



Important note: Transcripts are **not** substitutes for textbook assignments.

Learning Objectives

By the end of this lesson, you will be able to:

- Interpret the support, confidence, and lift of an association rule.
- Explain how the Apriori algorithm works.
- Mine association rules in R, inspect the rules with highest lift, and prune redundant rules.
- Explain why it's valuable to mine association rules on important subsets of data.



Association Rule Mining

Finding values that frequently occur together.

Examples:

- {Baby food} → {Diapers}
 - {Age > 50, High Cholesterol} → {High Blood Pressure}
- Antecedent Consequent

Association rule mining is a form of unsupervised learning in which we find sets of items, or values of variables, that frequently occur together. For example, maybe customers who buy baby food are more likely to also buy diapers than randomly chosen customers. Or maybe a patient who is over age 50 and has high cholesterol is more likely to also have high blood pressure, than a randomly chosen patient. In these rules, the left-hand side is called the antecedent and the right-hand side is called the consequent. Notice that we're not saying anything about a causal relationship here. Remember that correlation does not imply causation. We're just saying that these sets of items, or variables, tend to go together-- that is that there are associated.

Measuring the Quality of a Rule $A \rightarrow B$

Support: $P(A \text{ and } B)$

Confidence: $\frac{P(A \text{ and } B)}{P(A)} = P(B|A)$

Lift: $\frac{P(B|A)}{P(B)}$

How much more likely are you to buy B, knowing that you bought A?

There are many ways to measure the quality of a rule $A \rightarrow B$, we'll talk about three of them. The first is the support-- this is the proportion of people in the data who bought both all of the items in set A and all of the items in set B. In other words, this is a way to estimate the probability that a randomly chosen customer would buy everything in A and B.

Next is the confidence, this is the support of the rule $A \rightarrow B$, divided by the support of just A. In terms of probability, this is the conditional probability of buying B given that someone bought A. And finally, we have the lift. This is the confidence of the rule $A \rightarrow B$, divided by the support of B. In terms of probability, the lift measures how much more likely a customer is to buy B knowing that they bought A, as compared to the probability that a randomly chosen customer would have bought B without having any information about whether they bought A. Notice that a lift value of 1 means that the customer is equally likely to buy B, regardless of whether they bought A or not, so we're primarily interested in values of lift greater than 1.

Check Your Understanding

Find the support, confidence, and lift of the rule Bike \rightarrow Jog.

Preferred leisure activities

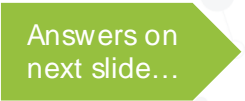
Person 1 Bike, jog, read

Person 2 Bike, jog

Person 3 Jog, TV

Person 4 Bike, read

Person 5 TV



Answers on
next slide...

Bike → Jog

Preferred leisure activities

Person 1 Bike, jog, read

Person 2 Bike, jog

Person 3 Jog, TV

Person 4 Bike, read

Person 5 TV

Support: $2/5$

Confidence: $2/3$

Lift: $(2/3) / (3/5) = 10/9$

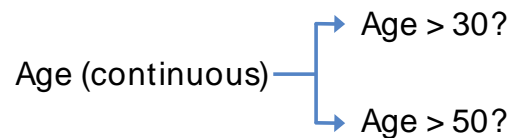
In this example, there are two people who like to bike and jog- that's person 1 and 2. So the support of the rule, bike arrow jog, is 2 divide by 5, the size of the entire dataset. To find the confidence, we take the support of the rule, $2/5$, and divide it by the support of the antecedent, the support of just bike. In this case, there are three people who like to bike-- persons 1, 2, and 4-- so the support of the antecedent is $3/5$. $2/5$ divided by $3/5$ is $2/3$. Finally to find the lift, we take the confidence, $2/3$, and divide it by the support of the consequent, jog. There are three people who like to jog-- persons 1, 2, and 3-- so the support of the consequent is $3/5$. That means the lift is $2/3$ divided by $3/5$, or $10/9$.

Market Basket Analysis

Simplified association rules for large data sets ($\approx 10^3$ variables or items, $\approx 10^4$ transactions or customers).

Single-item consequents.

All variables binary.



Market basket analysis refers to a simplified version of association rule mining that's especially good for large data sets up to about 10^3 variables or items and up to about 10^4 transactions or customers. The association rules here are simplified so that the consequent, or right-hand side, of each rule contains only a single item. And all of the variables are binary-- that is, they have yes or no responses or they can take on values of 0 or 1. A person either bought bread or they didn't, but you don't keep track of the difference between buying five loaves of bread and buying 10.

For example, if we have age, which is a continuous variable, you might split that up into two binary variables. Is the person's age greater than 30, yes or no? And then is the person's age greater than 50, yes or no?

Identifying High-Support Sets of Items

Goal: Find all sets K with $P(K) > t$.

Challenge: N items results in 2^N sets of items.

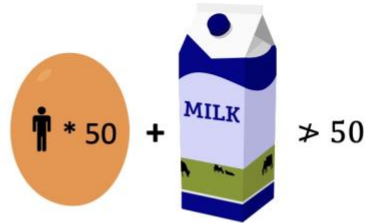


To conduct market basket analysis, we want to identify sets of items, or variables, that have high support, say greater than some threshold little t . Once we've done that, we'll be able to choose one item from each set to be the consequent and the rest of the items in the set will be the antecedent. The challenge here is that if you have n items or variables, then you have 2^n possible sets of items, because each item can be either in or out of a particular set. This number 2^n increases quickly as the number of items n increases, so it can be time consuming to check the support of every possible set.

Apriori Algorithm

Principle:

If $L \subset K$, then $P(L) \geq P(K)$.

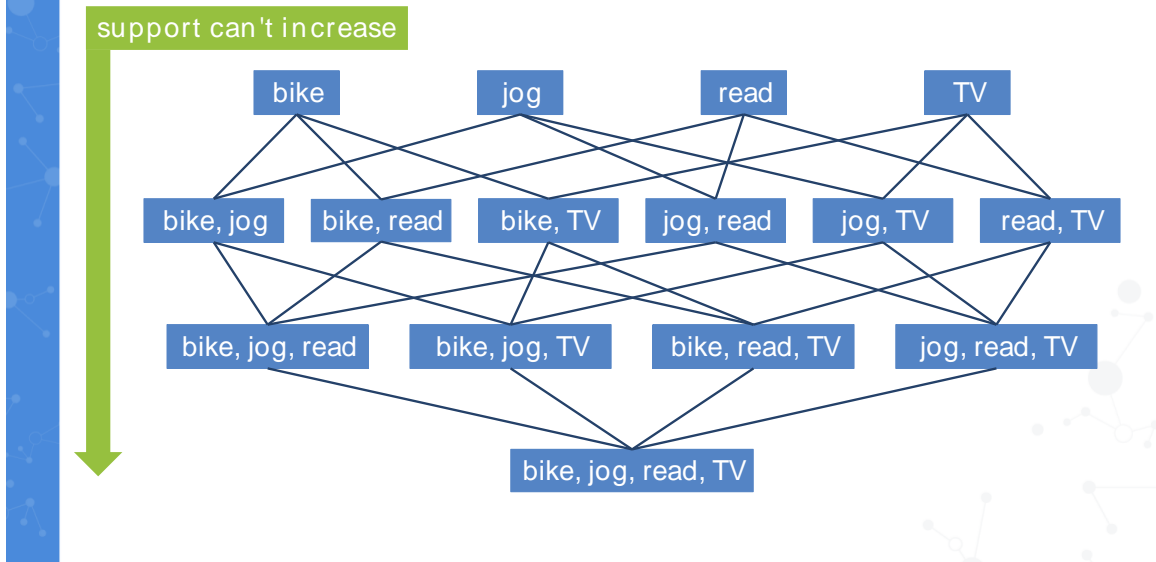


1. Check all **1-item** sets.
2. Check all **pairs** of "surviving" items.
3. Eliminate **trios** that contain 1 or more items that have been eliminated; check the rest.
4. etc...

The Apriori algorithm is an efficient method for market basket analysis that works on the principle that if L is a subset of K , then the support of L must be greater than or equal to the support of K . In other words, if you increase the number of items on a grocery list, then the proportion of people who buy all of the things on that list can't increase. For example, if there are 50 people in your data set who buy eggs, possibly in addition to other things, then the number of people in your dataset who buy eggs and milk can't be greater than 50.

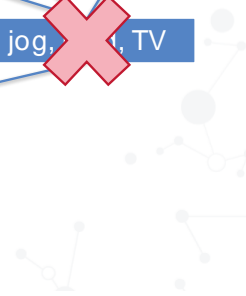
So the way that Apriori algorithm works is that it first checks the support of all one item sets and eliminates any items that have support less than min_support , whatever support threshold you chose. Then it checks all of the pairs of surviving items and eliminates any that have low support. Then it goes on to eliminate any true use of items that contain one or more items that have been eliminated and checks the support of the remaining trios. And it continues in this fashion until it's checked all possible item sets.

Apriori Algorithm in Graph Form



Here's a graphical way to think about the Apriori algorithm. In the first row, we have each of the one item subsets from a data set on people's hobby's. The second row contains all of the two items subsets, and so on. You'll notice that each set is connected to the sets in the row above and below it that are subsets or supersets of it. For example, the set bike, jog is connected to its subsets bike and jog, and to its supersets-- bike, jog, read and bike, jog, TV-- which are formed by adding one item to the set bike comma jog.

The principle on which the Apriori algorithm is based is that by following an edge from one row to the row below it we can't increase the support. For example, by going from the bike to the set bike comma jog, the support could stay the same-- that would be if everyone who bikes also jogs. Or the support could decrease, but it can't increase.



The Apriori algorithm will conduct a breadth first search across each row of this graph. So first, it will check the support of each of the one item sets, then each of the two item sets, and so on. But suppose that while we're checking the support of the two item sets, we notice that the set read comma TV has low support-- that means we can eliminate it from consideration. But we also know that as we travel along the edges from read comma TV to it's supersets, the support can't increase-- that means that if read comma TV has low support, then the sets bike, read, TV and jog, read, TV also have low support, so we can eliminate them right away. We don't even need to compute the support for those sets.

Similarly, we can also eliminate the four items set because it contains two sets-- bike, read, TV and jog, read, TV-- that we have low support. This is a way to eliminate some sets from consideration, so we can avoid having to check the support of 2 to the n item sets. The Apriori algorithm is most efficient when our support threshold, little t , is high enough that we can eliminate a large number of subsets right away in this fashion.

Sample Problem

What associations can we extract about characteristics of people who live in the Bay Area?

```
> library(arules)
> data(Income)
> summary(Income)
```

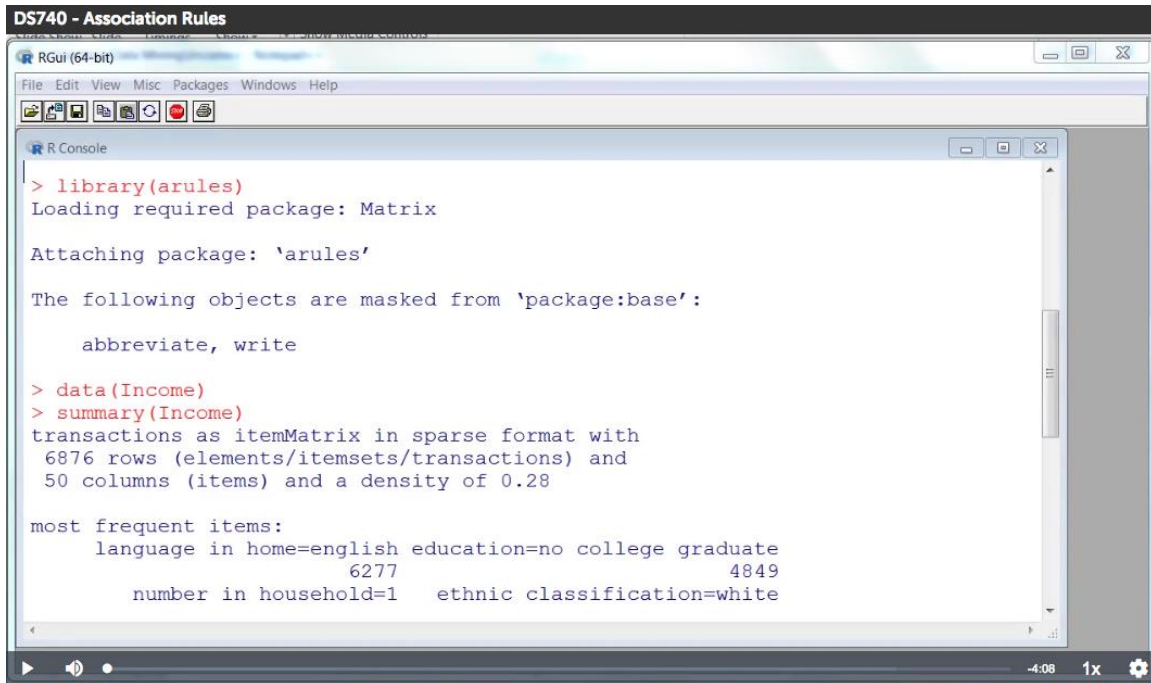
transactions as itemMatrix in sparse format with
6876 rows (elements/itemsets/transactions) and
50 columns (items) and a density of 0.28

most frequent items:

language in home=english	education=no college graduate	number in household=1
6277	4849	4757
ethnic classification=white	years in bay area=10+	(Other)
4605	4446	71330

Notes:

```
library(arules)
data(Income)
summary(Income)
```



```
> library(arules)
Loading required package: Matrix

Attaching package: 'arules'

The following objects are masked from 'package:base':

    abbreviate, write

> data(Income)
> summary(Income)
transactions as itemMatrix in sparse format with
6876 rows (elements/itemsets/transactions) and
50 columns (items) and a density of 0.28

most frequent items:
language in home=english education=no college graduate
                        6277                        4849
number in household=1   ethnic classification=white
```

This slide represents a video/screencast in the lecture. The transcript does not substitute video content.

The income data set is contained in the A rules package. By using the function `summary`, we can see that this dataset contains 6,876 rows which represent people. In a sales data set, the rows would represent transactions. It also contains 50 columns, where, here, each column represents a variable. In a sales data set, each column would represent one item. We can also see that the dataset has already been converted to a transactions or item matrix format, which is what we need in order to perform a priori in R.

By scrolling down through the summary, we can see more information about this data set, such as some of the most frequent items or values of variables in the dataset. Another good way to get an idea of what's going on with this data set is to look at an item frequency plot. This will show us each variable in the data set and what its support is. But with so many variables, this is pretty hard to read. So we can look at a restricted version of this graph by looking at only those items that have support greater than 0.4.

So here we can see some of the most common variables and what their supports are. To use the a priori algorithm in R, we use the `a priori` function. The first argument is the dataset, in this case, `income`. And then we can use the argument parameter `equals` and `s` list containing the support we want and the confidence. A support of 0.05 means that at least 5% of the people in the data must share all of the characteristics in the item set in order for that item set to be included as a potential association rule.

With this size of dataset, 5% of the people means at least 344 people. If we decreased the support and confidence, then we would get more rules. By looking at the summary, we see that we ended up with 30,526 rules. In order to see what these rules are, we would use the inspect function. But we probably don't want to inspect more than 30,000 rules at a time. So instead we can use the head function to look at just, say, the top three rules when they're sorted by their lift. For example, the rule in this set of association rules that has the highest lift, in this case 4.75, has an antecedent of single students who have been in the Bay Area 10 or more years, have one or more children, and live in a house. And the consequent is that they are more likely to live with parents or family than a randomly chosen person in the dataset.

Notes:

Be sure you're using an up-to-date version of R. Otherwise, the command `head(rules, n = 3, by = "lift")` may not sort the rules correctly.

Size of rules

- *apriori()* returns rules with one consequent
- By default, up to length 10 (9 antecedents)

```
more_rules = apriori(Income, parameter = list(support = .05,  
                                              confidence = 0.9,  
                                              maxlen = 12))
```

The a priori algorithm performs market basket analysis. So it returns so-called simple association rules, those with one consequent or one thing on the right-hand side. By default, the a priori function in R will return rules of up to length 10, meaning they have one consequent and up to nine antecedents on the left-hand side. You can change this by setting the maxlen parameter in the a priori function.

Notes:

```
more_rules = apriori(Income, parameter = list(support =  
.05,  
                                              confidence = 0.9,  
                                              maxlen = 12))
```

What affects the number of rules?

- Support and confidence
 - Few rules of the form cat food and X \rightarrow dog food?
 - Buying dog food is not associated with buying cat food
 - Not many people buy cat food
 - Not many people buy dog food
 - Not many people buy cat food *with* a variety of other things X (in the data)

Person	Cat food and dog food?	Other thing
1	Yes	Fish food
2	Yes	Leash
3	Yes	Cat toy
...

So besides any restrictions on the size of rules, what else affects the number of rules that are returned? Well, the simple answer is the support and confidence that you asked for when you generated the rules. But this has important consequences for how you interpret the rules that are returned. For example, suppose that in a particular data set you find that there are very few rules of the form cat food and something else with a consequence of dog food.

This could mean that buying dog food is not associated with buying cat food. Or in other words, if a person is a cat owner, they probably aren't a dog owner. But it could also be that we have low support for cat food or dog food or both. If you asked for rules with a support of 0.05 and less than 5% of the people in your data buy cat food, then you're not going to have any rules involving cat food, even if everybody with a cat also has a dog.

Another possibility is that we have low support for cat food with a variety of other things in the data. In this case, we might have one rule of the form cat food \rightarrow dog food that has high support, confidence, and lift. But if everybody who bought cat food and dog food bought a different other thing, then no one other thing x is going to rise to a sufficient level of support when combined with cat food and dog food.

Redundant Rules

LHS	redundant	RHS	Support	Confidence	Lift
Retired, 10+ years in Bay Area		No children	.0580	.952	1.531
Retired		No children	.0676	.953	1.532
more general				higher	

By inspecting the rules we can mine from the income data set, you'll notice pairs of rules like these. Both of these rules have the same consequence, or right-hand side. They're both telling us about the characteristics of people who are more likely to have no children compared to randomly selected people in the Bay Area. However, when you look at their antecedent, or left-hand side, you'll notice that the second rule is a subset of the first-- that means that it's more general. It applies to more people in the Bay Area, because there are more people who are simply retired than who are retired and have been there for 10 or more years. That means that the support of the second rule has to be greater than or equal to the support of the first rule.

So the second rule is more general, but now notice that in this case it also has higher confidence. Because the consequent is the same, higher confidence also means higher lift. So the second rule is more general and it gives us more predictive information about whether the person is likely to have no children. That means that the first rule here is redundant-- it's more specific and gives us less information.

Notes:

```
inspect( rules[ c(3, 55) ] )
```

Pruning Redundant Rules

not



```
non_redundant = !is.redundant(rules)
summary( rules[non_redundant] )
```

```
non_redundant = (interestMeasure(rules,
                                measure = "improvement",
                                quality_measure = "confidence") > 0)
```

If we only want to look at the non-redundant rules, we can do that by using the function `is dot redundant` to get a vector of trues and falses that tell which rules are redundant. Then we can use an exclamation point to negate those trues and falses. So the vector `non_redundant` contains true for every non-redundant rule. Then we can use `summary` of `rules square bracket non_redundant` to look at those rules.

A slightly more complicated way to do the exact same thing is to use the `interest measure` function to identify rules that have a positive improvement in their confidence measure. This approach is useful if you want to tweak it a little bit, for example, if you only want to identify rules that have an improvement in their confidence measure of at least 0.1.

Notes:

```
non_redundant = !is.redundant(rules)
summary( rules[non_redundant] )
```

```
non_redundant = (interestMeasure(rules,
                                measure = "improvement",
                                quality_measure = "confidence") > 0)
```

Association Rules are Only as Good as the Data

Data must represent the population of interest.

Simpson's Paradox:

- Other variables can change direction of association.

		Read?	
		Yes	No
TV?	Yes	527	570
	No	113	336

$$\text{Lift (TV} \rightarrow \text{Read)} = 1.16$$

An important thing to remember about association rule mining is that the rules we get out of it are only as good as the data we put into it. Your data have to represent the population that you're interested in. For example, the income data set was based on people sampled from San Francisco's Bay Area, so the rules that we found may or may not apply to people from Los Angeles, New York City, or any other region we might be interested in. Also, if the data were sampled in a way that was biased, that bias will carry through to the rules we found.

One tricky aspect of this is Simpson's Paradox, which is the fact that other variables that aren't included in the left-hand side of a particular rule can change the direction of an association. Here's an example. When thinking about people's hobbies, we might have this information about whether they like to read and whether they like to watch TV. In this example, the lift of the rule TV \rightarrow read is 1.16, so people who watch TV are more likely to also read than randomly selected people in the data set.

Mining Subsets of Data

With a college degree: $\text{Lift}(TV \rightarrow Read) = 0.96$

		Read?	
		Yes	No
TV?	Yes	511	314
	No	89	19

Without a college degree: $\text{Lift}(TV \rightarrow Read) = 0.90$

		Read?	
		Yes	No
TV?	Yes	16	256
	No	24	317

Good Practice

Mine association rules on key subsets as well as entire sample.

However, maybe we're interested in breaking this data set down to look at people with a college degree and people without a college degree, and those data could look like this. Now the rule TV arrow read has lift 0.96 for people with a college degree and 0.9 for people without a college degree. Both of those values are less than 1, which means that people who watch TV are actually less likely to read compared to people who are randomly selected from the data. So in this case, mining subsets of data completely reversed the direction of the conclusion that we found. This is an example of Simpson's Paradox. It happens because, in this case, people with a college degree are more likely to say they enjoy reading, but there are also more people with a college degree who watch TV compared to people without a college degree. So these two factors combine to make it appear that people who watch TV are also more likely to read.

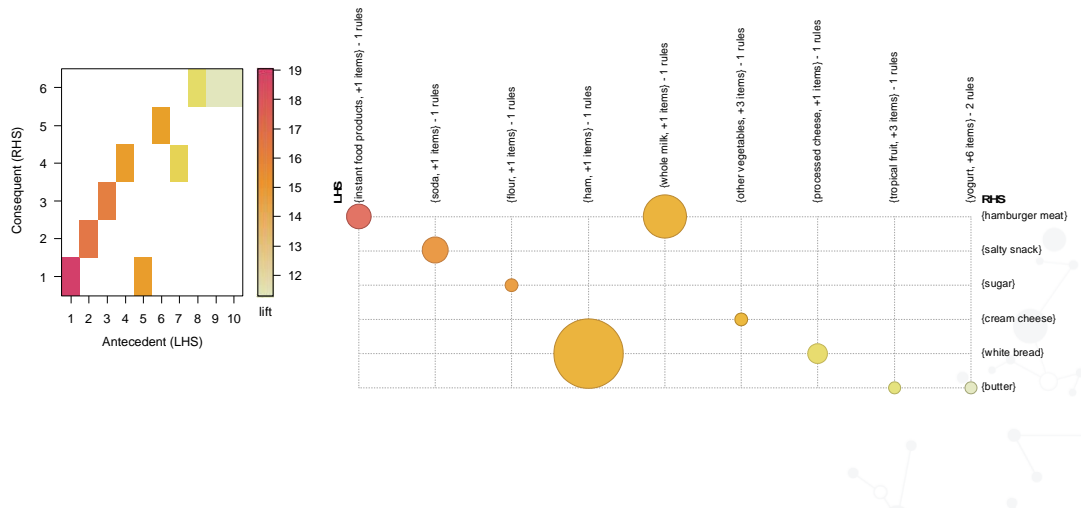
If there's a particular subset of your data that you're particularly interested in, it's a good idea to look at the association rules that you find from that subset as well as from the original whole data set. For example, you might want to mine association rules from men and from women, as well as from the whole population. Or, in a sales context, you might want to mine association rules separately for each branch of your company.

Notes:

For more about Simpson's paradox, see section 6.7.3 of [Introduction to Data Mining: Chapter 6 - Association Analysis: Basic Concepts and Algorithms](#). Pang-Ning Tan, Michael Steinbach, Vipin Kumar.

Visualizing Association Rules

arulesViz package



If you're interested in visualizing association rules, you might want to check out the arulesViz package. Here are two examples of graphs that you can get from this package. For example, both of these graphs are showing that there are three rules with consequent number 6, which is butter, but that are rule with consequent number 1, hamburger meat, has the strongest support shown in the bright red. In general, I think that most of the visualizations that are available in this package have a ways to go before they reach their full potential, but if visualization is important in your job, you may want to check this package out.

Notes:

```
grocery_rules = apriori(Groceries, parameter = list(support = .001, confidence = 0.5))
subrules = head(grocery_rules, n = 10, by = "lift")
plot(subrules, method="matrix", measure="lift")
plot(subrules, method="grouped")
```

DS740 - Association Rules

Inspecting the Bay Area rules:

```
```\n{r}\ninspectDT(rules)\n```\n
```



One function from the `arulesViz` package that's worth checking out is the `inspectDT` function. You apply this function to an existing set of rules, and it will create an HTML widget that allows you to explore the rules interactively.

So here it's showing 10 rules at a time, and we can scroll through them. We can also use these boxes up at the top to take subsets of rules. For example, we could subset to only rules that have a right-hand side or a consequent of age equal to 35 plus. And we can see that limits us to 271 rules.

Alternatively, we could use the boxes for the numerical quantities to use a slider bar to subset to rules with certain values of support, confidence, or lift.

Once we have rules that we want to look at, we can then use the arrows at the top to sort based on that characteristic. So for example, I could sort my rules based on lift by clicking on the arrow next to the lift. Initially it sorts in increasing order, and if we click it again, it will sort in decreasing order so that the rules with the highest lift are at the top.

This is an HTML widget, so it will only display if you're knitting your RMarkdown document to HTML. If you want to include this code in your RMarkdown document just as a record, but you want to knit your document to Word or PDF, up at the top you can include the code `always_allow_HTML: true`.

## Summary

The confidence of a rule  $A \rightarrow B$  is the probability that a person will buy (or have characteristic) B, given that they bought A.

The lift of  $A \rightarrow B$  is how much more likely a person is to buy B if they bought A, compared to a randomly chosen person.

The Apriori algorithm is an efficient way to find high-support itemsets, by eliminating any sets that contain other sets that are known to have low support.



## Summary

A rule is considered redundant if it has lower lift or confidence than a more general rule with the same consequent.

Association rules are only as good as the data from which they're mined.

It can be valuable to mine association rules on subsets of the data.

- to understand relationships within important subpopulations.
- to avoid the risk of the subpopulation variable reversing the direction of the association.

