MovieLens genre associations

Using associative rules learning to analyze genres co-occurrences

Anton Antonov
Mathematica for Prediction blog
Mathematica for Prediction project at GitHub
August 2013
September 2013
October 2013

Introduction

In this document we are going to discuss the application of associative rules learning to the genres assigned to movies in a given catalog (taken from MovieLens, see below). We try to see how often dramas are also comedies, and thrillers are also action movies. Associative rules learning is also known as market basket analysis and we can conveniently see each movie as a basket of genres. For a concise introduction into associative rules learning see [3], a more comprehensive one is given in [2].

This document is also a guide to using the *Mathematica* package AprioriAlgorithm.m provided at GitHub: see https://github.com/antononcube/MathematicaForPrediction .

In the first section we discuss how the catalog was obtained and how we turned the file into movie-genre data. In the second section we do simple, descriptive statistics analysis of that data (how many movies per genre, how many movies per year). The movie set is poorly interconnected because around 40% of the movies have only one genre. We use this set in this document since it is publicly available and the presented results can be reproduced.

The third and fourth sections give brief theoretical background on Apriori and association rules. Several measures are defined: support, confidence, lift, leverage, and conviction.

In the seventh section we convert the movie-genre data into a list of integer lists and apply the so-called Apriori algorithm, [1]. The eighth section shows two graphic interfaces for browsing and analyzing the associations found. A general computation sequence is also given for answering queries over Apriori's results.

The Apriori algorithm is based on the so-called Rymon trees. Although in this document we only outline how Apriori works and it is implemented, the package used contains a Rymon tree generator and we give in the final section examples of Rymon trees with brief explanations.

Load the data

The movie data was taken from the page MovieLens Data Sets (http://www.grouplens.org/taxonomy/term/14) of the site GroupLens Research (http://www.grouplens.org). More precisely the data set named “MovieLens 10M Data Set” was taken.

After downloading the data set and following the data format descriptions in README for the file “movies.dat” we can get the movie data with the following commands:

```
lines = ReadList["~/MovieLens/ml-10M100K/movies.dat", String];
lines // Length
```

```
10 681
```

```
movies = Map[StringSplit[#, {"::", "|"}] &, lines];
movies // Length
```

```
10 681
```

Here is a sample of the movie data, `movies`, obtained with the last command:

id	title (year)	genres		
6154	Deliver Us from Eva (2003)	Comedy	Romance	
3602	G. I. Blues (1960)	Comedy	Musical	Romance
4217	4 Little Girls (1997)	Documentary		
2123	All Dogs Go to Heaven (1989)	Animation	Children	
6083	Hammett (1982)	Crime	Mystery	
31600	Deceivers, The (1988)	Adventure	Drama	Thriller
44709	Akeelah and the Bee (2006)	Drama		
33646	Longest Yard, The (2005)	Comedy	Drama	
3944	Bootmen (2000)	Comedy	Drama	
3859	Eyes of Tammy Faye, The (2000)	Documentary		

Let us also separate the years from the titles with these commands:

```
movies[[All, 2]] =
  StringCases[#, (t : ___) ~~ ("(" ~~ y : NumberString ~~ ")" ~~ ___) :>
      {t, ToExpression[y]}][[1]] & /@ movies[[All, 2]];
movies = Flatten /@ movies;
```

Here is a sample of the movie data, `movies`, that is going to be used for the rest of the computations:

id	title	year	genres		
54094	Driving Lessons	2006	Comedy	Drama	
50068	Letters from Iwo Jima	2006	Drama	War	
43460	Tristram Shandy: A Cock and Bull Story	2005	Comedy	Drama	
2739	Color Purple, The	1985	Drama		
2627	Endurance	1999	Documentary	Drama	
917	Little Princess, The	1939	Children	Drama	
52606	Big Nothing	2006	Comedy	Crime	T
8302	Front Page, The	1931	Comedy	Drama	F
51927	Dead Silence	2007	Horror	Mystery	T
6313	Sibling Rivalry	1990	Comedy		

Preliminary statistics

The genres

Here are all the movie genres with their number of appearances:

Drama	5339
Comedy	3703
Thriller	1706
Romance	1685
Action	1473
Crime	1118
Adventure	1025
Horror	1013
Sci-Fi	754
Fantasy	543
Children	528
War	511
Mystery	509
Documentary	482
Musical	436
Animation	286
Western	275
Film-Noir	148
IMAX	29
(no genres listed)	1

Note that “IMAX” and “(no genres listed)” are not genres per se, but we are not going to remove them.

Number of genres per movie

Let us compute different descriptive statistics for the number of genres the movies have.

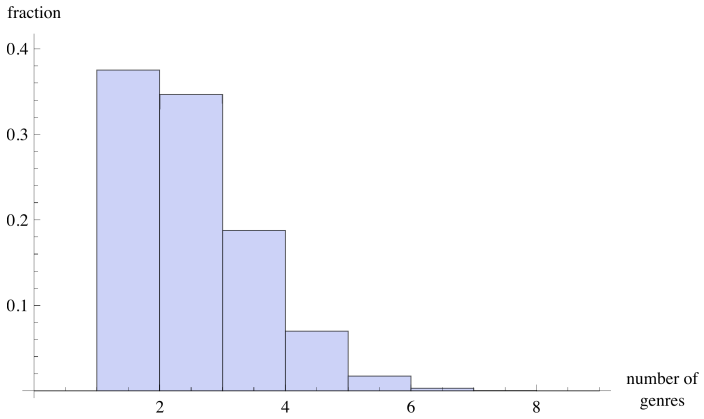
We find the number of genres for each movie with the command:

```
ngenres = Map[Length[Take[#, {4, -1}]] &, movies];
```

Here is a table with different statistics for the number of genres:

Min	Max	Mean	Median	StandardDeviation
1	8	2.01891	2	1.02917

Here is a histogram with the distribution of the number of genres:



Let us look at the outliers -- the movies with more than 6 genres. We find these movies with this command:

```
pos = Flatten[Position[ngenres, Max[ngenres] | Max[ngenres] - 1]]
{2903, 9571, 9827, 10 094}
```

and here is a table of their data:

id	title	year	genres			
2987	Who Framed Roger Rabbit?	1988	Adventure	Animation	Children	Com
46948	Monster House	2006	Adventure	Animation	Children	Com
51709	Host, The (Gwoemul)	2006	Action	Adventure	Comedy	Dra
56152	Enchanted	2007	Adventure	Animation	Children	Com

Since ≈ 40% of the movies have only one genre it is good to know what is the breakdown of the one-genre movies across the genres. As it can be seen in the following table nearly one half of the one genre movies are dramas, and nearly one fourth are comedies. The remaining one fourth is made by all the other genres.

genre	number of movies with only that genre	total number of movies containing the genre
Drama	1817	5339
Comedy	1047	3703
Documentary	350	482
Horror	267	1013
Thriller	127	1706
Western	92	275
Action	82	1473
Sci-Fi	54	754
Romance	38	1685
Adventure	32	1025
Crime	26	1118
Musical	26	436
War	19	511
Film-Noir	12	148
Mystery	6	509
Children	4	528
Animation	3	286
Fantasy	2	543
IMAX	1	29
(no genres listed)	1	1

We can say that the movie set is poorly inter-connected since nearly 40% of the movies have only one genre.

Release years distribution

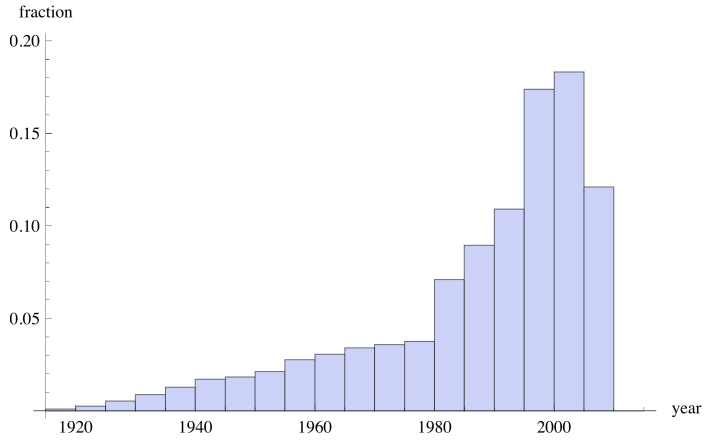
It is also a good idea to see the distribution of the movies across the release years.

```
years = movies[[All, 3]];
```

Here is a table with different statistics for the movie release years:

Min	Max	Mean	Median	StandardDeviation
1915	2008	1986.95	1994	19.03

Here is a histogram with the distribution of the movies across the release years:



Apriori algorithm

For the full theoretical justification and description of the Apriori algorithm we refer to [1]. In this section we are going to give the necessary definitions and explanations in order to use Apriori for discovery of rules from data.

Definition 1: Let I be a set of items, and T is a set of sets of items called *transactions set*. Each element of T is a transaction of items (or a basket of items that are purchased). □

Definition 2: The *support count* of a subset $p \in \mathcal{P}(I)$ is defined to be

$$suppcount_T(p) := |\{t : t \in T \bigwedge p \subset t\}|.$$

□

(With $\mathcal{P}(I)$ we denote the set of all subsets of I.)

Definition 3: The *support* of a set of items p is defined as

$$supp_T(p) := \frac{|\{t : t \in T \bigwedge p \subset t\}|}{|T|}.$$

□

Definition 4: An item set $p \in \mathcal{P}(I)$ is μ-frequent relatively to the transaction set T if $supp_T(p) \geq \mu$. □

The Apriori algorithm is used to find all μ-frequent sets for a given set of transactions.

The Apriori algorithm is based on the observation that if $suppcount_T(p) \geq m$, $m \in \mathbb{N}$ for $p \in \mathcal{P}(I)$, then for any subset s of p it is also true that $suppcount_T(s) \geq m$. Here is how we can use this observation: suppose we a want to find which pairs of items can be found in more than n baskets in T. We first find all items $i \in I$ for which $suppcount_T(i) > n$, and then from that subset of I we form all pairs $\{i_1, i_2\}$, $i_1 \neq i_2$, and select from these pairs the ones for which $suppcount_T(\{i_1, i_2\}) \geq n$.

Apriori has the two steps we just described. Apriori first finds all μ-frequent one element sets. At each step k, Apriori first generates candidate sets of k elements using the already

found μ-frequent sets of $k-1$ elements, then those which are μ-frequent are selected from the k-element candidates. The candidate generation is done with Rymon trees. (See [1] and the last section of the document.) Apriori stops at the step for which no frequent sets are found, but obviously it can be stopped before that.

Association rules

In this section we define what association rules are and different measures for extracting them from μ-frequent sets.

Definition 5: Each μ-frequent set K can be partitioned into two dis-joined sets, an antecedent A and a consequent B, written as

$A \Rightarrow B$.

The support of the rule $A \Rightarrow B$ is the same as the support of the set $A \bigcup B$.

$supp_T(A \Rightarrow B) = supp_T(A \bigcup B)$.

□

In order to discuss the rules in terms of probabilities given a set of items A we associate with A the event E_A which is "A is a subset of a transaction in T" and $P(E_A)$ is estimated by $supp_T A$, $P(E_A) \approx supp_T A$. For shorthand we use $P(A) := P(E_A)$.

Note that with the probability notation we have

$P(A \Rightarrow B) = P(A \bigcap B)$.

Definition 6: *Confidence* of a rule is

$$conf_T(A \Rightarrow B) := \frac{supp_T(A \Rightarrow B)}{supp_T(A)} \approx \frac{P(A \bigcap B)}{P(A)} = P(B / A).$$

□

Definition 7: *Lift* of a rule is defined as

$$lift_T(A \Rightarrow B) := \frac{conf_T(A \Rightarrow B)}{supp_T(B)} = \frac{supp_T(A \bigcup B)}{supp_T(A)\, supp_T(B)} \approx \frac{P(A \bigcap B)}{P(A)\, P(B)} = \frac{P(B / A)}{P(B)}.$$

□

Definition 8: *Leverage* of a rule is defined as

$leverage_T(A \Rightarrow B) := supp_T(A \Rightarrow B) - supp_T(A)\, supp_T(B) \approx P(A \bigcap B) - P(A)\, P(B)$.

□

Definition 9: *Conviction* of a rule is defined as

$$conv_T(A \Rightarrow B) := \frac{1 - supp_T(B)}{1 - conf_T(A \Rightarrow B)} \approx \frac{P(A) - P(A)\, P(B)}{P(A) - P(A \bigcap B)} = \frac{1 - P(B)}{1 - P(B / A)}.$$

□

The following tables summarize the definitions and properties of the measures.

measure	definition	interpretation
support	$\text{supp}_T(A \Rightarrow B)$	$P(A \bigcap B)$
confidence	$\frac{\text{supp}_T[A \Rightarrow B]}{\text{supp}_T[A]}$	$P(B \,/\, A)$
lift	$\frac{\text{conf}_T[A \Rightarrow B]}{\text{supp}_T[B]}$	$\frac{P(B \,/\, A)}{P(B)}$
leverage	$\text{supp}_T[A \Rightarrow B] - \text{supp}_T[A]\ \text{supp}_T[B]$	$P(A \bigcap B) - P(A)\ P(B)$
conviction	$\frac{1-\text{supp}_T[B]}{1-\text{conf}_T[A \Rightarrow B]}$	$\frac{1-P(B)}{1-P(B \,/\, A)}$

measure	min value, incompatibility	value at independance	max value, logical rule
support	0	$\text{supp}_T(A)\ \text{supp}_T(B)$	$\text{supp}_T(A)$
confidence	0	$\text{supp}_T(B)$	1
lift	0	1	$\frac{1}{\text{supp}_T(B)}$
leverage	$-\text{supp}_T(A)\ \text{supp}_T(B)$	0	$\text{supp}_T(A)\ (1 - \text{supp}_T(B))$
conviction	$1 - \text{supp}_T(B)$	1	∞

A much more comprehensive comparison and discussion of a collection of 20 measures is given in [6].

Load the Apriori algorithm

Here is how we load the package [4] which has implementations of the Apriori algorithm and related functions

```
Get["~/MathFiles/MathematicaForPrediction/AprioriAlgorithm.m"]
```

Association rules finding (running the algorithm)

With this command for each movie we make a basket of genres.

```
baskets = Take[#, {4, -1}] & /@ movies;
baskets // Length
```

```
10 681
```

With the following commands we find all the μ-frequent sets for $\mu = 2.5\,\%$ of 5 elements or less. The items in the baskets can be of any type. That is why in this implementation two lists of rules are created (i) one mapping the items into integer ID's and (ii) the other mapping the integer ID's to the items. The function `AprioriApplication` returns the fre-

quent sets as lists of integers together with these two lists of rules.

```
μ = 0.0025; mt = 5;
Print["Number of baskets corresponding to μ: ",
 Length[baskets] * μ]
t =
 AbsoluteTiming[
  {aprioriRes, itemToIDRules, idToItemRules} =
    AprioriApplication[baskets, μ, "MaxNumberOfItems" → mt];
 ]
```

```
Number of baskets corresponding to μ: 26.7025
{8.794782, Null}
```

The output of

```
Length /@ aprioriRes
```

```
{19, 90, 84, 7}
```

is interpreted in the following table:

Number of items	Number of μ−frequent sets
1	19
2	90
3	84
4	7

Here we convert the baskets of genres into baskets of ID's.

```
dataWithIDs = baskets /. itemToIDRules;
```

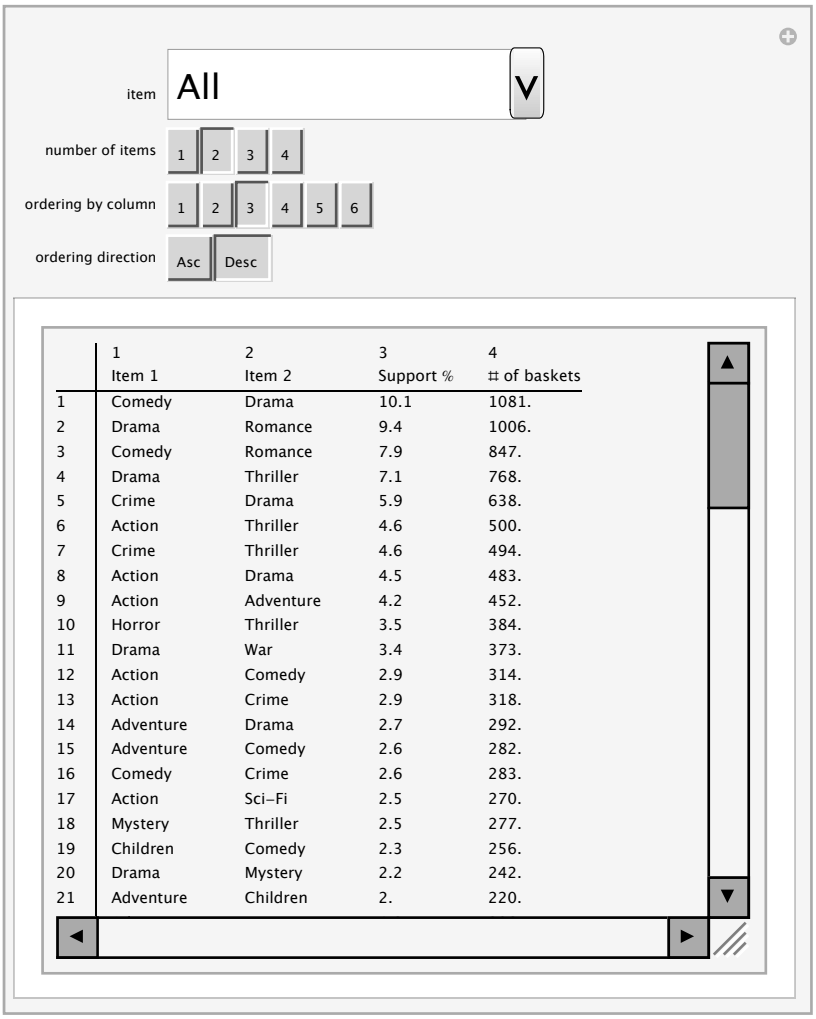
The data structure `dataWithIDs` is used in the interfaces in the next section.

Interfaces

In this section we present couple of interfaces for browsing and investigating the frequent association sets found in the previous section and the association rules derived from them.

Interface for browsing of frequent sets

This interface for browsing the found μ-frequent sets allows their filtering to contain a particular item or all items for a selected number of items. The ordering of the filtered sets can be specified by selection of column and direction. The items in each row of the tabulated result is are ordered lexicographically.

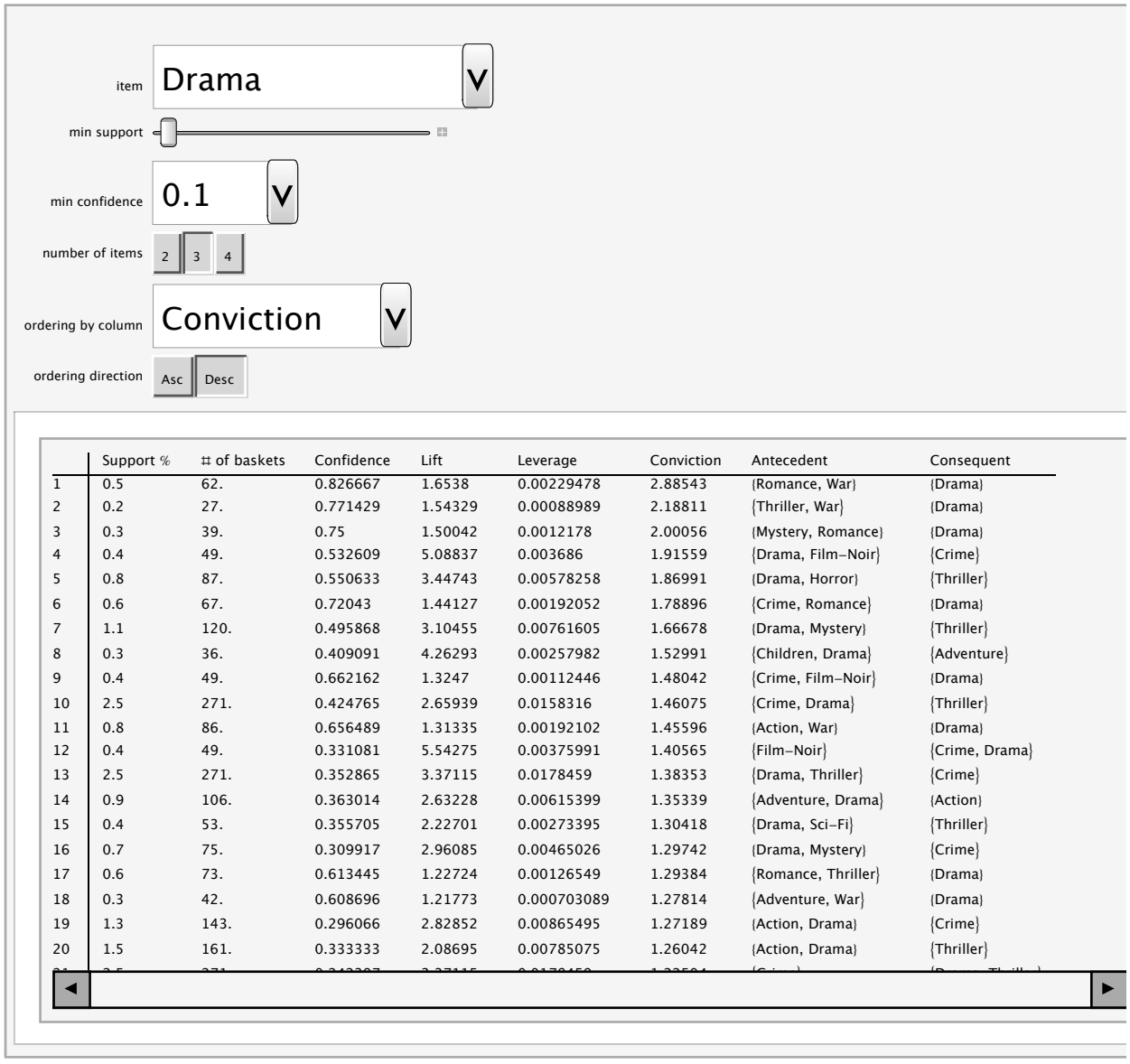


	1 Item 1	2 Item 2	3 Support %	4 # of baskets
1	Comedy	Drama	10.1	1081.
2	Drama	Romance	9.4	1006.
3	Comedy	Romance	7.9	847.
4	Drama	Thriller	7.1	768.
5	Crime	Drama	5.9	638.
6	Action	Thriller	4.6	500.
7	Crime	Thriller	4.6	494.
8	Action	Drama	4.5	483.
9	Action	Adventure	4.2	452.
10	Horror	Thriller	3.5	384.
11	Drama	War	3.4	373.
12	Action	Comedy	2.9	314.
13	Action	Crime	2.9	318.
14	Adventure	Drama	2.7	292.
15	Adventure	Comedy	2.6	282.
16	Comedy	Crime	2.6	283.
17	Action	Sci-Fi	2.5	270.
18	Mystery	Thriller	2.5	277.
19	Children	Comedy	2.3	256.
20	Drama	Mystery	2.2	242.
21	Adventure	Children	2.	220.

Interface for browsing of association rules

This interface is similar to the previous one but now the frequent sets are converted to rules which are generated and filtered according to specified minimal support and minimal

confidence.



	Support %	# of baskets	Confidence	Lift	Leverage	Conviction	Antecedent	Consequent
1	0.5	62.	0.826667	1.6538	0.00229478	2.88543	{Romance, War}	{Drama}
2	0.2	27.	0.771429	1.54329	0.00088989	2.18811	{Thriller, War}	{Drama}
3	0.3	39.	0.75	1.50042	0.0012178	2.00056	{Mystery, Romance}	{Drama}
4	0.4	49.	0.532609	5.08837	0.003686	1.91559	{Drama, Film-Noir}	{Crime}
5	0.8	87.	0.550633	3.44743	0.00578258	1.86991	{Drama, Horror}	{Thriller}
6	0.6	67.	0.72043	1.44127	0.00192052	1.78896	{Crime, Romance}	{Drama}
7	1.1	120.	0.495868	3.10455	0.00761605	1.66678	{Drama, Mystery}	{Thriller}
8	0.3	36.	0.409091	4.26293	0.00257982	1.52991	{Children, Drama}	{Adventure}
9	0.4	49.	0.662162	1.3247	0.00112446	1.48042	{Crime, Film-Noir}	{Drama}
10	2.5	271.	0.424765	2.65939	0.0158316	1.46075	{Crime, Drama}	{Thriller}
11	0.8	86.	0.656489	1.31335	0.00192102	1.45596	{Action, War}	{Drama}
12	0.4	49.	0.331081	5.54275	0.00375991	1.40565	{Film-Noir}	{Crime, Drama}
13	2.5	271.	0.352865	3.37115	0.0178459	1.38353	{Drama, Thriller}	{Crime}
14	0.9	106.	0.363014	2.63228	0.00615399	1.35339	{Adventure, Drama}	{Action}
15	0.4	53.	0.355705	2.22701	0.00273395	1.30418	{Drama, Sci-Fi}	{Thriller}
16	0.7	75.	0.309917	2.96085	0.00465026	1.29742	{Drama, Mystery}	{Crime}
17	0.6	73.	0.613445	1.22724	0.00126549	1.29384	{Romance, Thriller}	{Drama}
18	0.3	42.	0.608696	1.21773	0.000703089	1.27814	{Adventure, War}	{Drama}
19	1.3	143.	0.296066	2.82852	0.00865495	1.27189	{Action, Drama}	{Crime}
20	1.5	161.	0.333333	2.08695	0.00785075	1.26042	{Action, Drama}	{Thriller}

General queries

Suppose we want an answer for the following query:

Give all movies for which "Drama" is the consequent,
that have confidence 70 % and the support is at least 3 %. (1)

In this sub-section we are going to describe the steps for computing the answer of this question.

The package function `AssociationRules` is very useful for the computation of the answer. The function has two signatures:

```
AssociationRules[T : {{_Integer ...} ...},
 basket : {_Integer ...}, minConfidence_?NumberQ]
```
(2)

and

```
AssociationRules[T : {{_Integer ..} ..},
 aprioriResRecs : {{_Integer ..} ...},
 minConfidence_?NumberQ, minSupport_?NumberQ]
```
(3)

The second signature takes the association sets from `assocItemSets` that have support at least `minSupport` and finds for them the association rules that have confidence at least `minConfidence`. Note that the arguments corresponding to the transactions and the association sets have to be lists of integers.

The returned result of (3) has the form

```
{support_?NumberQ, confindence_?NumberQ,
 lift_?NumberQ, leverage_?NumberQ,
 conviction_?NumberQ, antecedent : {_Integer ..},
 consequent : {_Integer ..}}
```
(4)

Here is an example of finding all the rules with support 5% and confidence 0.4:

```
AssociationRules[dataWithIDs, aprioriRes[[2]], 0.4, 0.03] /.
 idToItemRules
```

```
{{0.0423181, 0.440976, 3.1976, 0.0290838,
  1.54214, {Adventure}, {Action}}, {0.0792997, 0.502671,
  1.44991, 0.0246069, 1.31364, {Romance}, {Comedy}},
 {0.0597322, 0.570662, 1.14164, 0.00741101, 1.16491,
  {Crime}, {Drama}}, {0.0462504, 0.44186, 2.76642,
  0.0295319, 1.5055, {Crime}, {Thriller}},
 {0.0941859, 0.597033, 1.1944, 0.0153297, 1.24114,
  {Romance}, {Drama}}, {0.0719034, 0.450176, 0.900605,
  -0.00793563, 0.909637, {Thriller}, {Drama}},
 {0.0349218, 0.729941, 1.46029, 0.0110076, 1.85197, {War}, {Drama}}}
```

Using the result of invocation of (3) we can easily answer the request (1) by using the function `Cases`:

```
Cases[%, {__, _List, {___, "Drama"}}, ∞]
```

```
{{0.0597322, 0.570662, 1.14164, 0.00741101, 1.16491,
  {Crime}, {Drama}}, {0.0941859, 0.597033, 1.1944, 0.0153297,
  1.24114, {Romance}, {Drama}}, {0.0719034, 0.450176,
  0.900605, -0.00793563, 0.909637, {Thriller}, {Drama}},
 {0.0349218, 0.729941, 1.46029, 0.0110076, 1.85197, {War}, {Drama}}}
```

We can define a function that gets association rules for a particular item using the results of AprioriApplication:

```
Clear[ItemRules];
ItemRules[item_, minConfidence_?NumberQ, minSupport_?NumberQ] :=
  Block[
   {aprioriRes = Cases[#, {___, item /. itemToIDRules, ___}, ∞] & /@
      aprioriRes, t},
   t = AssociationRules[dataWithIDs, aprioriRes[[#]], minConfidence,
     minSupport] & /@ Range[2, Length[aprioriRes]];
   DeleteCases[t /. idToItemRules, {}, 2]
  ];
```

Here is an example invocation

```
t = ItemRules["Thriller", 0.5, 0.01]
```

```
{{{0.0259339, 0.544204, 3.40718,
    0.0183224, 1.84354, {Mystery}, {Thriller}}},
 {{0.0107668, 0.766667, 5.55924, 0.00883005,
    3.69468, {Adventure, Thriller}, {Action}},
  {0.015448, 0.518868, 3.24855, 0.0106926, 1.74646,
    {Action, Crime}, {Thriller}}, {0.0253722, 0.548583,
    1.09747, 0.00225348, 1.10793, {Crime, Thriller}, {Drama}}}}
```

Rymon tree generation

In this section we formulate and give examples of Rymon trees.

Rymon trees are used to enumerate all subsets of a given set. Each element of the set is replaced with an integer. The integers are between 1 and the number of elements in the set, n.

The root of the tree is the empty set. A vertex K of the tree is a subset $\{i_{p_1}, i_{p_2}, \ldots, i_{p_k}\}$ with $i_{p_1} < i_{p_2} < \ldots < i_{p_k}$; the vertex K has $n - i_{p_k}$ children obtained with $K \bigcup \{j\}$, where $j \in (i_{p_k}, n]$.

Rymon tree properties are at the foundation of the Apriori algorithm.

Remark 1: The Apriori algorithm implementation of the package used in this document, [4], does not use Rymon tree generation directly since that would be memory expensive. We can describe Apriori as a way to derive and traverse a subtree of the Rymon tree of a given set.

Remark 2: The description in this section duplicates the one given in [5]. We describe Rymon trees here since the package used in the document (see [4]) has definitions for Rymon tree generation and visualization. For more mathematical and algorithmic details see the chapter “DATA MINING ALGORITHMS II: FREQUENT ITEM SETS” in [1].

Examples

The function `RymonTree` can be used to generate Rymon trees of a given order:

```
? RymonTree
```

RymonTree[numberOfItems] gives the Rymon tree for numberOfItems.

Here are examples of using the function RymonTree defined above :

```
RymonTree[2]
```

```
{{}, {{1}, {{1, 2}}}, {{2}}}
```

```
RymonTree[3]
```

```
{{}, {{1}, {{1, 2}, {{1, 2, 3}}}, {{1, 3}}}, {{2}, {{2, 3}}}, {{3}}}
```

The data structure used for the tree is nested lists, each node is of the form:
$\{node, \{child_1, \ldots, child_k\}\}$.

The function `TreeToRules` can be used to convert the tree structures into list rules (which can be used in the graph plot functions).

```
? TreeToRules
```

TreeToRules[tree] returns rules for the argument tree that can be used in GraphPlot.

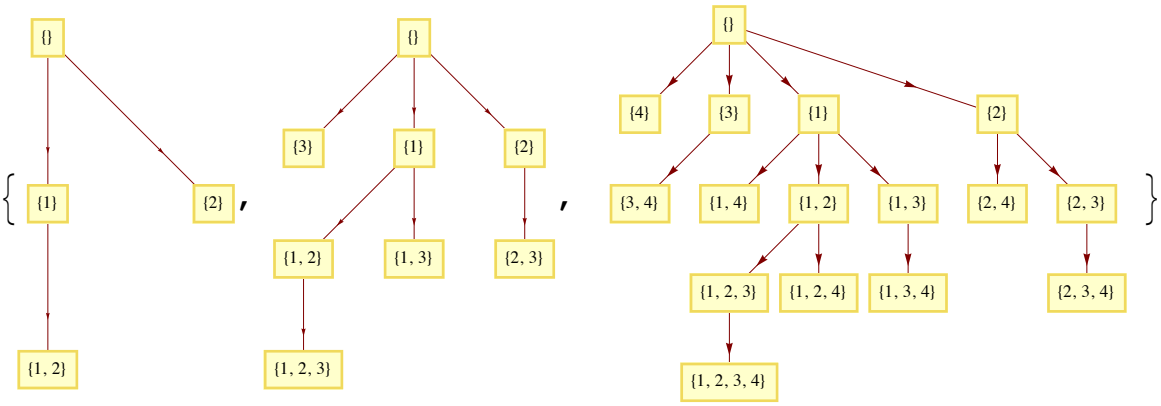
Here is conversion from a tree structure to node-to-node rules:

```
TreeToRules[RymonTree[3]]
```

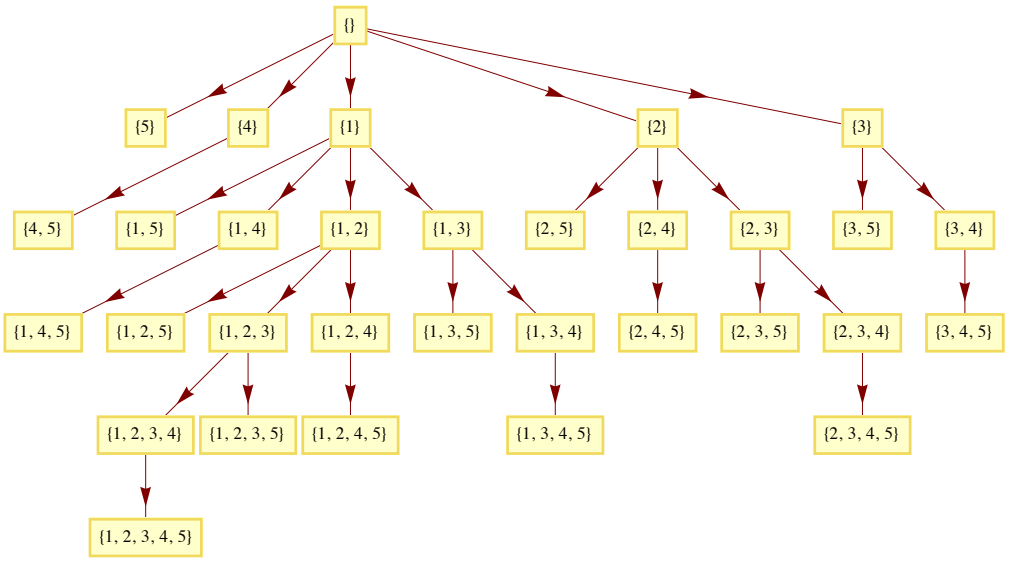
```
{{} → {1}, {} → {2}, {} → {3}, {1} → {1, 2},
  {1} → {1, 3}, {1, 2} → {1, 2, 3}, {2} → {2, 3}}
```

Using the rule representation we plot Rymon trees with the `GraphPlot` functions:

```
LayeredGraphPlot[TreeToRules[RymonTree[#]],
    VertexLabeling → True] & /@ {2, 3, 4}
```




```
LayeredGraphPlot[TreeToRules[RymonTree[5]],
 VertexLabeling → True, ImageSize → 600]
```



References

[1] Nayak, A. and Stojmenovic, I., eds. *Handbook of Applied Algorithms: Solving Scientific, Engineering, and Practical Problems*, Wiley-IEEE Press, 2008.

[2] Hastie, T. et al., *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed., Springer, 2009.

[3] Wikipedia entry about Associative rules learning, http://en.wikipedia.org/wiki/Association_rule_learning.

[4] Antonov, A., Implementation of the Apriori algorithm in *Mathematica*, source code at https://github.com/antononcube/MathematicaForPrediction, (2013).

[5] Antonov, A., Rymon tree generation, http://antonantonov.files.wordpress.com/2011/03/rymontreegeneration.pdf, (2011).

[6] Lenca, P. et al., On selecting interestingness measures for association rules: user oriented description and multiple criteria decision aid, (2006). Available at http://sma.uni.lu/sma/pub/pm-pp-06-01-v01.pdf .