}

Apply apriori on transactions:

```
rules = apriori(data_trx,
                    parameter = list(
                      supp = 3/7, conf = 0.6,
                      minlen = 2),
                    control = list(verbose=F)
)
```

Recap of extracted rules (2)

Create dataframe with extracted rules

```
df_rules = as(rules, "data.frame")
df_rules
```

```
                 rules   support confidence      lift count
1  {Bread} => {Butter} 0.4285714  1.0000000 1.1666667     3
2   {Cheese} => {Wine} 0.4285714  0.7500000 1.0500000     3
3   {Wine} => {Cheese} 0.4285714  0.6000000 1.0500000     3
4 {Cheese} => {Butter} 0.4285714  0.7500000 0.8750000     3
5   {Wine} => {Butter} 0.5714286  0.8000000 0.9333333     4
6   {Butter} => {Wine} 0.5714286  0.6666667 0.9333333     4
```

Appearance of frequent itemsets

Frequent itemsets for Cheese and Wine

```
supp_cheese_wine =
    apriori(trans,
        parameter = list(
          target = "frequent itemsets",
          supp = 3/7),
        appearance = list(
          items = c("Cheese",  "Wine"))
)
```

```
inspect(supp_cheese_wine)
```

```
    items         support   count
[1] {Cheese}      0.5714286 4
[2] {Wine}        0.7142857 5
[3] {Cheese,Wine} 0.4285714 3
```

Appearance of extracted rules

Specific rules for Cheese

```
rules_cheese_rhs = apriori(data = trans,
                   parameter = list(supp=3/7,conf=0.2, minlen=2),
                   appearance = list(rhs="Cheese"),
                   control = list (verbose=F))
```

```
inspect(rules_cheese_rhs)
```

```
    lhs         rhs      support   confidence lift  count
[1] {Wine}   => {Cheese} 0.4285714 0.6        1.050 3
[2] {Butter} => {Cheese} 0.4285714 0.5        0.875 3
```

Redundant rules

What is a redundant rule?

A rule is redundant if a more general rule with the same or a higher confidence exists.

Super-rule:

A rule is more general if it has the same RHS but one or more items removed from the LHS.

Example:

Super-rules of $\{A\} \rightarrow \{C\}$:

- $\{A, B\} \rightarrow \{C\}$
- $\{A, B, D\} \rightarrow \{C\}$

Non-redundant rules are defined as:

- all other rules are super-rules of that rule
- all other rules have a lower confidence

Rule redundancy (1)

Set of generated rules

```
rules = apriori(trans,control = list(verbose=F),
                parameter = list(supp=0.05, conf=0.5, minlen=2),
                appearance = list(rhs="Bread", default = "lhs"))
```

Set of pruned rules (non-redundant)

```
redundant_rules = is.redundant(rules)
non_redundant_rules = rules[!redundant_rules]
```

Rule redundancy (2)

Comparing extracted rules and non-redundant rules

```
inspect(rules)
```

```
    lhs                       rhs     support   confidence lift     count
[1] {Butter}               => {Bread} 0.4285714 0.5        1.166667 3
[2] {Butter,Wine}          => {Bread} 0.2857143 0.5        1.166667 2
[3] {Butter,Cheese,Wine} => {Bread} 0.1428571 0.5        1.166667 1
```

```
inspect(non_redundant_rules)
```

```
    lhs           rhs     support   confidence lift     count
[1] {Butter} => {Bread} 0.4285714 0.5        1.166667 3
```

Let's follow the rules!

MARKET BASKET ANALYSIS IN R