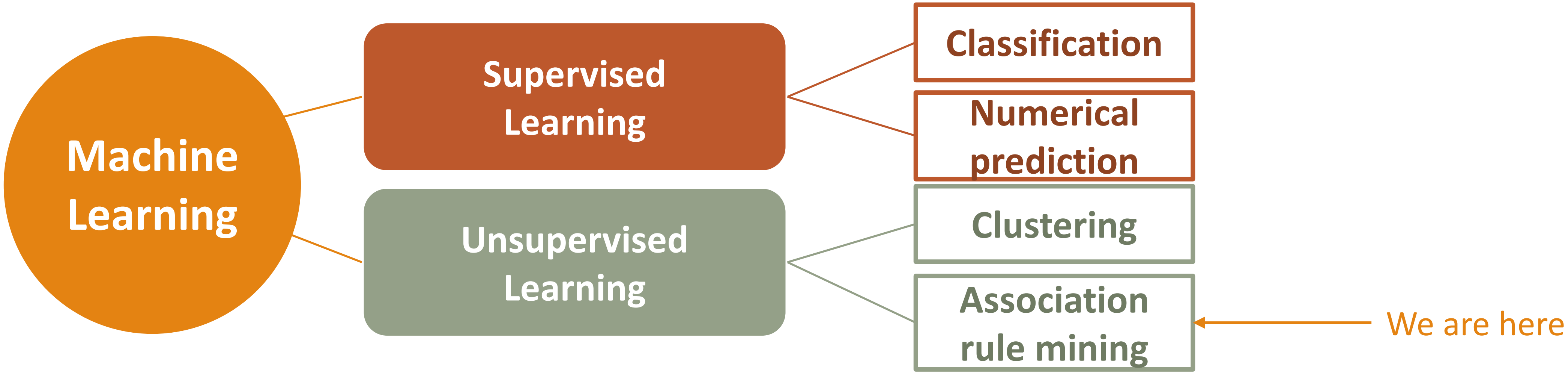
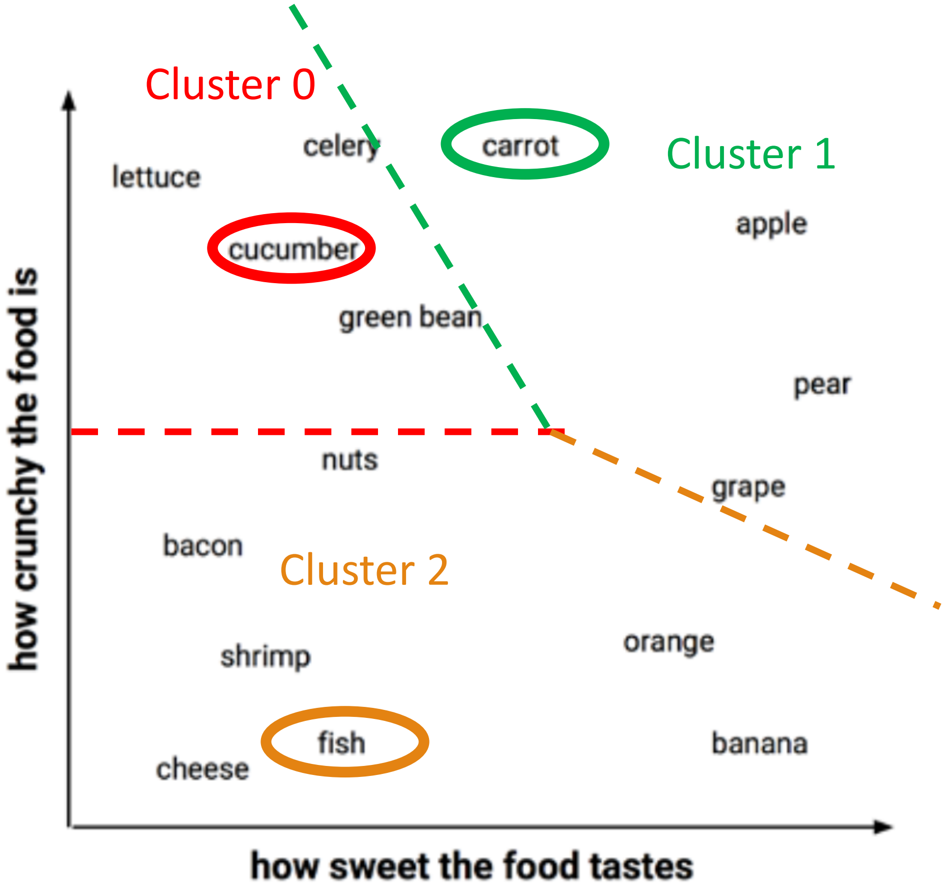
Lecture 10:Unsupervised Learning: Association Rule Mining

Data Mining Tasks: Recap



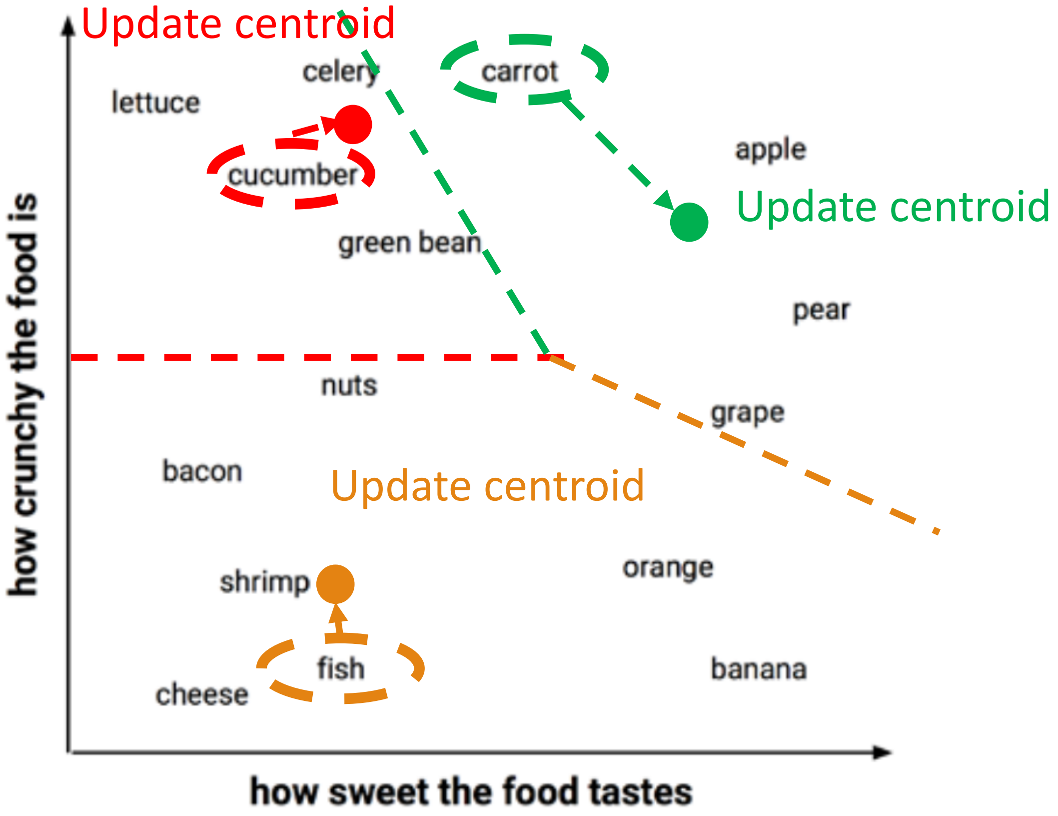
The k-means Clustering Algorithm: Recap



Keep in mind that as we are using distance calculations, all the features need to be numeric, and the values should be normalized to a standard range ahead of time.

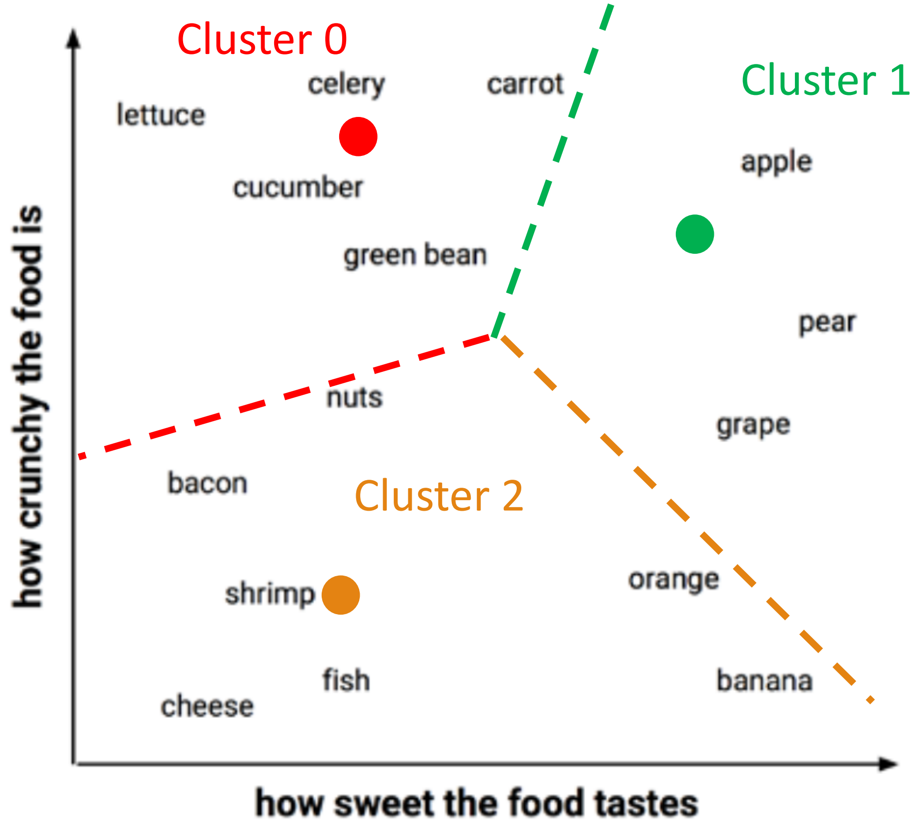
The k-means Clustering Algorithm: Recap

- Step 3: shifting the initial centers to a new location, known as the **centroid**, which is calculated as the average position of the points currently assigned to that cluster.



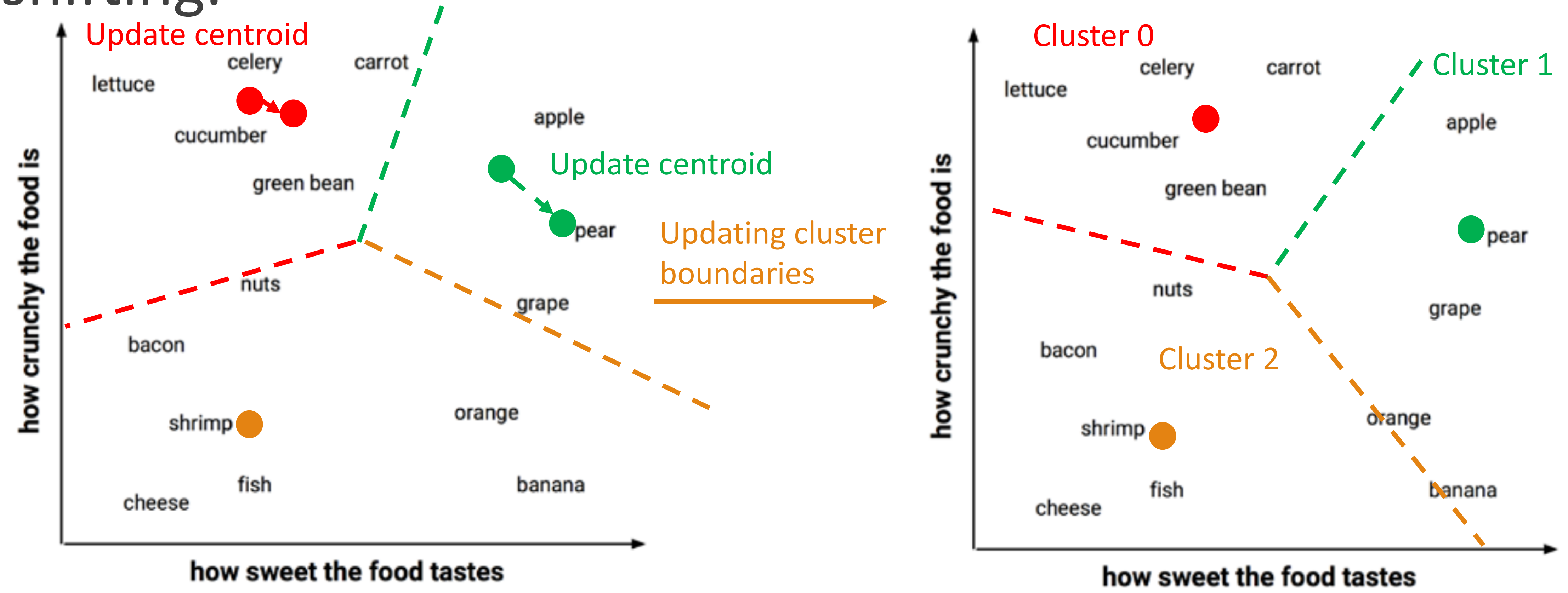
The k-means Clustering Algorithm: Recap

- Step 4: Updating the cluster boundaries, and reassigning points into new clusters.

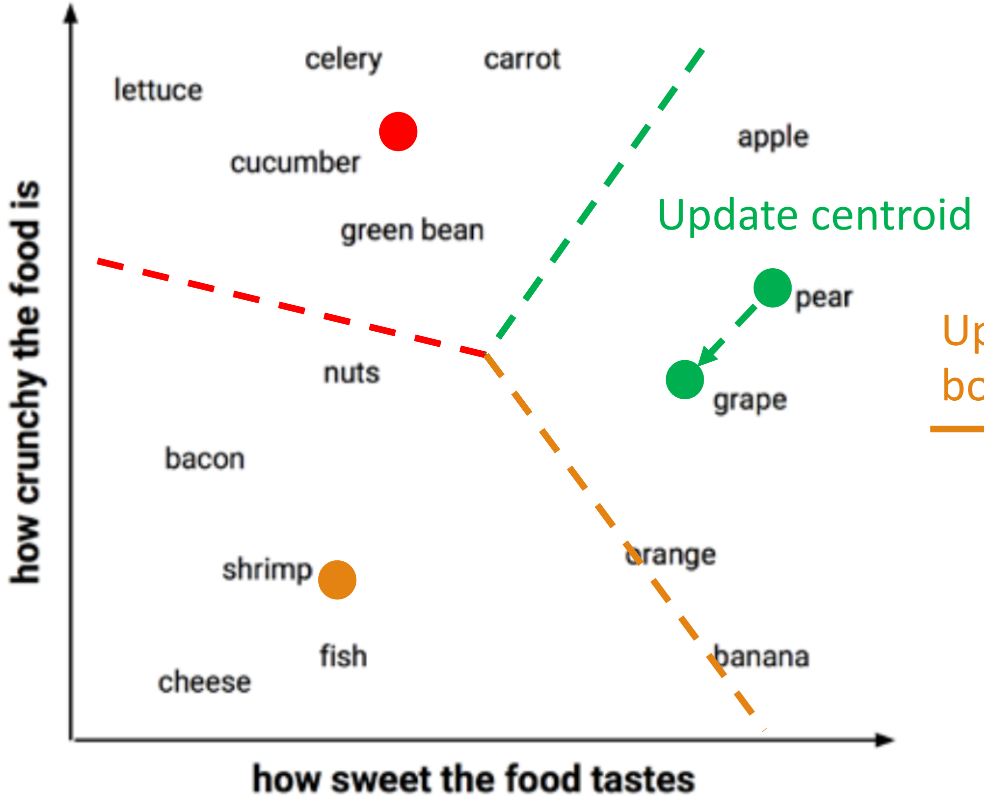


The k-means Clustering Algorithm: Recap

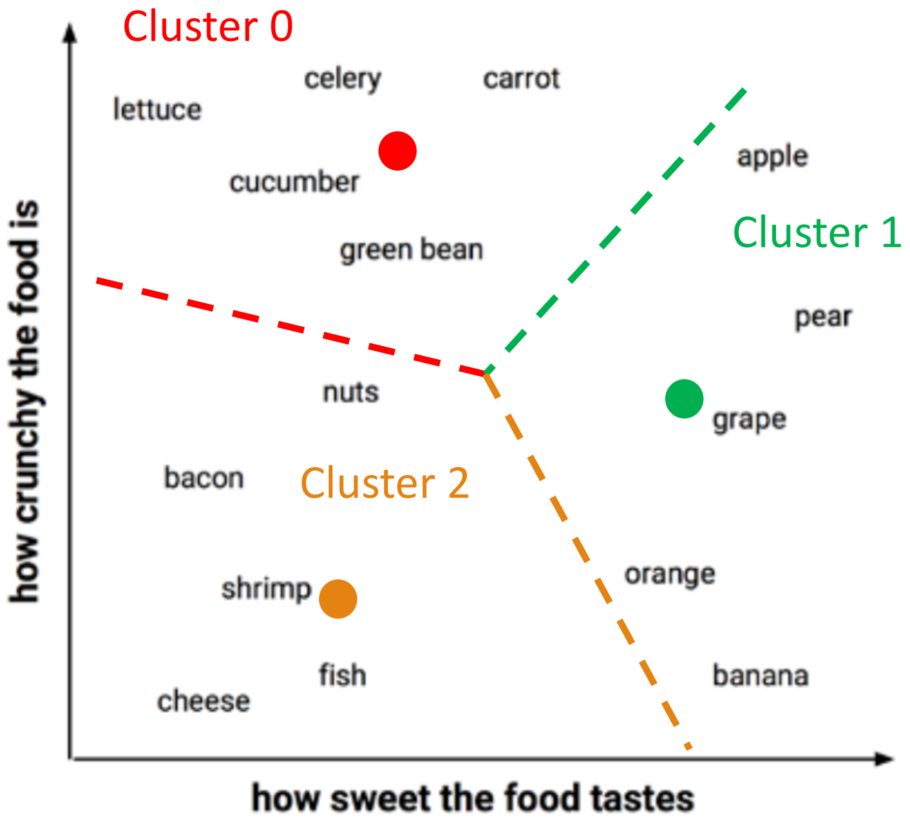
- Step 5: Repeat step 3 and step 4 until the centroids stop shifting.



The k-means Clustering Algorithm: Recap



Updating cluster boundaries



The k-means Clustering Algorithm: Recap

- k=2

```
Final cluster centroids:
                                   Cluster#
Attribute               Full Data         0          1
                         (5493.0)  (2678.0)   (2815.0)
======================================================
Age                        3.4837    4.5732     2.4472
DistToWork                11.4828   11.5362     11.432
DualInc                    0.2439    0.4671     0.0316
Education                  3.8724    4.4145     3.3567
Gender                     0.5385    0.6247     0.4565
Income                     5.1606    6.9712      3.438
NbrInHouseHold             2.9053    2.7054     3.0956
NbrInHouseholdUnder18      0.7073    0.5963     0.8128
YrsInArea                  4.2922    4.5063     4.0885
Rider                      0.4285    0.1617     0.6824
Language_English           0.9148    0.9563     0.8753
Language_Spanish           0.0553    0.0295     0.0799
OwnRent_own                0.4353    0.8794     0.0128
OwnRent_Parent             0.2554         0     0.4984
```

Cluster size

Centroids

The k-means Clustering Algorithm: Recap

- k=3

Final cluster centroids:

Attribute	Full Data (5493.0)	Cluster# 0 (2368.0)	1 (1210.0)	2 (1915.0)
Age	3.4837	4.6917	3.2603	2.1311
DistToWork	11.4828	11.5051	11.5231	11.4298
DualInc	0.2439	0.4472	0.1694	0.0397
Education	3.8724	4.4054	4.1347	3.0475
Gender	0.5385	0.5904	0.4893	0.5055
Income	5.1606	6.9949	5.6868	2.5598
NbrInHouseHold	2.9053	2.7758	2.2653	3.47
NbrInHouseholdUnder18	0.7073	0.6326	0.3702	1.0125
YrsInArea	4.2922	4.5743	3.9256	4.1749
Rider	0.4285	0.1858	0.0149	0.9901
Language_English	0.9148	0.951	0.9612	0.8407
Language_Spanish	0.0553	0.0329	0.0256	0.1018
OwnRent_own	0.4353	0.9996	0	0.0125
OwnRent_Parent	0.2554	0	0.1157	0.6595

Based on the cluster size and centroids, which k (k=2 or k=3) you will use, and why?

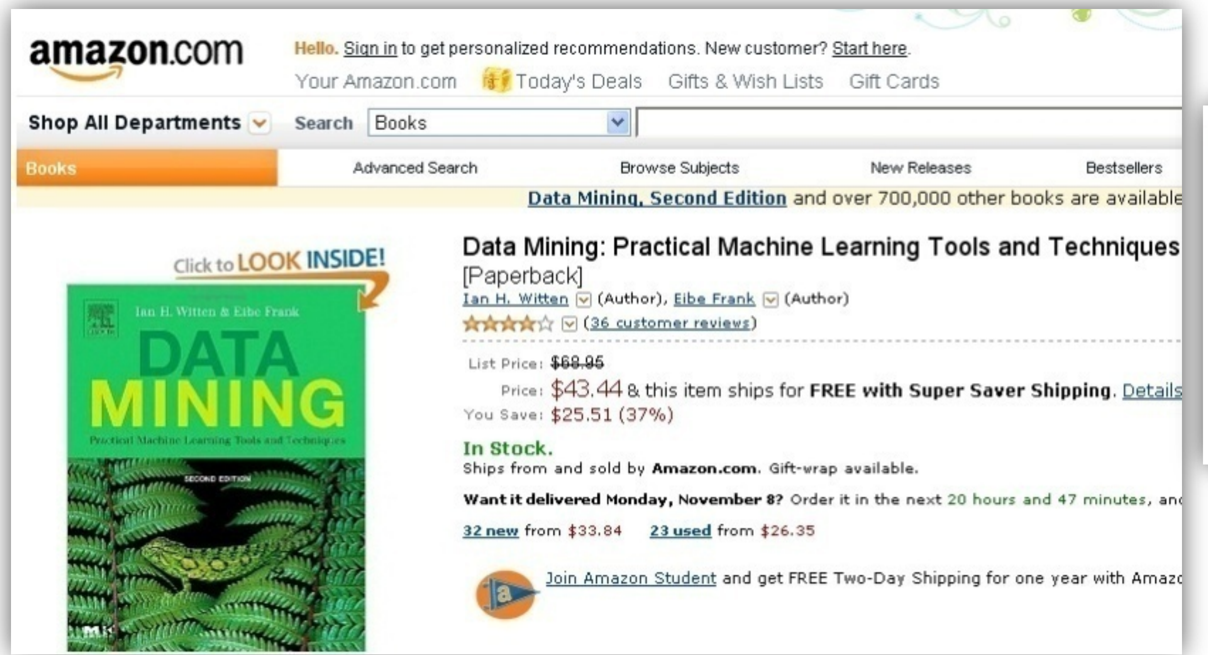
If we segment residents into 3 clusters, what marketing plans you can use to target each cluster?

Outline

- Finding Patterns – Market Basket Analysis
- Understanding Association Rules
- Apriori Algorithm for Association Rule Learning
- Measuring Rule Interest – Support, Confidence, and Lift
- Building a set of Rules with the Apriori Principle

Market Basket Analysis

- **Unsupervised learning-Association rule mining**



Promotions to
retain customers
& increase sales

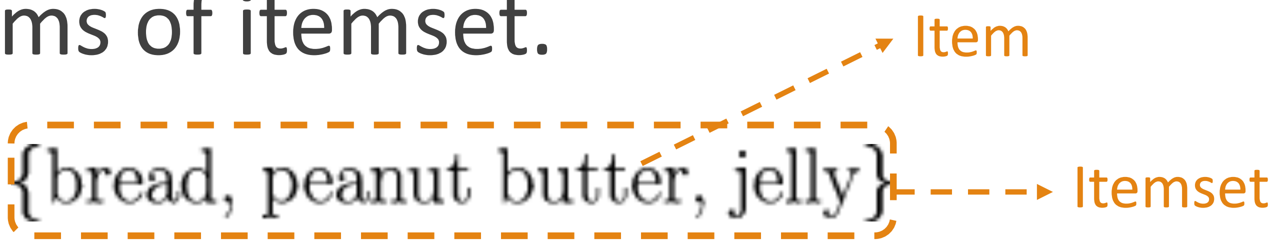
Market Basket Analysis

- **Unsupervised learning-Association rule mining**
  - A classic case: Diaper and Beer

Understanding Association Rules

Market Basket Analysis

- The building blocks of a market basket analysis are the **items** that may appear in any given transaction.
- Groups of one or more items are surrounded by brackets to indicate that they form a set, or more specifically, an **itemset** that appears in the data with some regularity. Transactions are specified in terms of itemset.



Understanding Association Rules

- The result of a market basket analysis is a collection of **association rules** specify patterns found in the relationships among items in the itemsets.
  - Association rules are always composed from subsets of itemsets and are denoted by relating one itemset on the ***left-hand side (LHS)*** of the rule to another itemset on the ***right-hand side (RHS)*** of the rule.
  - The ***LHS is the condition*** that needs to be met in order to trigger the rule, and the ***RHS is the expected result*** of meeting that condition.

$$\{\text{peanut butter, jelly}\} \rightarrow \{\text{bread}\}$$

Understanding Association Rules

$$\{\text{peanut butter, jelly}\} \rightarrow \{\text{bread}\}$$

- If peanut butter and jelly are purchased together, then bread is also likely to be purchased. In other words, "peanut butter and jelly imply bread."

Understanding Association Rules

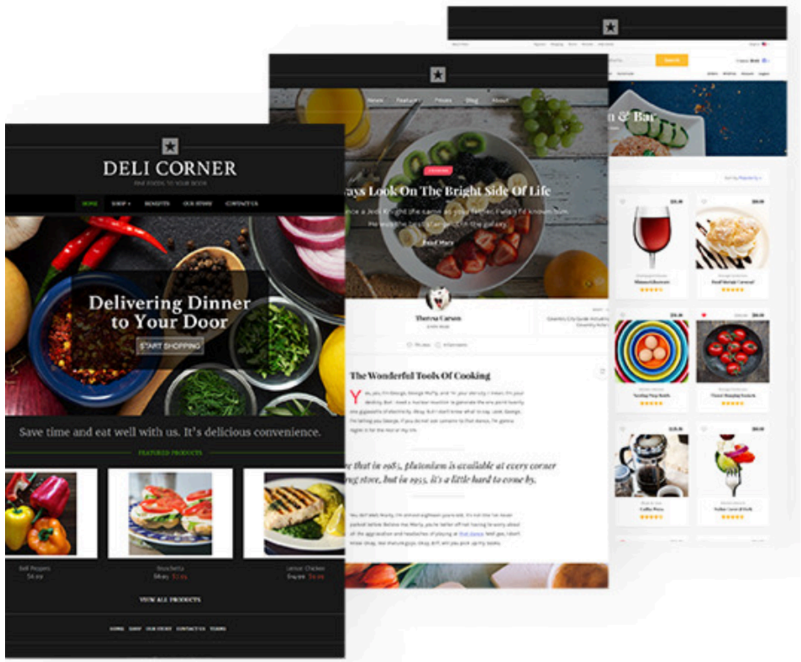
- Association rule learners are **unsupervised**
  - No need for the algorithm to be trained
  - Data does not need to be labeled ahead of time
  - The program is simply unleashed on a dataset in the hope that interesting associations are found
  - Hard to to objectively measure the performance of a rule learner

Understanding Association Rules

- Although association rules are most often used for market basket analysis, they are helpful for finding patterns in many different types of data.
  - Searching for interesting and frequently occurring patterns of DNA and protein sequences in cancer data
  - Finding patterns of purchases or medical claims that occur in combination with fraudulent credit card or insurance use

Understanding Association Rules

- ***Web Log analysis:*** Web sequential patterns are very useful to better structure a company's website for providing easier access to the most popular links.



How this application works?

- We can use association rules in order to discover the pages which are visited together even if they are not directly connected.
- The rules can be used for restructuring Web sites by adding links between those pages which are visited together.

Understanding Association Rules

- Association rule analysis: search for interesting connections among a very large number of elements.
  - Human beings are capable of such insight quite intuitively
  - It often takes expert-level knowledge or a great deal of experience to do what a rule learning algorithm can do in minutes or even seconds.
  - Additionally, some datasets are simply too large and complex

The Apriori Algorithm for Association Rule Learning

- The most-widely used approach for efficiently searching large databases for rules is known as **Apriori**.
  - The following table shows five completed transactions in an hospital's gift shop:

Transaction number	**Purchased items**
1	*{flowers, get well card, soda}*
2	*{plush toy bear, flowers, balloons, candy bar}*
3	*{get well card, candy bar, flowers}*
4	*{plush toy bear, balloons, soda}*
5	*{flowers, get well card, soda}*

The Apriori Algorithm for Association Rule Learning

- The Apriori algorithm uses statistical measures of an itemset's "**interestingness**" to locate association rules in much larger transaction databases.
- Whether or not an association rule is deemed interesting is determined by two statistical measures: **support** and **confidence** measures.

The Apriori Algorithm for Association Rule Learning

- The **support** of an itemset or rule measures how frequently it occurs in the data.
  - Support for the itemset *X*: $\text{support}(X) = \dfrac{\text{count}(X)}{N}$
    - *N* is the number of transactions in the database and *count(X)* is the number of transactions containing itemset *X*
  - Support for a single item:
    - The support for *{candy bar}* is *2 / 5 = 0.4*
  - Support for itemset:
    - *{get well card, flowers}* has support of *3 / 5 = 0.6* in the hospital gift shop data
    - The support for *{get well card}* → *{flowers}* is also 0.6

The Apriori Algorithm for Association Rule Learning

- A rule's **confidence** is defined as the support of the itemset containing both *X* and *Y* divided by the support of the itemset containing only *X*:

$$\text{confidence}(X \rightarrow Y) = \frac{\text{support}(X, Y)}{\text{support}(X)}$$

  - The confidence tells us the proportion of transactions where the presence of item or itemset *X* results in the presence of item or itemset *Y*

The Apriori Algorithm for Association Rule Learning

- **Confidence that *X* leads to *Y* is not the same as the confidence that *Y* leads to *X***
  - The confidence of *{flowers}* → *{get well card}* is *0.6 / 0.8 = 0.75*
  - The confidence of *{get well card}* → *{flowers}* is *0.6 / 0.6 = 1.0*
    - a purchase involving flowers is accompanied by a purchase of a get well card 75 percent of the time, while a purchase of a get well card is associated with flowers 100 percent of the time.

Transaction number	Purchased items
1	*{flowers, get well card, soda}*
2	*{plush toy bear, flowers, balloons, candy bar}*
3	*{get well card, candy bar, flowers}*
4	*{plush toy bear, balloons, soda}*
5	*{flowers, get well card, soda}*

The Apriori Algorithm for Association Rule Learning

- Rules like *{get well card}* → *{flowers}* are known as **strong rules (interesting rules)**, because they have both high support and confidence.
  - Support value: 0.6
  - Confidence value: 1.0
- The way to find more strong rules would be to examine ***every possible combination of the items*** in the gift shop, measure the support and confidence value, and report back only those rules that meet certain levels of interest.

The Apriori Algorithm for Association Rule Learning

- Transactional datasets are typically extremely large, both in terms of the number of transactions as well as the number of items or features that are monitored.
  - Number of potential itemsets grows exponentially with the number of items.
  - Given *k* items that can appear or not appear in a set, there are *2^k* possible itemsets that could be potential rules.
    - A retailer that sells only 100 different items could have on the order of *2^100 = 1.27e+30* itemsets that an algorithm must evaluate—a seemingly impossible task
    - *Example: milk, chocolate cookies, cheese*

The Apriori Algorithm for Association Rule Learning

- **Apriori property:** all ***subsets*** of a ***frequent itemset*** must also be ***frequent***.
  - For example, the set {*get well card, flowers*} can only be frequent if both {*get well card*} and {*flowers*} occur frequently as well.
  - Consequently, if either *get well card* or *flowers* is infrequent, any set containing these items can be excluded from the search.

Apriori Algorithm

- Consider a supermarket scenario
  - Sell five items: Onion, Burger, Potato, Milk, Beer.
  - The database consists of six transactions where 1 represents the presence of the item and 0 the absence.

Transaction ID	Onion	Potato	Burger	Milk	Beer
t_1	1	1	1	0	0
t_2	0	1	1	1	0
t_3	0	0	0	1	1
t_4	1	1	0	1	0
t_5	1	1	1	0	1
t_6	1	1	1	1	0

The Apriori Algorithm makes the following assumptions.

- All subsets of a frequent itemset should be frequent.
- The itemset containing infrequent items should be infrequent.
- Set a threshold support level. In our case, we shall fix it at 50%.

Apriori Algorithm

- **Step 1.** Create a frequency table of all the items that occur in all the transactions.
- **Step 2.** Here, support threshold is 50%, hence only those items are frequent which occur in equal or more than three transactions and such items are Onion(O), Potato(P), Burger(B), and Milk(M).
- **Step 3.** make all the possible pairs of the significant items (the order doesn't matter)

1-item itemsets

Item	Frequency (No. of transactions)
Onion(O)	4
Potato(P)	5
Burger(B)	4
Milk(M)	4
Beer(Be)	2

Generate frequent items

1-item itemsets that meet the minimum support threshold

Item	Frequency (No. of transactions)
Onion(O)	4
Potato(P)	5
Burger(B)	4
Milk(M)	4

2-item itemsets

Itemset	Frequency (No. of transactions)
OP	4
OB	3
OM	2
PB	4
PM	3
BM	2

- **Step 4.** Only those itemsets are significant which cross the support threshold (OP, OB, PB, and PM)
- **Step 5.** Now we would like to look for 3-item itemsets that are purchased together. We will use the item sets found in step 4 and create 3-item itemsets.
- **Step 6.** Applying the threshold rule again, we find that OPB is the only frequent 3-item itemset.

2-item itemsets

Itemset	Frequency (No. of transactions)
OP	4
OB	3
OM	2
PB	4
PM	3
BM	2

2-item itemsets that meet the minimum support threshold

Itemset	Frequency (No. of transactions)
OP	4
OB	3
OM	2
PB	4
PM	3
BM	2

3-item itemsets

Itemset	Frequency (No. of transactions)
OPB	4

How about OPM, OBM, and PBM?

- OM and BM does not meet a desired support threshold

The Apriori Algorithm for Association Rule Learning

- Frequent itemsets
  - O, P, B, M, OP, OB, PB, PM, and OPB
- From frequent itemsets to association rules
  - O -> P, P -> O
  - O -> B, B -> O
  - P -> B, B -> P
  - P -> M, M -> P
  - {O,P} -> B, {B,P} -> O, {O,B} -> P, B -> {O,P}, O -> {B,P}, P -> {O,B}
- Exam the confidence value to generate **strong rules**

The Apriori Algorithm for Association Rule Learning

- Building a set of rules with the Apriori principle
  - Apriori algorithm creates rules in two phase:
    1. Identifying all the itemsets that meet a minimum support threshold.
    2. Creating rules from these itemsets using those meeting a minimum confidence threshold.

The Apriori Algorithm for Association Rule Learning

- The first phase occurs in multiple iterations.
- Each successive iteration involves evaluating the support of a set of increasingly large itemsets.
  - Iteration 1 involves evaluating the set of 1-item itemsets (1 itemsets)
  - Iteration 2 evaluates 2-itemsets, and so on.
- The result of each iteration *i* is a set of all the frequent i-itemsets that meet the minimum support threshold.
- All the freqnent itemsets from iteration *i* are combined in order to generate candidate itemsets for the evaluation in iteration *i + 1*.

The Apriori Algorithm for Association Rule Learning

- The second phase of the Apriori algorithm may begin.
  - Given the set of frequent itemsets, association rules are generated from all possible subsets.
    - For instance, {A, B} would result in candidate rules for {A} → {B} and {B} → {A}.
  - These are evaluated against a minimum confidence threshold, and any rule that does not meet the desired confidence level is eliminated.

The Apriori Algorithm for Association Rule Learning

Strengths	Weaknesses
• Is capable of working with large amounts of transactional data • Results in rules that are easy to understand • Useful for "data mining" and discovering unexpected knowledge in databases	• Not very helpful for small datasets • Requires effort to separate the true insight from common sense • Easy to draw spurious conclusions from random patterns

Identifying Frequently Purchased Groceries with Association Rules

- Market basket analysis on grocery data
  - We will utilize the purchase data collected from one month of operation at a real-world grocery store. The data contains 9,835 transactions.
    - Transactional data is stored in a slightly different format than that we used previously. Transactional data is a more free form.
      - Each row in the data specifies a single transaction.
      - Rather than having a set number of features, each record comprises a comma-separated list of any number of items, from one to many.
      - The items may differ from example to example

Identifying Frequently Purchased Groceries with Association Rules

- The first five rows of the raw grocery.csv file are as follows:

	A	B	C	D	E
1	citrus fruit	semi-finished bread	margarine	ready soups	
2	tropical fruit	yogurt	coffee		
3	whole milk				
4	pip fruit	yogurt	cream cheese	meat spreads	
5	other vegetables	whole milk	condensed milk	long life bakery product	

- These lines indicate five separate grocery store transactions.

Identifying Frequently Purchased Groceries with Association Rules

```
> summary(groceries)
transactions as itemMatrix in sparse format with
 9835 rows (elements/itemsets/transactions) and
 169 columns (items) and a density of 0.02609146

most frequent items:
      whole milk other vegetables       rolls/buns             soda           yogurt          (Other)
            2513             1903             1809             1715             1372            34055

element (itemset/transaction) length distribution:
sizes
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24   26   27   28
2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55   46   29   14   14    9   11    4    6    1    1    1    1
  29   32
   3    1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   3.000   4.409   6.000  32.000
```

The output 9835 rows refers to the number of transactions

The output 169 columns refers to the 169 different items that might appear in someone's grocery basket.

Items that were most commonly found in the transactional data

A total of 2,159 transactions contained only a single item, while one transaction had 32 items

Identifying Frequently Purchased Groceries with Association Rules

- groceries_rules <- apriori(groceries, parameter = list(support = 0.01, confidence = 0.3, maxlen=2))
  - Find the support and confidence parameters that produce a reasonable number of association rules
    - Set these levels too high: find no rules or rules that are too generic to be very useful
    - Set the threshold too low: an unwieldy number of rules; the operation might take a very long time or run out of memory during the learning phase.

Identifying Frequently Purchased Groceries with Association Rules

- One way to approach the problem of setting a minimum support threshold
  - Think about the smallest number of transactions you would need before you would consider a pattern interesting.
    - For instance, you could argue that if an item is purchased twice a day (about 60 times in a month of data), it may be an interesting pattern.
    - 60 out of 9,835 equals 0.006
- Setting the minimum confidence involves a delicate balance.
  - Confidence is too low: overwhelmed with a large number of unreliable rules
  - set confidence too high: limited to the rules that are obvious or inevitable

Identifying Frequently Purchased Groceries with Association Rules

- To obtain a high-level overview of the association rules, we can use summary() function.
  - The **rule length distribution** tells us how many rules have each count of items.
  - In our rule set, all rules have only two items.

```
> summary(groceries_rules)
set of 69 rules

rule length distribution (lhs + rhs):sizes
 2
69

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      2       2       2       2       2       2

summary of quality measures:
    support           confidence          lift            count
 Min.   :0.01007   Min.   :0.3079   Min.   :1.205   Min.   : 99.0
 1st Qu.:0.01373   1st Qu.:0.3361   1st Qu.:1.521   1st Qu.:135.0
 Median :0.01973   Median :0.3737   Median :1.645   Median :194.0
 Mean   :0.02292   Mean   :0.3776   Mean   :1.707   Mean   :225.4
 3rd Qu.:0.02888   3rd Qu.:0.4108   3rd Qu.:1.814   3rd Qu.:284.0
 Max.   :0.07483   Max.   :0.4972   Max.   :3.040   Max.   :736.0

mining info:
      data ntransactions support confidence
 groceries          9835    0.01        0.3
```

Identifying Frequently Purchased Groceries with Association Rules

- We see the summary statistics of the rule quality measures: **support, confidence, and lift**.
- We might be alarmed if most or all of the rules had support and confidence very near the minimum thresholds
  - This would mean that we may have set the bar too high.
  - Not the case here: there are many rules with much higher values of each measure.

```
> summary(groceries_rules)
set of 69 rules

rule length distribution (lhs + rhs):sizes
 2 
69 

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
      2       2       2       2       2       2 

summary of quality measures:
    support          confidence          lift           count      
 Min.   :0.01007   Min.   :0.3079   Min.   :1.205   Min.   : 99.0  
 1st Qu.:0.01373   1st Qu.:0.3361   1st Qu.:1.521   1st Qu.:135.0  
 Median :0.01973   Median :0.3737   Median :1.645   Median :194.0  
 Mean   :0.02292   Mean   :0.3776   Mean   :1.707   Mean   :225.4  
 3rd Qu.:0.02888   3rd Qu.:0.4108   3rd Qu.:1.814   3rd Qu.:284.0  
 Max.   :0.07483   Max.   :0.4972   Max.   :3.040   Max.   :736.0  

mining info:
      data ntransactions support confidence
 groceries          9835    0.01        0.3
```

Identifying Frequently Purchased Groceries with Association Rules

- The first rule can be read in plain language as, "if a customer buys butter, he/she will also buy whole milk."

```
> # Display top 5 rules
> inspect(sort(groceries_rules, by = "confidence")[1:5])
    lhs                     rhs                support    confidence lift     count
[1] {butter}             => {whole milk}       0.02755465 0.4972477  1.946053 271
[2] {curd}               => {whole milk}       0.02613116 0.4904580  1.919481 257
[3] {domestic eggs}      => {whole milk}       0.02999492 0.4727564  1.850203 295
[4] {onions}             => {other vegetables} 0.01423488 0.4590164  2.372268 140
[5] {whipped/sour cream} => {whole milk}       0.03223183 0.4496454  1.759754 317
```

Identifying Frequently Purchased Groceries with Association Rules

- Support for association rule X->Y: $support(X,Y) = \frac{count(X,Y)}{N}$

- Confidence is defined as the support of the itemset containing both *X* and *Y* divided by the support of the itemset containing only *X*:

$$\text{confidence}(X \rightarrow Y) = \frac{\text{support}(X,Y)}{\text{support}(X)}$$

- The lift of a rule measures: how much more likely one item or itemset is purchased relative to its typical rate of purchase, given that you know another item or itemset has been purchased.
  - Unlike confidence where the item order matters, lift(X → Y) is the same as lift(Y → X).

$$\text{lift}(X \rightarrow Y) = \frac{\text{confidence}(X \rightarrow Y)}{\text{support}(Y)} = \frac{support(X,Y)}{support(X) * support(Y)}$$

Identifying Frequently Purchased Groceries with Association Rules

```
    lhs                        rhs             support    confidence lift     count
[1] {butter}                => {whole milk}    0.02755465 0.4972477  1.946053 271
```

0.2555160142 (2513/9835)   0.0554143366 (545/9835)

- Support: 0.02755465 (271/9835)
  - The probability of buying butter and whole milk together is 0.02755465
  - 2.755% transactions include both butter and whole milk. This rule covers 2.755% transactions.
- Confidence: 0.4972477 (271/545)
  - When customers bought butter, they will also buy whole milk about 49.72477% of the time.
  - Among transactions including butter, 49.72477% of them also include whole milk.
- Lift: 1.946053 (0.02755465 /(0.0554143366*0.2555160142))
  - Butter and whole milk have positive effect (because lift value greater than 1) on each other.
  - Purchasing butter increases the probability of buying whole milk by 1.946 times.
  - Purchasing whole milk increases the probability of buying butter by 1.946 times.

Identifying Frequently Purchased Groceries with Association Rules

In spite of the fact that the confidence and lift are high, does {butter} → {whole milk} seem like a very useful rule?

Identifying Frequently Purchased Groceries with Association Rules

- A common approach is to take the association rules and divide them into the following three categories:
  - Actionable
  - Trivial
  - Inexplicable
- Obviously, the goal of a market basket analysis is to find actionable rules that provide a clear and useful insight.
  - Some rules are clear, others are useful
  - it is less common to find a combination of both of these factors.

Identifying Frequently Purchased Groceries with Association Rules

- **Trivial rules** include any rules that are so obvious that they are not worth mentioning
  - They are clear, but not useful.
  - {diapers} $\rightarrow$ {formula}
- Rules are **inexplicable** if the connection between the items is so unclear that figuring out how to use the information is impossible or nearly impossible.
  - Simply be a random pattern in the data

Identifying Frequently Purchased Groceries with Association Rules

- The best rules are hidden gems—those undiscovered insights into patterns that seem obvious once discovered.
  - One could evaluate each and every rule.
    - Not the best judge of whether a rule is actionable, trivial, or inexplicable.
  - Improve the utility of our work
    - Employ methods to sort and share the learned rules
    - The most interesting results might float to the top

Identifying Frequently Purchased Groceries with Association Rules

```
> # Display top 5 rules
> inspect(sort(groceries_rules, by = "confidence")[1:5])
    lhs                    rhs                support    confidence lift     count
[1] {butter}            => {whole milk}       0.02755465 0.4972477  1.946053 271
[2] {curd}              => {whole milk}       0.02613116 0.4904580  1.919481 257
[3] {domestic eggs}     => {whole milk}       0.02999492 0.4727564  1.850203 295
[4] {onions}            => {other vegetables} 0.01423488 0.4590164  2.372268 140
[5] {whipped/sour cream} => {whole milk}      0.03223183 0.4496454  1.759754 317
```

Identifying Frequently Purchased Groceries with Association Rules

```
> inspect(sort(groceries_rules, by = "lift"))
     lhs                        rhs                support    confidence lift     count
[1]  {beef}                  => {root vegetables}  0.01738688 0.3313953  3.040367 171
[2]  {onions}                => {other vegetables} 0.01423488 0.4590164  2.372268 140
[3]  {curd}                  => {yogurt}           0.01728521 0.3244275  2.325615 170
[4]  {berries}               => {yogurt}           0.01057448 0.3180428  2.279848 104
[5]  {root vegetables}       => {other vegetables} 0.04738180 0.4347015  2.246605 466
[6]  {cream cheese}          => {yogurt}           0.01240468 0.3128205  2.242412 122
[7]  {chicken}               => {other vegetables} 0.01789527 0.4170616  2.155439 176
[8]  {hamburger meat}        => {other vegetables} 0.01382816 0.4159021  2.149447 136
[9]  {whipped/sour cream}    => {other vegetables} 0.02887646 0.4028369  2.081924 284
[10] {butter}                => {whole milk}       0.02755465 0.4972477  1.946053 271
[11] {beef}                  => {other vegetables} 0.01972547 0.3759690  1.943066 194
[12] {pork}                  => {other vegetables} 0.02165735 0.3756614  1.941476 213
```

Identifying Frequently Purchased Groceries with Association Rules

- Suppose the marketing team is excited about the possibilities of creating an advertisement to promote berries.
- Before finalizing the campaign, however, they ask you to investigate whether berries are often purchased with other items.
- To answer this question, we'll need to find all the rules that include berries in some form.
- The `subset()` function provides a method to search for subsets rules.

```
> vegetables_rules <- subset(groceries_rules2, items %in% "other vegetables")
> # Display rules containing "other vegetables"
> inspect(vegetables_rules)
    lhs                                rhs                support    confidence lift     count
[1] {whipped/sour cream}            => {other vegetables} 0.02887646 0.4028369  2.081924 284
[2] {root vegetables}               => {other vegetables} 0.04738180 0.4347015  2.246605 466
[3] {other vegetables,root vegetables} => {whole milk}    0.02318251 0.4892704  1.914833 228
[4] {root vegetables,whole milk}    => {other vegetables} 0.02318251 0.4740125  2.449770 228
[5] {other vegetables,yogurt}       => {whole milk}       0.02226741 0.5128806  2.007235 219
```

Identifying Frequently Purchased Groceries with Association Rules

- The criteria for choosing the subset can be defined with several keywords and operators:
  - `items` matches an item appearing anywhere in the rule.
    - To limit the subset to where the match occurs only on the left- or right-hand side, use `lhs` and `rhs` instead.

```
> vegetables_rules_l <- subset(groceries_rules2, lhs %in% "other vegetables")
> inspect(vegetables_rules_l)
    lhs                                      rhs          support    confidence lift     count
[1] {other vegetables,root vegetables} => {whole milk} 0.02318251 0.4892704  1.914833 228
[2] {other vegetables,yogurt}          => {whole milk} 0.02226741 0.5128806  2.007235 219
> # Find and display rules containing "other vegetables" on the right-hand side
> vegetables_rules_r <- subset(groceries_rules2, rhs %in% "other vegetables")
> inspect(vegetables_rules_r)
    lhs                                rhs                support    confidence lift     count
[1] {whipped/sour cream}          => {other vegetables} 0.02887646 0.4028369  2.081924 284
[2] {root vegetables}             => {other vegetables} 0.04738180 0.4347015  2.246605 466
[3] {root vegetables,whole milk} => {other vegetables} 0.02318251 0.4740125  2.449770 228
```