Association rules mining

What are Association Rules?

- Study of “what goes with what”
  - “Customers who bought X also bought Y”
  - What symptoms go with what diagnosis
- Transaction-based or event-based
- Also called “market basket analysis” and “affinity analysis”
- Originated with study of customer transactions databases to determine associations among items purchased

Used in many recommender systems

Bound Away
Last Train Home

Share your own customer images

List Price: $16.98
Price: **$16.98** and eligible for **FREE Super Saver Shipping** on orders over $25. See details.

Availability: Usually ships within 24 hours

Want it delivered Tomorrow? Order it in the next 4 hours and 9 minutes, and choose **One-Day S** checkout. See details.

41 used & new from $6.99

See more product details

Based on customer purchases, this is the #82 Early Adopter Product in Alternative Rock.

801x612

Buy this title for only $.01 when you get a new Amazon Visa® Card

Apply now and if you're approved instantly, **save $30** off your first purchase, earn **3% rewards**, get a **0% APR**,* and pay **no**

Amazon Visa discount: $30.00
Applied to this item: - $16.97
Discount remaining: $13.03
Find out how
(Don't show again)

Customers who bought this title also bought:

- Time and Water ~ Last Train Home (Why?)
- Cold Roses ~ Ryan Adams & the Cardinals (Why?)
- Tambourine ~ Tift Merritt (Why?)
- Last Train Home ~ Last Train Home (Why?)
- True North ~ Last Train Home (Why?)
- Universal United House of Prayer ~ Buddy Miller (Why?)
- Wicked Twisted Road [ENHANCED] ~ Reckless Kelly (Why?)
- Hacienda Brothers ~ Hacienda Brothers (Why?)

Terms

"IF" part = **antecedent**

"THEN" part = **consequent**

"Item set" = the items (e.g., products) comprising the antecedent or consequent

- Antecedent and consequent are *disjoint* (i.e., have no items in common)

Example

IF ***diaper and milk*** THEN ***beer***

diaper and milk are antecedent

beer is consequent

In R we call them
diaper and milk – LHS-left hand side
Beer – RHS – right hand side

General representation of the rule

{Bread, Milk} ⟶ {Diapers}

Tiny Example: Phone Faceplates

Market basket transaction example

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

Many Rules are Possible

For example: Transaction 2 supports several rules, such as

- "If **bread**, then **diapers**" ("If a bread is purchased, then so are diapers")
- "If **diapers** then **Eggs**"
- "If **bread and beer**, then **Eggs**"
- + several more

Frequent Item Sets

- Ideally, we want to create all possible combinations of items
- **Problem:** computation time grows exponentially as # items increases

The total number of rules extracted from a dataset with d items:

$$R = 3^d - 2^{d+1} + 1$$

- **Solution:** consider only "frequent item sets"
- Criterion for frequent: *support*

Support

Support for an itemset = # (or percent) of transactions that include an itemset

Example: support for the item set {milk, diapers} is 3 out of 5, or 60%

Support for a rule = # (or percent) of transactions that include both the antecedent and the consequent

Support

Support for the itemset {Bread, Milk} ??

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

Apriori Algorithm

Apriori algorithm

Generate only rules for frequent itemsets

- If the itemset is frequent, then all of its subsets must also be frequent
- If the itemset is not frequent then all of its supersets will be not frequent as well.

- Algorithm looks only for frequent itemsets and then generate rules based on them.
- You need to specify the threshold for support.

Apriori algorithm

If itemset is infrequent, then all rules created from only the items from that itemset are infrequent as well.

If itemset {Beer, Diapers, Milk} are infrequent, then all the following rules are infrequent as well and can be pruned

$$\{\text{Beer}, \text{Diapers}\} \longrightarrow \{\text{Milk}\}, \quad \{\text{Beer}, \text{Milk}\} \longrightarrow \{\text{Diapers}\},$$
$$\{\text{Diapers}, \text{Milk}\} \longrightarrow \{\text{Beer}\}, \quad \{\text{Beer}\} \longrightarrow \{\text{Diapers}, \text{Milk}\},$$
$$\{\text{Milk}\} \longrightarrow \{\text{Beer},\text{Diapers}\}, \quad \{\text{Diapers}\} \longrightarrow \{\text{Beer},\text{Milk}\}.$$

Apriori algorithm

- If the itemset is frequent, then all of its subsets must also be frequent
- If the itemset is infrequent then all of its supersets will be infrequent as well.

- Algorithm looks only for frequent itemsets and then generate rules based on them.
- You need to specify the threshold for support.

The downward closure property of frequent patterns.

Any subset of a frequent itemset must be frequent

If {beer, diaper, milk} is frequent, so is {beer, diaper}

i.e., every transaction having {beer, diaper, milk} also contains {beer, diaper}

Apriori algorithm

Threshold for support: 0.3 (30%), so any itemset that has support of higher then 0.3 is considered as frequent.

Support for itemset {Bread, Milk, Diapers} is 40%, it is frequent,
Subset of this itemset, {Bread, Milk} is going to be at least 40%, in our case, 60%

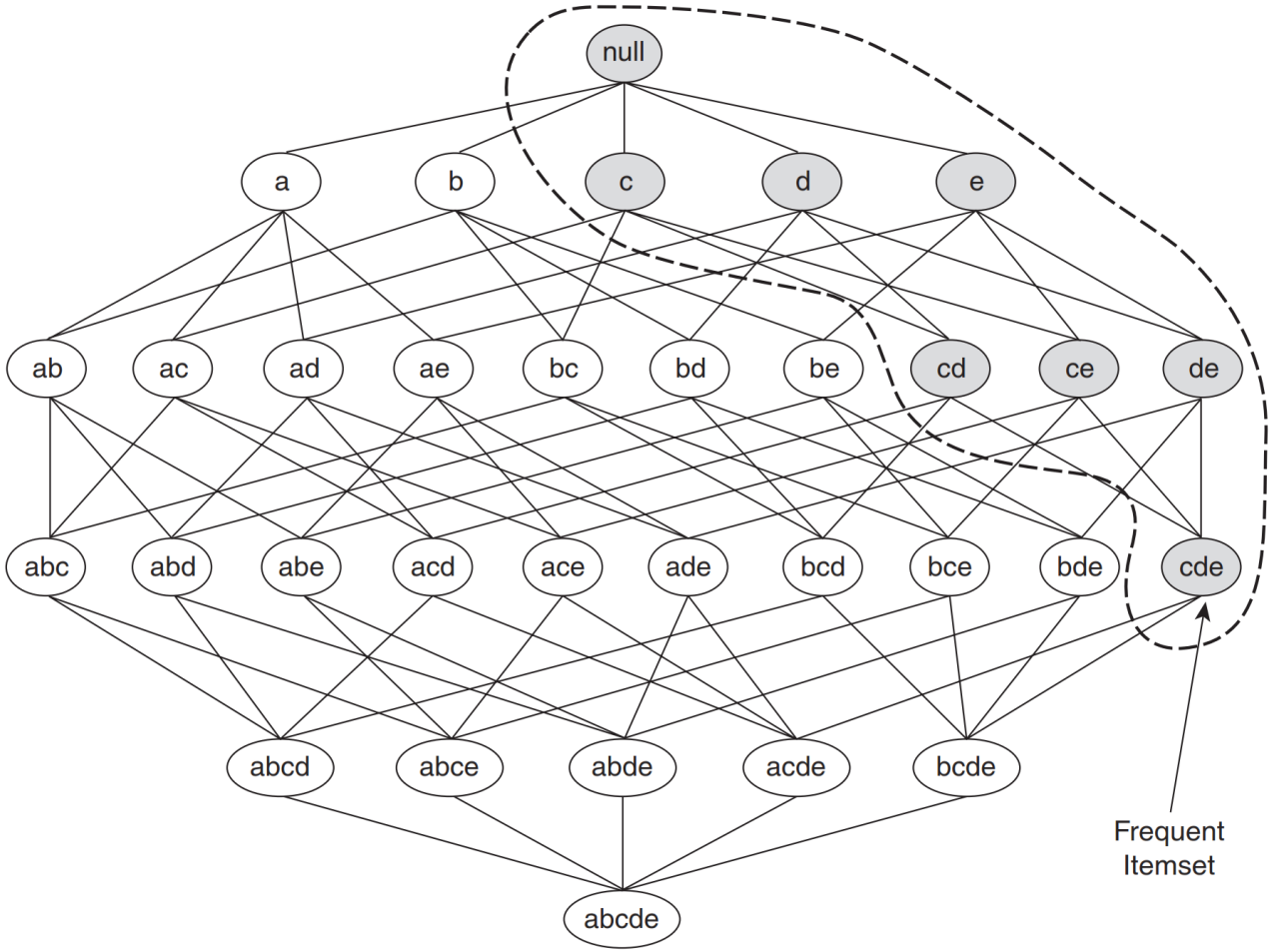
TID	Items
1	{Bread, Milk}
2	{Bread, Diapers, Beer, Eggs}
3	{Milk, Diapers, Beer, Cola}
4	{Bread, Milk, Diapers, Beer}
5	{Bread, Milk, Diapers, Cola}

Apriori algorithm

- Suppose we have the following items: a,b,c,d,e
- Suppose itemset {c,d,e} is frequent, then any transaction that contains c,d,e also contains {c,d}, {d,e},{c,e}, {c}, {d}, {e}
- As a result if {c,d,e} is frequent, then all its subets are frequent as well

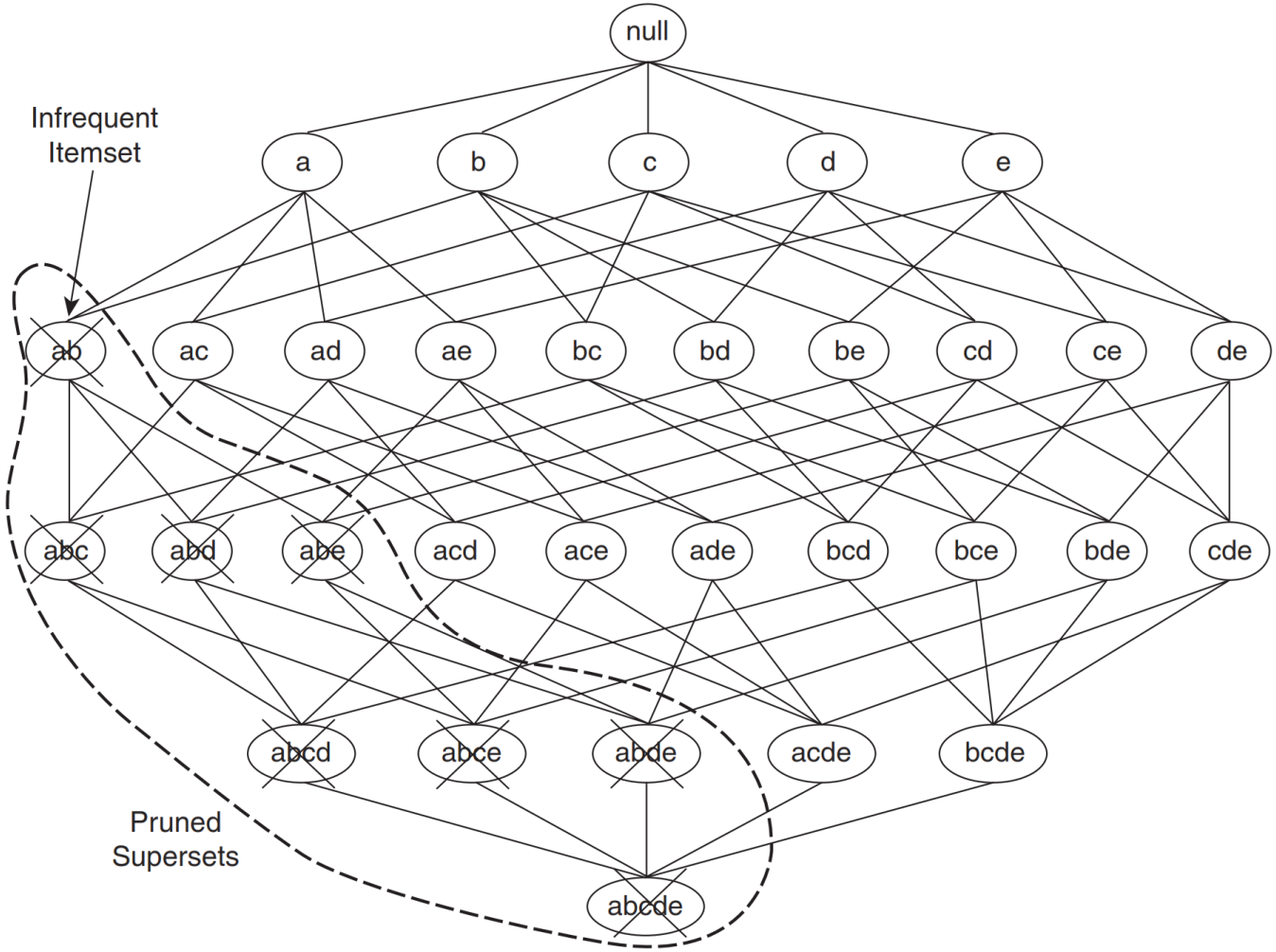
Apriori algorithm

All possible subsets of frequent itemset {c,d,e} the shaded area



Apriori algorithm

- If the itemset {a,b} is infrequent then all its supersets (sets containing a and b) are going to be infrequent as well



Apriori algorithm

- Generate frequent itemsets that satisfies have support over some threshold value, called **minsup**

- Generate rules on this subset of itemsets, that satisfies some condition for confidence (we look at this later)

The result of apriori algorithm is a set of rules filtered by some threshold for ***support and confidence***

Measures of rules interestingness

$$\{A\} \Rightarrow \{B\}$$

Measure 1

Support of the rule = $P(A \cap B)$ percentage of the transactions that include both A and B

Measure 2

Confidence = $P(B|A) = \frac{P(B \cap A)}{P(A)}$ – % of antecedent transactions that contains also consequent itemsets

Measure 3.

Lift = $\frac{\mathrm{P(B|A)}}{\mathrm{P(B)}} = \frac{\boldsymbol{P(A \cap B)}}{\boldsymbol{P(A)}} * \frac{\mathbf{1}}{\boldsymbol{P(B)}}$

Measures of Rule Performance

***Confidence*:** the % of antecedent transactions that also have the consequent item set

Lift = *confidence*/(*benchmark confidence*)

Benchmark confidence = transactions with consequent as % of all transactions

Lift > 1 indicates a rule that is useful in finding consequent items sets (i.e., more useful than just selecting transactions randomly)

Alternate Data Format: Binary Matrix

TID	Bread	Milk	Diapers	Beer	Eggs	Cola
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

Process of Rule Selection

Generate all rules that meet specified support & confidence

- Find frequent item sets (those with sufficient support – see above)
- From these item sets, generate rules with sufficient confidence

Example: Rules from {bread, diapers, milk}

{bread, diapers} -> {milk}

Support: 2/5, 0.4

Confidence: 2/3

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

Example: Rules from {bread, diapers, milk}

{bread} -> {milk}

Support: 3/5, 0.6

Confidence: 3/4

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

Interpretation

- *Lift ratio* shows how effective the rule is in finding consequents (useful if finding particular consequents is important)
- *Confidence* shows the rate at which consequents will be found (useful in learning costs of promotion)
- *Support* measures overall impact

Faceplat

Transaction	Red	White	Blue	Orange	Green	Yellow
1	1	1	0	0	1	0
2	0	1	0	1	0	0
3	0	1	1	0	0	0
4	1	1	0	1	0	0
5	1	0	1	0	0	0
6	0	1	1	0	0	0
7	1	0	1	0	0	0
8	1	1	1	0	1	0
9	1	1	1	0	0	0
10	0	0	0	0	0	1

Calculate the support, confidence and lift for the rules

{Red, White} ->Blue
{Red}->{White}

Doing it in R

Get the libraries first

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

Read the data

```
faceplate1<-read.csv("faceplate.csv")
```

Make it a matrix and take out the first column with transaction ID's

```
faceplate<-as.matrix(faceplate1[,-1])
# Create transaction object
txn<-as(faceplate, "transactions")
```

Doing it in R

use function inspect() to look at the transactions

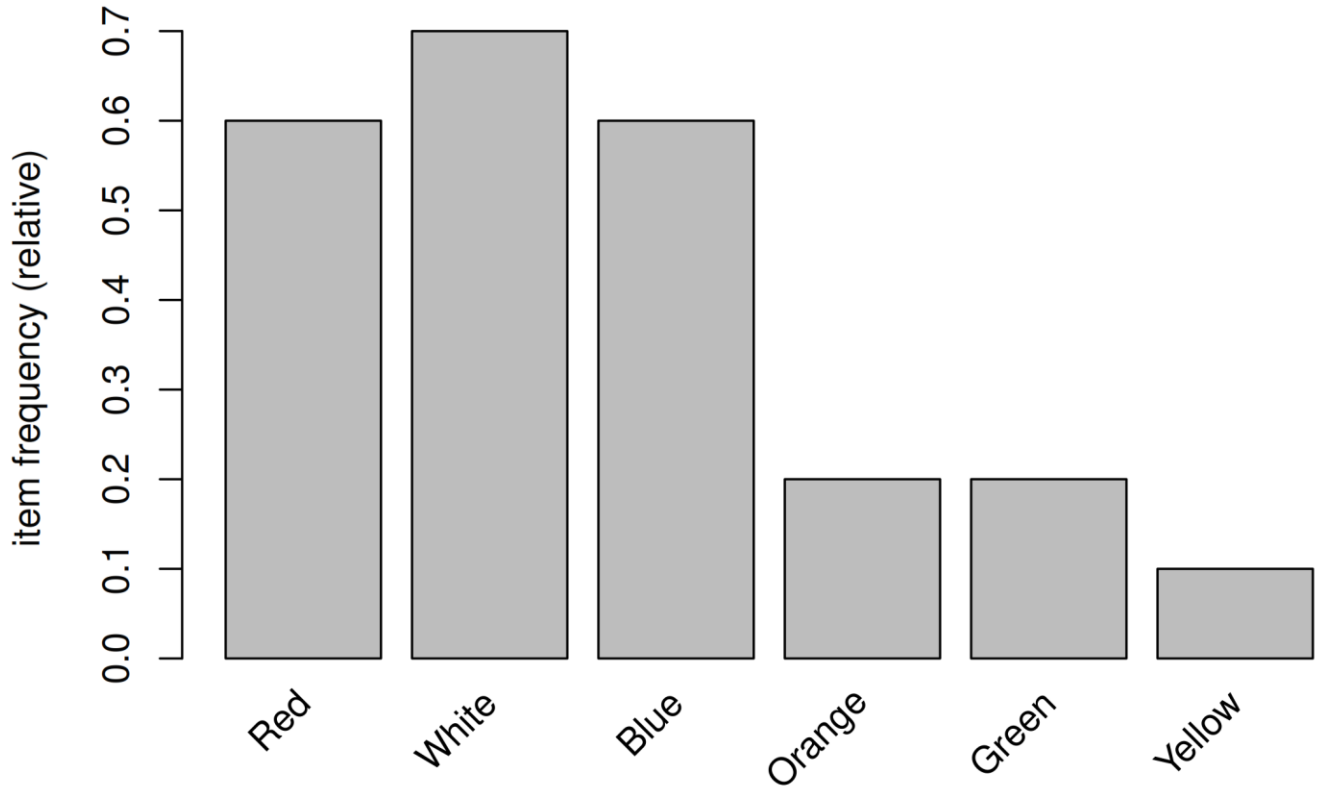
```
inspect(txn)
```

```
##      items
## [1]  {Red,White,Green}
## [2]  {White,Orange}
## [3]  {White,Blue}
## [4]  {Red,White,Orange}
## [5]  {Red,Blue}
## [6]  {White,Blue}
## [7]  {Red,Blue}
## [8]  {Red,White,Blue,Green}
## [9]  {Red,White,Blue}
## [10] {Yellow}
```

Doing it in R

Plot item frequency (look for help of the command for different features)

```
itemFrequencyPlot(txn)
```



Doing it in R

```
rules <- apriori(txn, parameter = list(minlen = 2, maxlen = 2,
                                       supp = 0.001, conf = 0.001))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##       0.001    0.1    1 none FALSE            TRUE       5   0.001      2
##  maxlen target   ext
##       2  rules FALSE
##
```

parameters for the algorithm:
minlen,maxlen: minimum and maximum length of the rule: the number of items on left and right hands together
supp: threshold for support
conf: threshold for confidence

Doing it in R

Inspect the rules

```
inspect(rules)
```

```
##      lhs          rhs      support confidence lift
## [1]  {Orange} => {Red}    0.1     0.5000000  0.8333333
## [2]  {Red}    => {Orange} 0.1     0.1666667  0.8333333
## [3]  {Orange} => {White}  0.2     1.0000000  1.4285714
## [4]  {White}  => {Orange} 0.2     0.2857143  1.4285714
## [5]  {Green}  => {Blue}   0.1     0.5000000  0.8333333
## [6]  {Blue}   => {Green}  0.1     0.1666667  0.8333333
## [7]  {Green}  => {Red}    0.2     1.0000000  1.6666667
## [8]  {Red}    => {Green}  0.2     0.3333333  1.6666667
## [9]  {Green}  => {White}  0.2     1.0000000  1.4285714
## [10] {White}  => {Green}  0.2     0.2857143  1.4285714
## [11] {Blue}   => {Red}    0.4     0.6666667  1.1111111
## [12] {Red}    => {Blue}   0.4     0.6666667  1.1111111
## [13] {Blue}   => {White}  0.4     0.6666667  0.9523810
## [14] {White}  => {Blue}   0.4     0.5714286  0.9523810
## [15] {Red}    => {White}  0.4     0.6666667  0.9523810
## [16] {White}  => {Red}    0.4     0.5714286  0.9523810
```

Doing it in R

Get the first 5 rules

```
inspect(rules[1:5])
```

```
##     lhs          rhs      support confidence lift
## [1] {Orange} => {Red}    0.1     0.5000000  0.8333333
## [2] {Red}    => {Orange} 0.1     0.1666667  0.8333333
## [3] {Orange} => {White}  0.2     1.0000000  1.4285714
## [4] {White}  => {Orange} 0.2     0.2857143  1.4285714
## [5] {Green}  => {Blue}   0.1     0.5000000  0.8333333
```

Doing it in R

```
# create a dataframe from the rules
rules_df<-as(rules, "data.frame")
# lets calculate metrics for the first rule: {Orange}=>{Red}
rules_df[1,]
```

```
##                rules support confidence      lift
## 1 {Orange} => {Red}     0.1        0.5 0.8333333
```

Doing it in R

Do the calculations by hand: support

```
# support
table(faceplate1$Orange, faceplate1$Red)
```

```
##
##     0 1
##   0 3 5
##   1 1 1
```

```
# So we have one trasnaction with both orange and red, the support is:
1/10
```

Doing it in R

Confidence

```
# Confidence is the Probability of Red being in the transaction if Orange was there
# P(Red and Orange)/P(Orange), or P(Red|Orange)
prop.table(table(faceplate1$Orange, faceplate1$Red),1)
```

```
##
##          0     1
##   0 0.375 0.625
##   1 0.500 0.500
```

```
1/2
```

Doing it in R

Lift

```
# Lift
# First get the probability of someone buying a Red faceplate
table(faceplate1$Red)
```

```
##
## 0 1
## 4 6
```

```
6/10
```

```
## [1] 0.6
```

```
# this is the baseline "prediction": Now if we know that someone has bought Orange,
# by how much is the probability of buying Red is increased (lifted):
# P(Red|Orange)/P(Red)
0.5/0.6
```

```
## [1] 0.8333333
```

Doing it in R

Now lets ease a little the rules: no restriction on maximum length the default parameters are: minimum support of 0.1, minimum confidence of 0.8, maximum of 10 items (maxlen), and a maximal time for subset checking of 5 seconds (maxtime).

```
rules1 <- apriori(txn,
                  parameter = list(supp = 0.001, conf = 0.001))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##       0.001    0.1    1 none FALSE            TRUE       5   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
...
```

Doing it in R

Lets have a look at the first 5 rules

```
inspect(rules1[1:6])
```

```
##     lhs    rhs       support confidence lift
## [1] {}  => {Yellow} 0.1     0.1        1
## [2] {}  => {Orange} 0.2     0.2        1
## [3] {}  => {Green}  0.2     0.2        1
## [4] {}  => {Blue}   0.6     0.6        1
## [5] {}  => {Red}    0.6     0.6        1
## [6] {}  => {White}  0.7     0.7        1
```

These are the relative frequencies of each item in the transaction set

Doing it in R

Remove it

```
rules1 <- apriori(txn,
                  parameter = list(supp = 0.001, conf = 0.001, minlen=2))
```

```
inspect(rules1)
```

```
##      lhs                rhs       support confidence lift
## [1]  {Orange}        => {Red}     0.1     0.5000000  0.8333333
## [2]  {Red}           => {Orange}  0.1     0.1666667  0.8333333
## [3]  {Orange}        => {White}   0.2     1.0000000  1.4285714
## [4]  {White}         => {Orange}  0.2     0.2857143  1.4285714
## [5]  {Green}         => {Blue}    0.1     0.5000000  0.8333333
## [6]  {Blue}          => {Green}   0.1     0.1666667  0.8333333
## [7]  {Green}         => {Red}     0.2     1.0000000  1.6666667
## [8]  {Red}           => {Green}   0.2     0.3333333  1.6666667
## [9]  {Green}         => {White}   0.2     1.0000000  1.4285714
## [10] {White}         => {Green}   0.2     0.2857143  1.4285714
## [11] {Blue}          => {Red}     0.4     0.6666667  1.1111111
## [12] {Red}           => {Blue}    0.4     0.6666667  1.1111111
## [13] {Blue}          => {White}   0.4     0.6666667  0.9523810
```

Doing it in R

You can sort the rules by one of the characteristics

```
rules_l<-sort(rules1, by="lift")
inspect(rules_l)
```

```
##      lhs                    rhs      support confidence lift
## [1]  {Red,White}         => {Green}  0.2     0.5000000  2.5000000
## [2]  {Red,White,Blue}    => {Green}  0.1     0.5000000  2.5000000
## [3]  {Green}             => {Red}    0.2     1.0000000  1.6666667
## [4]  {Blue,Green}        => {Red}    0.1     1.0000000  1.6666667
## [5]  {White,Green}       => {Red}    0.2     1.0000000  1.6666667
## [6]  {White,Blue,Green}  => {Red}    0.1     1.0000000  1.6666667
## [7]  {Red}               => {Green}  0.2     0.3333333  1.6666667
## [8]  {Orange}            => {White}  0.2     1.0000000  1.4285714
## [9]  {Green}             => {White}  0.2     1.0000000  1.4285714
```

Doing it in R

Do some other subseting: only rules having Red in right side

```
rules_r<-subset(rules1, rhs %in% 'Red')
inspect(rules_r)
```

```
##     lhs                    rhs   support confidence lift
## [1] {Orange}           => {Red} 0.1     0.5000000  0.8333333
## [2] {Green}            => {Red} 0.2     1.0000000  1.6666667
## [3] {Blue}             => {Red} 0.4     0.6666667  1.1111111
## [4] {White}            => {Red} 0.4     0.5714286  0.9523810
## [5] {White,Orange}     => {Red} 0.1     0.5000000  0.8333333
## [6] {Blue,Green}       => {Red} 0.1     1.0000000  1.6666667
## [7] {White,Green}      => {Red} 0.2     1.0000000  1.6666667
## [8] {White,Blue}       => {Red} 0.2     0.5000000  0.8333333
## [9] {White,Blue,Green} => {Red} 0.1     1.0000000  1.6666667
```

Doing it in R

Left-hand side contains Orange or white

Subset lhs

```
rules_3<-subset(rules1, lhs %in% c('Orange', 'White'))
inspect(rules_3)
```

```
##      lhs                  rhs      support confidence lift
## [1]  {Orange}          => {Red}    0.1     0.5000000  0.8333333
## [2]  {Orange}          => {White}  0.2     1.0000000  1.4285714
## [3]  {White}           => {Orange} 0.2     0.2857143  1.4285714
## [4]  {White}           => {Green}  0.2     0.2857143  1.4285714
## [5]  {White}           => {Blue}   0.4     0.5714286  0.9523810
## [6]  {White}           => {Red}    0.4     0.5714286  0.9523810
## [7]  {Red,Orange}      => {White}  0.1     1.0000000  1.4285714
## [8]  {White,Orange}    => {Red}    0.1     0.5000000  0.8333333
## [9]  {Red,White}       => {Orange} 0.1     0.2500000  1.2500000
```

Doing it in R

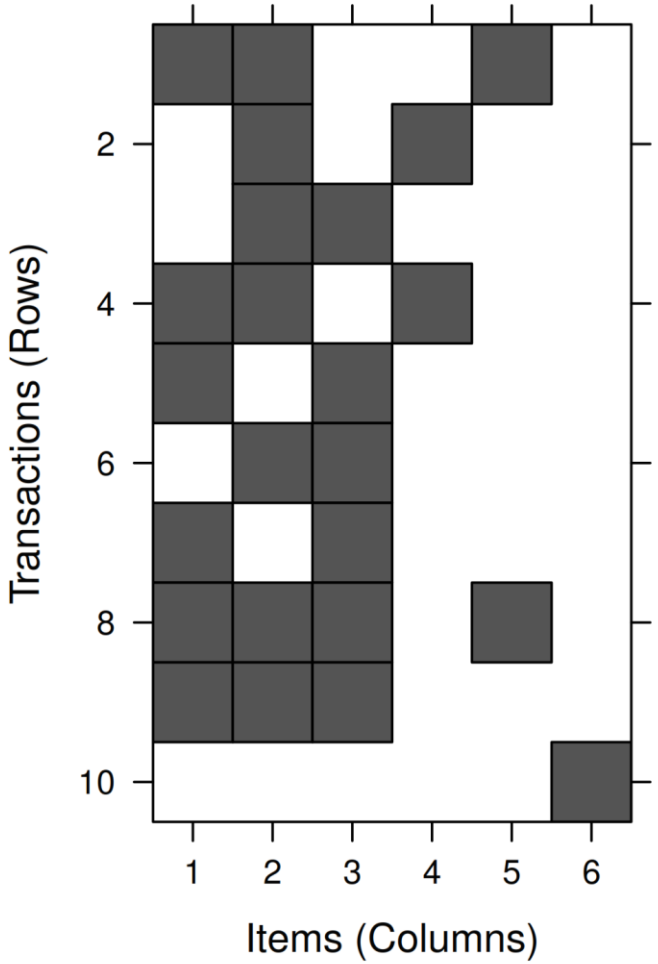
Take the rules that have both red and white

```
rules_3<-subset(rules1, lhs %ain% c('Red', 'White'))
inspect(rules_3)
```

```
##     lhs                    rhs      support confidence lift
## [1] {Red,White}         => {Orange} 0.1     0.25       1.2500000
## [2] {Red,White}         => {Green}  0.2     0.50       2.5000000
## [3] {Red,White}         => {Blue}   0.2     0.50       0.8333333
## [4] {Red,White,Green}   => {Blue}   0.1     0.50       0.8333333
## [5] {Red,White,Blue}    => {Green}  0.1     0.50       2.5000000
```

Doing it in R: Vizualization

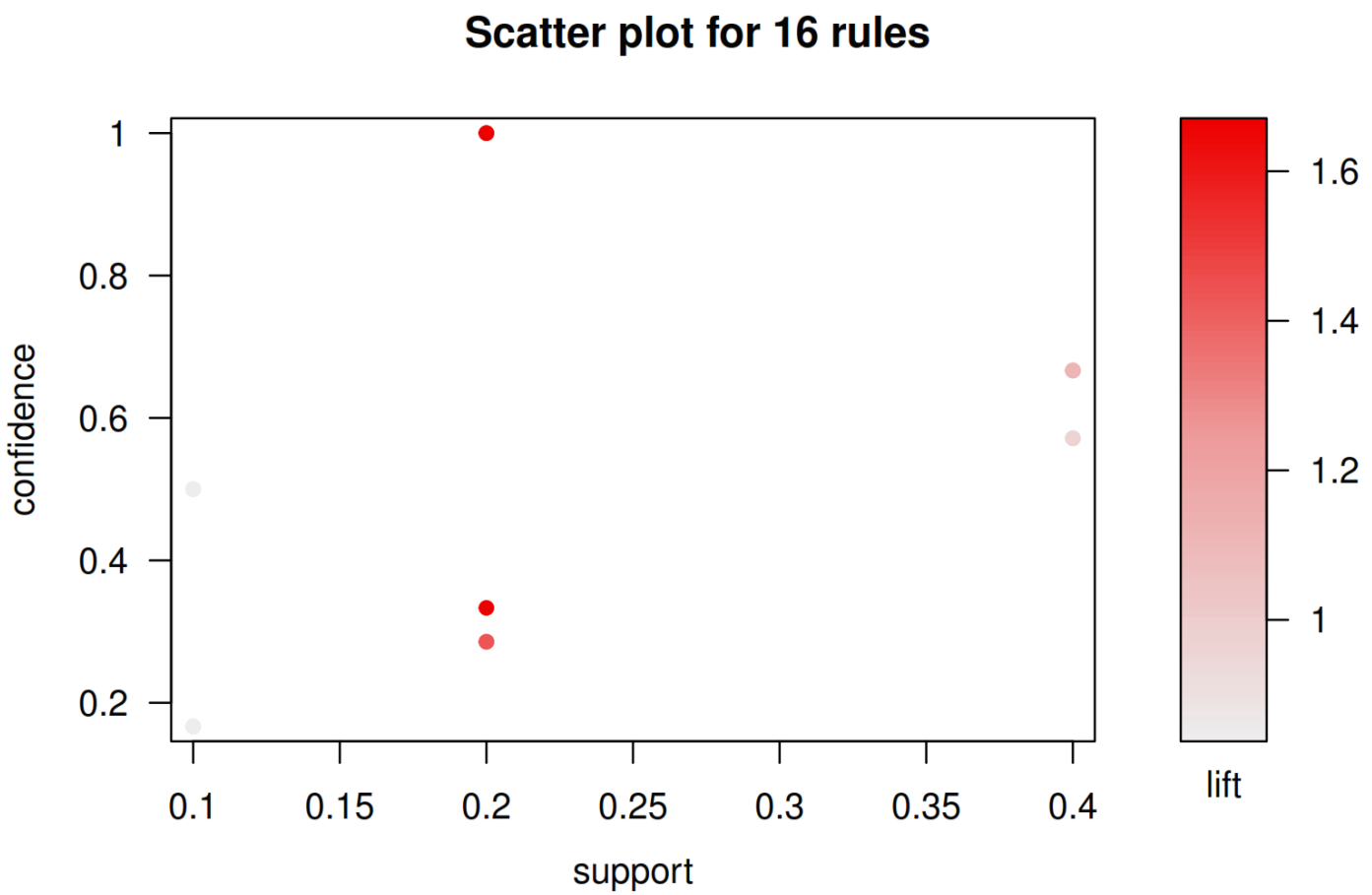
```
arules::image(txn)
```



arules in R

library(arulesViz)

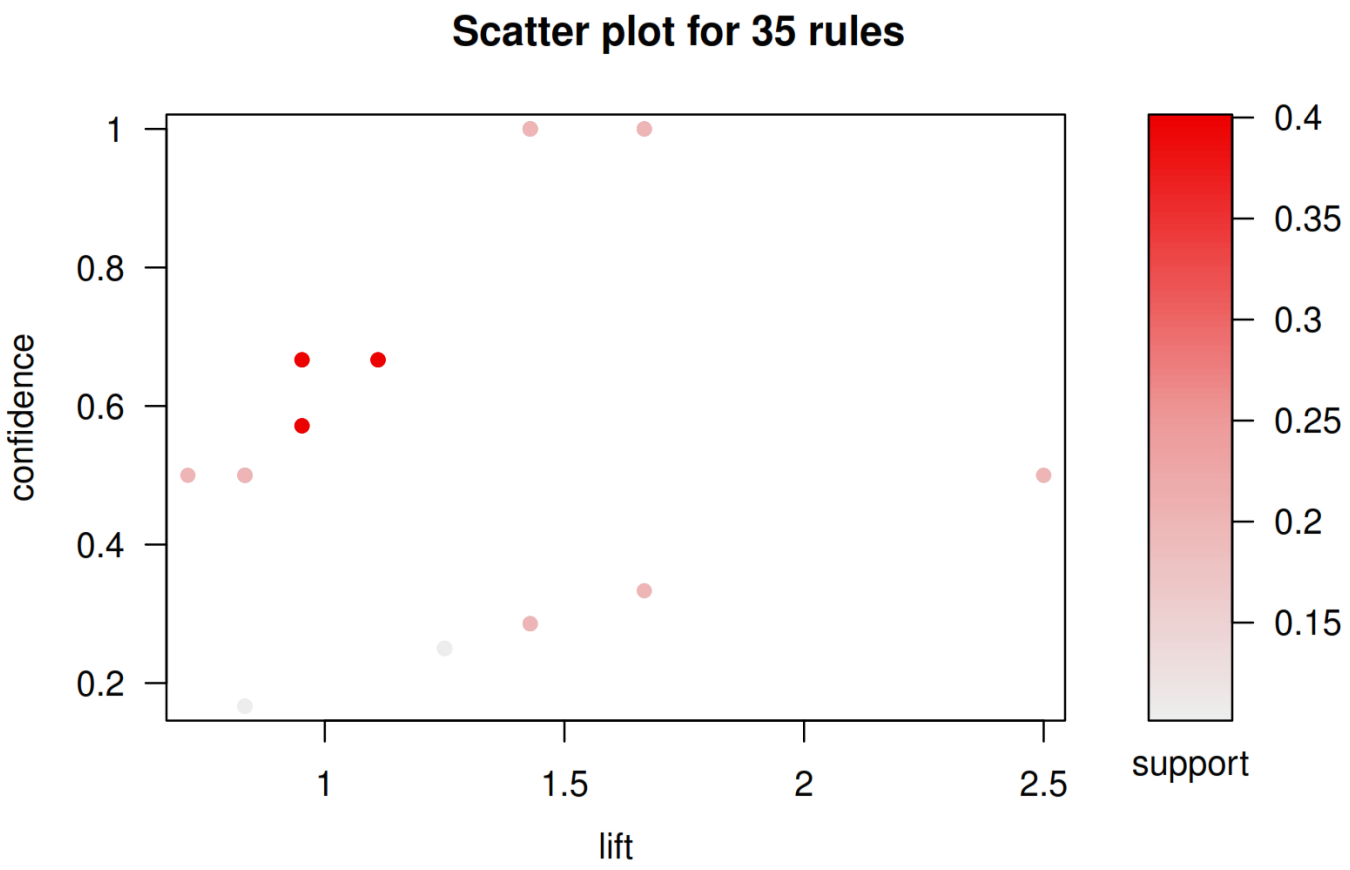
```
plot(rules, type="scatterplot")
```



arules in R

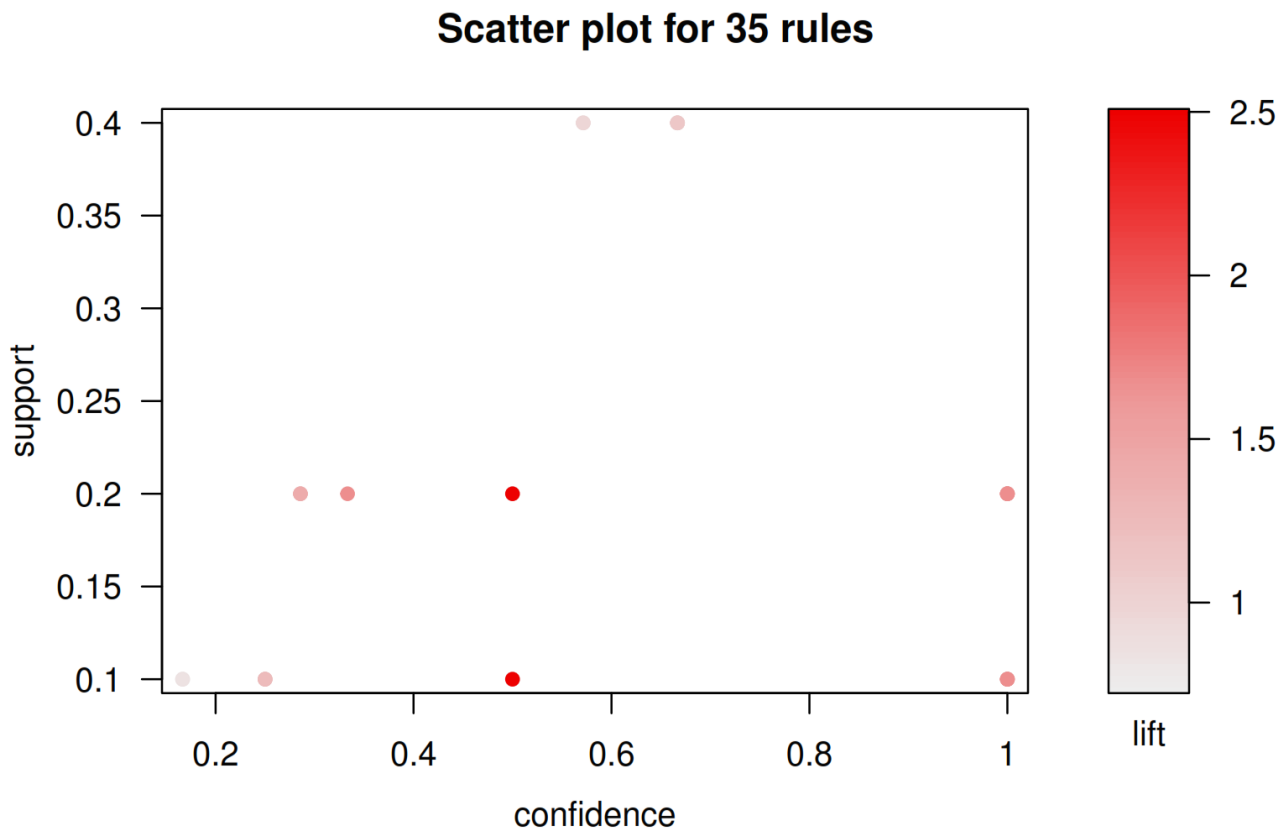
Change plotting controls

```
plot(rules1, type="scatterplot",
     measure=c("lift", "confidence"), shading="support")
```



arules in R

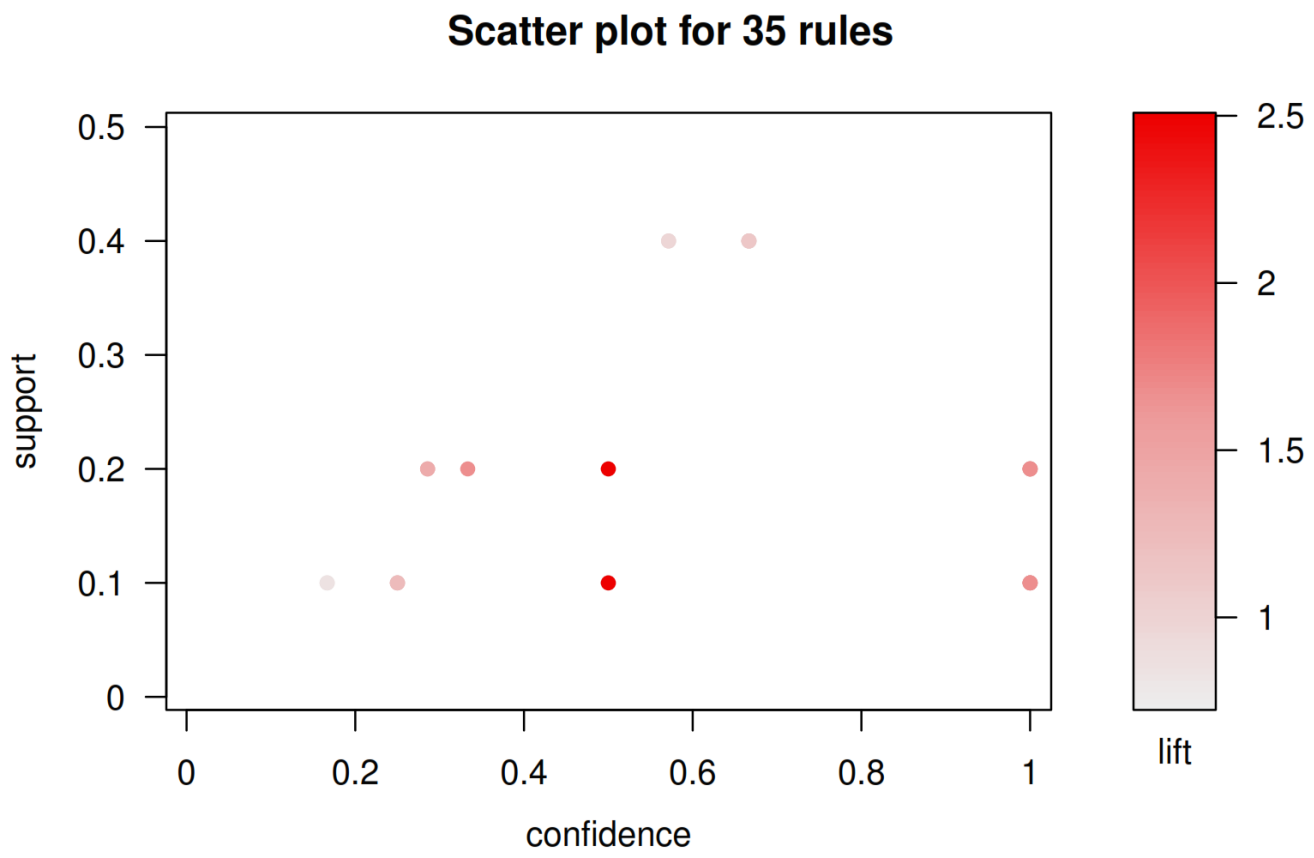
```
plot(rules1, type="scatterplot",
     measure=c("confidence", "support"), shading="lift")
```



arules in R

Not all the rules are displayed as we have limitations on x and y axes

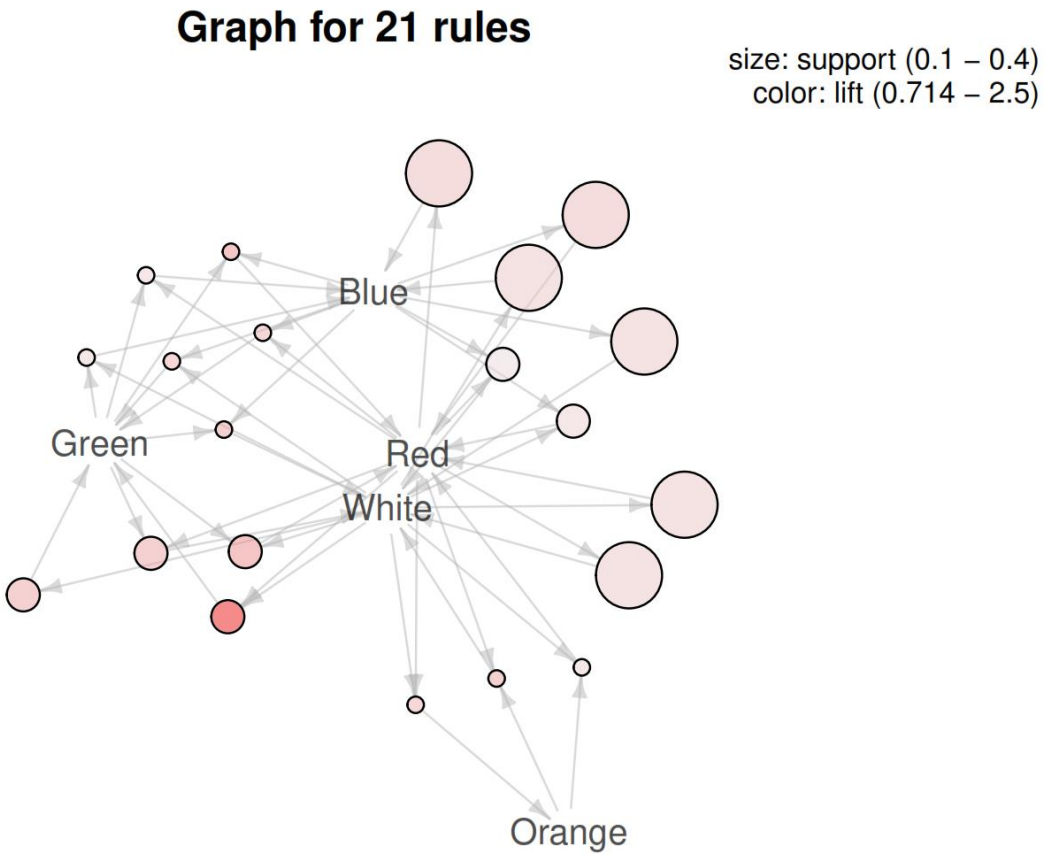
```
plot(rules1, type="scatterplot",
     measure=c("confidence", "support"), shading="lift",
     xlim=c(0.001,1), ylim=c(0.001,0.5))
```



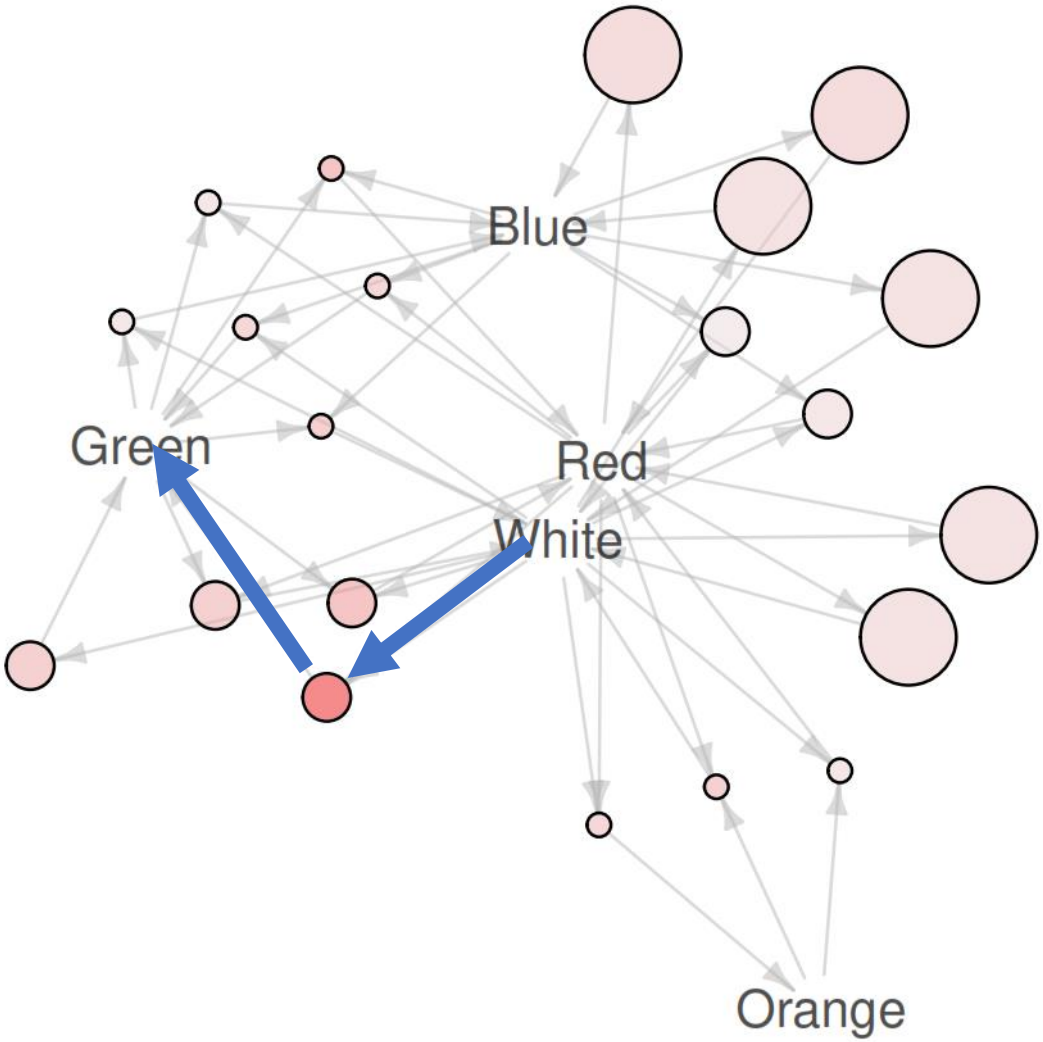
arules in R

Create a graph object

```
plot(rules1[10:30], method="graph")
```

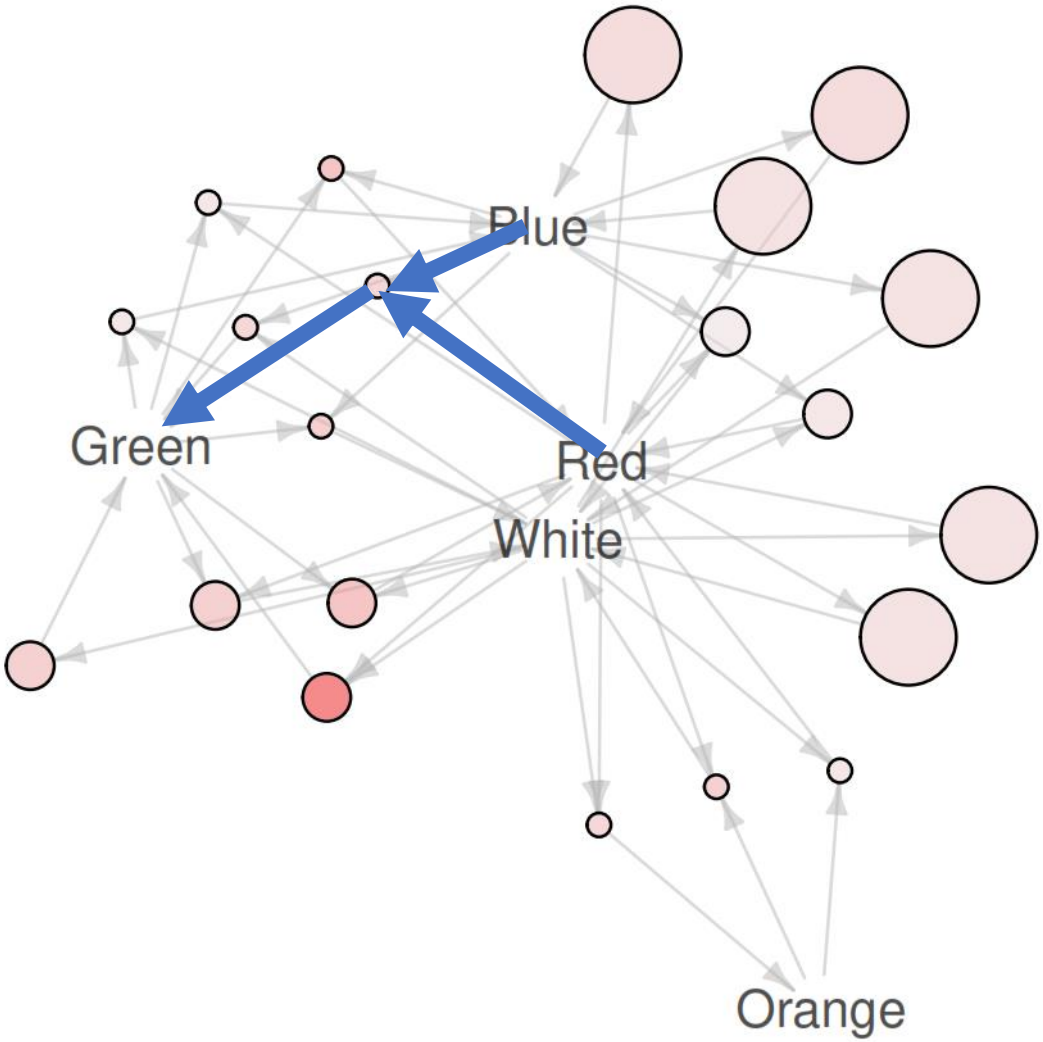


arules in R



IF White THEN Green

arules in R

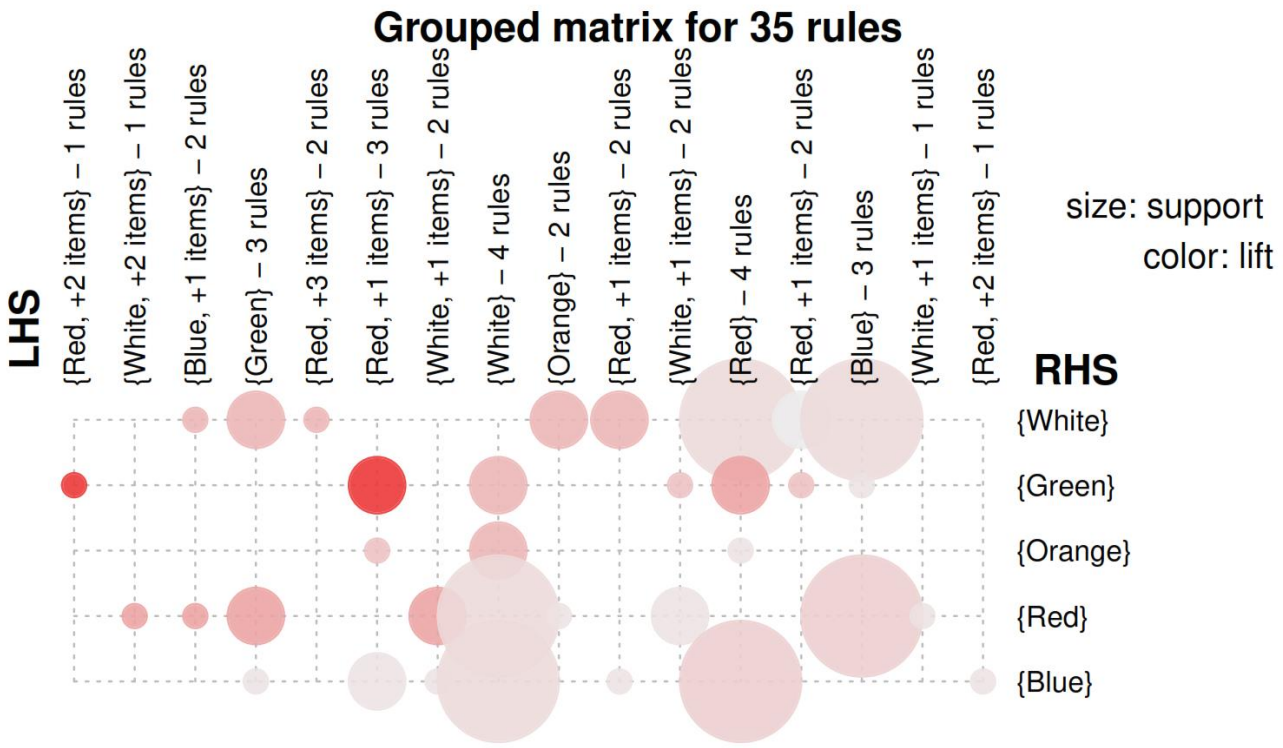


IF Blue and Red THEN Green

arules in R

Grouped matrix

```
plot(rules1, method="grouped")
```



What was done

- Around 20,000 posts are scrapped from the web,
- Posts come in rough, unstructured way. Algorithm is developed to structure them.

A variable for each “section”

JOB DESCRIPTION: N/A

JOB RESPONSIBILITIES:
- Search for news themes;
- Prepare texts and rewrite them;
- Post news on the agency's website.

REQUIRED QUALIFICATIONS:
- Graduate/ undergraduate education, preferably in Journalism;
- At least 6 months of work experience;
- Advanced PC user;
- Willingness to develop in journalism;
- Interest in business journalism;
- Communication skills;
- Initiative taking and active person;
- Creativity;
- Ability to work in a team;
- Perfect knowledge of Armenian and Russian languages.

APPLICATION PROCEDURES: To apply for this position, please send your CV to: arka@arminco.com , mentioning "Newswriter" in the subject line of the email.
Please clearly mention in your application letter that you learned of this job opportunity through Career Center and mention the URL of its website - www.careercenter.am, Thanks.

OPENING DATE: 19 November 2015

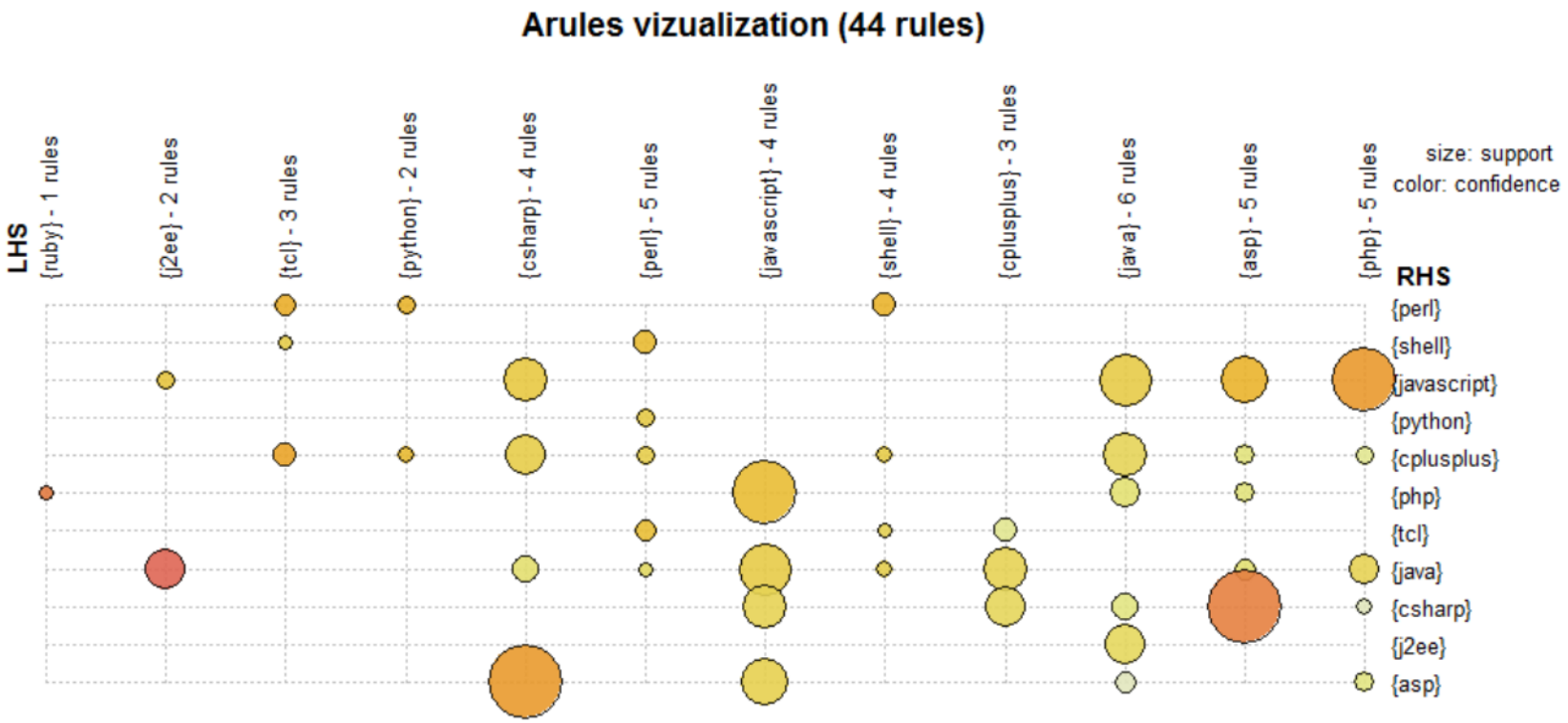
APPLICATION DEADLINE: 18 December 2015

ABOUT COMPANY: To learn more about the company, please visit: http://arka.am/en/about/.

Arules

- Association rules mining is used to analyse the co-occurrence of programming languages in a job post
- R package “”arules” and “arulesViz” are used for the analysis
- Analysis is done for IT jobs only

Visualizing the rules



Association Mining for Programming languages: C++

- Set of association rules is generated for top20 programming languages.
- Rules are sub seting with min support of 0.01 and min confidence of 0.1

One item on the left

```
> inspect(Cplusplus)
   lhs             rhs       support    confidence lift
7  {cplusplus} => {tcl}     0.02238254 0.1260417  3.140006
36 {cplusplus} => {csharp}  0.04236034 0.2385417  1.708022
41 {cplusplus} => {java}    0.04624491 0.2604167  1.417737
```

Two items on the left

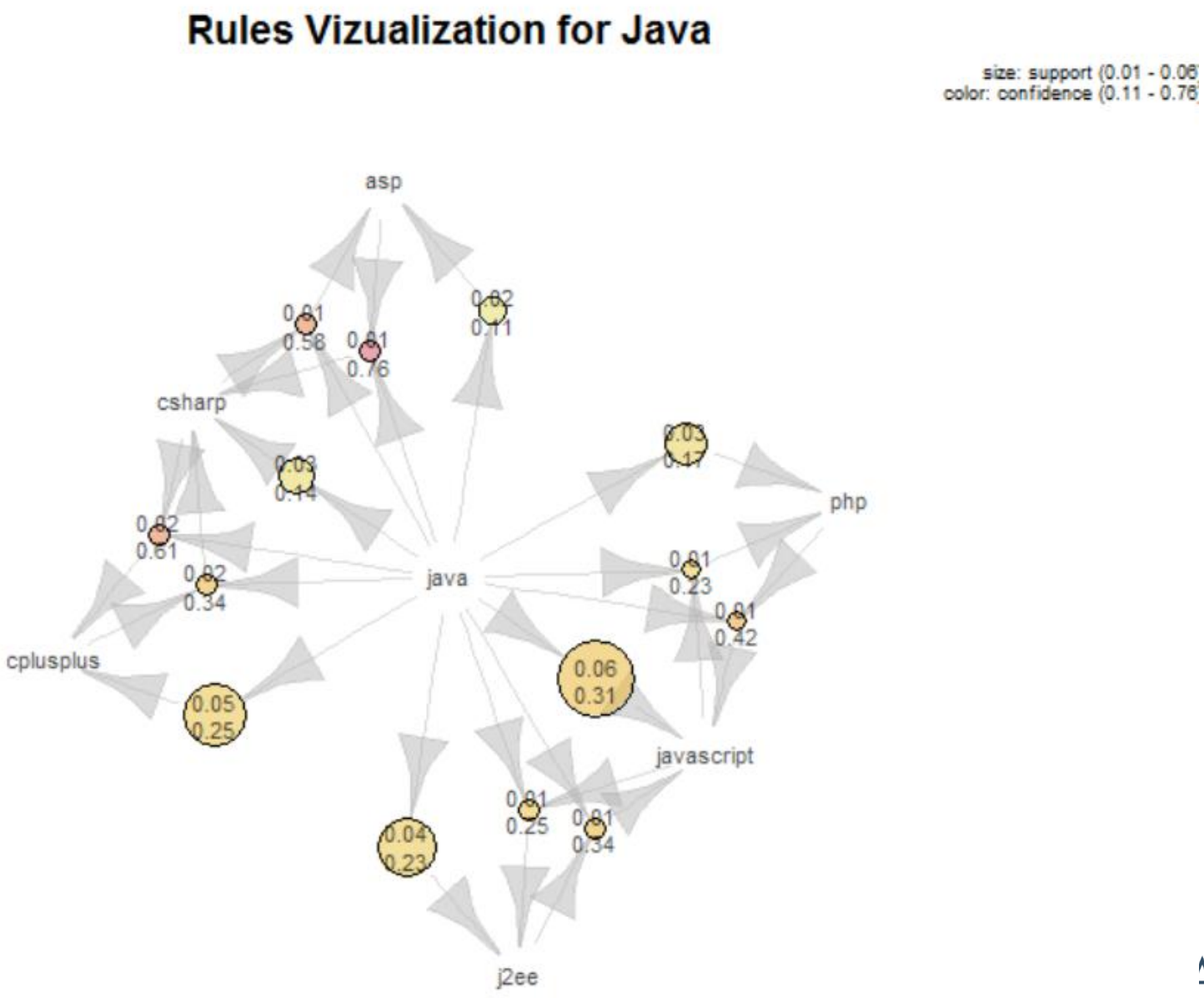
```
> inspect(Cplusplus3)
   lhs                     rhs       support    confidence lift
11 {asp,cplusplus}     => {csharp} 0.01405845 0.8260870  5.915001
12 {csharp,cplusplus}  => {asp}    0.01405845 0.3318777  2.970416
19 {csharp,cplusplus}  => {java}   0.01572327 0.3711790  2.020739
21 {cplusplus,java}    => {csharp} 0.01572327 0.3400000  2.434490
```

Association Mining for Programming languages: Java

```
> inspect(java)
   lhs        rhs            support    confidence lift
12 {java} => {j2ee}       0.04291528 0.2336354  4.9922262
25 {java} => {php}        0.03107658 0.1691843  1.4448819
32 {java} => {asp}        0.01979282 0.1077543  0.9644365
38 {java} => {csharp}     0.02589715 0.1409869  1.0095036
42 {java} => {cplusplus}  0.04624491 0.2517623  1.4177367
43 {java} => {javascript} 0.05734369 0.3121853  1.7256377
```

```
> inspect(java3)
   lhs                    rhs            support    confidence lift
1  {j2ee,java}         => {javascript} 0.01442841 0.3362069  1.858420
3  {java,javascript}   => {j2ee}       0.01442841 0.2516129  5.376361
7  {java,php}          => {javascript} 0.01294858 0.4166667  2.303170
9  {java,javascript}   => {php}        0.01294858 0.2258065  1.928451
14 {asp,java}          => {csharp}     0.01498335 0.7570093  5.420387
15 {csharp,java}       => {asp}        0.01498335 0.5785714  5.178406
20 {csharp,java}       => {cplusplus}  0.01572327 0.6071429  3.418973
21 {cplusplus,java}    => {csharp}     0.01572327 0.3400000  2.434490
```

Rules visualization: Java (all rules)



Rules Vizualization for Javascript

size: support (0.01 - 0.07)
color: confidence (0.16 - 0.9)