

2020-1 Data Analysis with Applications

Lecture 16. Recommender Systems

Department of Industrial and Information Systems Engineering,
Soongsil University

Introduction to Recommender Systems

Recommender Systems

- 추천시스템 (recommender systems)은 사용자들의 히스토리를 기반으로 상품, 영화, 음악, 책, 웹 페이지, 뉴스 기사 등을 추천해주는 시스템을 말한다.
- Netflix의 영화 추천, amazon의 상품 추천, instagram의 검색 추천, google의 웹페이지 추천 등을 예로 들 수 있다.
- 최근에는 온라인 기반의 상품 판매 및 서비스 제공 기업의 수익 증대 및 고객 관리를 위한 핵심적인 시스템으로 자리잡고 있다.



[책 구매 히스토리를 활용한 책 추천 예]

Recommender Systems

- 전통적인 추천시스템은 크게 association rule 기반의 추천과 collaborative filtering 기반의 추천으로 나눌 수 있다.
- Association rule 방식은 데이터베이스에 저장된 대량의 구매 기록으로부터 어떠한 상품들이 동시에 구매되는지 패턴을 찾아서 상품 기반의 추천을 수행한다.
- Collaborative filtering 방식은 사용자에 대한 개별 추천이 목적이며, 추천 대상자와 유사한 상품들을 구매한 다른 사용자가 구매한 상품을 추천하거나, 동일한 영화에 유사한 평점을 부여한 다른 사용자가 관람한 영화를 추천하는 방식으로 작동한다.

The image shows a product page for the Apple iPhone 11 Pro. The main product image is on the left, showing the back and front of the phone. The product title is "Apple iPhone 11 Pro 공기계". It has a rating of 4.5 stars from 5,076 reviews. The price is listed as 1,390,000원 with free delivery options. Below the main image are two smaller images: one of the phone's back and another of the phone with a clear case. To the right, there is a section titled "다른 고객이 함께 구매한 상품" (Products bought by other customers) with three items listed:

- Apple iPhone 11 Pro 클리어 케이스
9,000원
4.5 stars (97 reviews)
- 링케 퓨전 휴대폰 케이스
9,000원
4.5 stars (216 reviews)
- 슈피겐 크리스탈하이브리드 휴대폰 케이스 077CS27114
18,400원
4.5 stars (105 reviews)

[Association rule 방식의 상품 추천 예]

Association Rules

Association Rules

- Association rule을 사용하는 분석은 market basket analysis라고도 부르는데, 슈퍼마켓 구매 데이터나 온라인 쇼핑 장바구니 데이터에 자주 활용되기 때문이다.
- 동시에 빈번하게 구매되는 상품의 그룹을 찾는 것이 목적이며, 이를 활용하여 온라인 쇼핑에서의 ‘함께 구입한 상품’ 추천에 활용되거나, 오프라인 매장에서 상품 진열대 레이아웃 디자인, 상품 마케팅 전략 수립에 활용될 수 있다.
- 쇼핑 데이터 뿐만 아니라, 의료 데이터로부터 증상(symptom)의 패턴 찾기, 문서 데이터로부터 표절로 의심되는 단어의 조합 찾기 등 다양한 분야에 활용될 수 있다.



Transaction Database

- 핸드폰 악세사리 판매점에서 핸드폰 케이스에 대한 프로모션을 진행 중인데, 6개의 색을 가지는 케이스를 여러 개 구입할 경우 할인을 받을 수 있다. 이때 고객들이 어떤 색의 케이스를 같이 구매하는지 패턴을 알아보고자 한다.
- 다음과 같은 가상의 transaction database를 사용하여 association rule 분석을 진행해보자.
↳ 고객들이 구매한 기록을 하나하나.
↳ 하나하나를 item.

Transaction	구매 목록			
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			

Itemsets and Rules

Item set $\rightarrow 2^n$ 개 (不尽枚举)

- 다음 예와 같은 서로 다른 item들의 집합을 **itemset**이라 한다.

{red, white, green}

- Itemset을 IF part와 THEN part로 분할하여 **rule**을 생성한다. “red와 white를 구매하면 green도 구매한다.”로 해석할 수 있다.

if red, white then green

- 가능한 모든 rule 중에서 item들 사이의 연관 관계를 나타낼 수 있는 rule을 찾아야 한다. 하지만 모든 item들의 조합은 매우 많기 때문에, item의 수가 많을 때 모든 rule을 다 고려하는 것은 계산적으로 거의 불가능하다.

- Item의 수가 p 개일 때, 가능한 모든 rule의 수는 $3^p - 2^{p+1} + 1$ 개이며, 예제에서 $p = 6$ 일 때 rule의 수는 602개이다.
- 현실적인 방법은 transaction database에서 자주 발생하는 itemset (**frequent itemsets**)에 대해서 rule을 생성하는 것이다.

The Apriori Algorithm

- **Apriori algorithm**은 transaction database로부터 frequent itemset을 찾기 위해 전통적으로 사용되는 **heuristic algorithm**이다.
→ 초기화 단계에서 가능한 조건들을 찾는다.
- **IDEA:** Itemset $\{A, B\}$ 가 자주 발생한다면, $\{A\}$ 와 $\{B\}$ 도 자주 발생해야 한다. 즉, $\{A\}$ 또는 $\{B\}$ 가 자주 발생하지 않는다면 $\{A, B\}$ 도 자주 발생하지 않는다.
- 따라서 크기가 $k - 1$ 인 frequent itemset으로부터 크기가 k 인 frequent itemset을 반복적으로 찾는 **recursive algorithm**의 형태를 가진다.

*→ 그 다음 item set에서 조건을 찾는 algorithm
= 초기화 단계*

- Itemset이 자주 발생하는 정도를 나타내는 척도를 **support**라고 하며, itemset을 포함하는 transaction의 수 또는 비율로 정의된다.

= supp. (발생하는 transaction 수)

(red, white, blue)

*↓
(R, W, B) ↗ R, B, W*

Itemset	Support
{red}	6
{white}	7
{blue}	6
{orange}	2
{green}	2
{red, white}	4
{red, blue}	4
{red, green}	2
{white, blue}	4
{white, orange}	2
{white, green}	2
{red, white, blue}	2
{red, white, green}	2

The Apriori Algorithm

C : 가능한候选集合 candidate set.

候选 item candidate
itemset의 집합 C
의 집합을 L_1 이라
자주 \approx

- Apriori algorithm에서는 먼저 하나의 item을 가지는 itemset의 집합 C_1 에서 최소 support 조건을 만족하는 itemset을 모두 찾는다. 이들의 집합을 L_1 이라 하자.
 ↗ 자주 발생하는 item
 - 다음으로 L_1 에 속한 itemset으로 만들 수 있는 크기가 2인 itemset들의 집합 C_2 를 만든 후, C_2 의 부분집합 중 L_1 에 속하지 않는 것이 있다면 제거한다. C_2 의 itemset 중에서 최소 support 조건을 만족하는 itemset을 찾아 집합 L_2 를 만든다.
 - 위의 과정을 더 이상 최소 support 조건을 만족하는 itemset이 발견되지 않을 때까지 반복한다. $L_1 \cup L_2 \cup \dots$ 이 최종적으로 생성되는 frequent itemset의 집합이 된다.
 ↗ 핵심!!
 - L_{k-1} 로부터 C_k 를 만드는 과정은 다음과 같다.

1) $\{i_1, i_2, \dots, i_{k-1}\} \in L_{k-1}$, $\{j_1, j_2, \dots, j_{k-1}\} \in L_{k-1}$ 일 때, $i_1 = j_1, \dots, i_{k-2} = j_{k-2}$ 0 | 고

$i_{k-1} \neq j_{k-1}$ 이면 $\{i_1, i_2, \dots, i_{k-1}, j_{k-1}\} \in C_k$ 이다. 이것이 풀리면 됩니다!!

2) Itemset $S \in C_k$ 에 대해 크기가 $k - 1$ 인 S 의 부분집합 중 L_{k-1} 에 속하지 않는 것이 있다면 S 를 C_k 에서 제외한다.

{A, B}
{A, C}

L₇

A, B, C}
C₃

- {A, B, C}의 부분집합인 {B,C}는 L_2 에 속하지 않기 때문에 {A, B, C}는 C_3 에서 제거한다.

✓ BC 흙막이 3m ↑ BC는 2001 흙막이망원경. 2, CMB를 관찰한다.

The Apriori Algorithm

- 핸드폰 케이스 예제에서 최소 support 기준이 2일 때 Apriori algorithm을 적용해보자.

C_1	Support	L_1
{red}	6	0
{white}	7	0
{blue}	6	0
{orange}	2	0
{green}	2	0
{yellow}	1	X

frequent itemset

①
②

최소기준(2)
2m 선택

bm(L_1)로 부터 (2개)

C_2	Support	L_2
{red, white}	4	0
{red, blue}	4	0
{red, orange}	1	X
{red, green}	2	0
{white, blue}	4	0
{white, orange}	2	0
{white, green}	2	0
{blue, orange}	0	X
{blue, green}	1	X
{orange, green}	0	X

⑥
bm 블록화

- {yellow}는 support < 2이므로 L_1 에 포함되지 않음

{red}, {blue, red}

{white}, {blue, white}

중복되는 경우, red, white가 다른 경우,

- 크기가 1인 5개의 itemset (L_1)으로부터 크기가 2인 10개의 itemset (C_2)을 만들고, 그중에서 support가 2 이상인 6개의 itemset (L_2)이 선택되었다.

The Apriori Algorithm

(2)

C_3	$Support$	L_3
{red, white, blue}	2	0
{red, white, green}	2	0
{red, blue, green}	1	X
{white, red, orange}	{red, orange} not in L_2	
{white, blue, orange}	{blue, orange} not in L_2	
{white, blue, green}	{blue, green} not in L_2	
{white, orange, green}	{orange, green} not in L_2	

→

C_4	$Support$	L_4
{red, white, blue, green}	{red, blue, green} not in L_3	

- {white, red}와 {white, orange}는 L_2 에 속하기 때문에 {white, red, orange}는 C_3 의 원소로 고려되지만, 이 itemset의 부분집합인 {red, orange}은 크기가 2이지만 L_2 에 속하지 않으므로 {white, red, orange}는 C_3 에 포함될 수 없다.

- {red, white, blue}와 {red, white, green}으로부터 {red, white, blue, green}을 생성했지만, 이 itemset의 부분집합인 {red, blue, green}은 크기가 3이지만 L_3 에 속하지 않으므로 {red, white, blue, green}은 C_4 에 포함될 수 없다.
- L_4 는 공집합이므로 알고리즘을 종료한다. 빨강으로 표시된 itemset들이 최종적으로 생성된 frequent itemset들이다.

$$5+6+2 = 13 \geq n$$

Selecting Strong Rules

- Apriori algorithm으로 생성된 frequent itemset으로부터 강한 연관관계를 가지는 rule을 찾아야 한다.

- 연관관계의 정도를 나타내기 위해 **confidence**와 **lift**의 척도를 사용한다.

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

↑
P(X,Y)로 바꾸었다. 즉 P(X)가 P(X,Y)에 포함된다.
→ 조건부 확률
↑
P(X)

- $\text{confidence}(X \rightarrow Y)$ 은 itemset X를 구매할 경우 itemset Y를 같이 구매할 조건부 확률 $P(Y|X)$ 의 추정값으로 해석할 수 있다.
 $= \frac{P(X \cap Y)}{P(X)}$
- $\text{confidence}(X \rightarrow Y)$ 은 $\text{confidence}(Y \rightarrow X)$ 와 의미가 다르기 때문에 같은 값을 가지지 않을 수 있다.

Rule	Confidence
$\frac{\text{red, white, green}}{X} \rightarrow \text{green}$	$\frac{\text{red, white, green이 히트되는 경우}}{\text{red, white 히트되는 경우}} = \frac{2}{4} = 0.5$
$\text{green} \rightarrow \text{red, white}$	$\frac{\text{support(red, white, green)}}{\text{support(green)}} = \frac{2}{2} = 1$

↳ green을 가지면 red, white를 찾는다. → 2개 모두가 가능하다.

Selecting Strong Rules

X와 Y가 연관관계가 있다면!

- 만약 itemset X와 Y가 독립이면, 다음 식이 성립한다.

$$P(Y|X) = \frac{P(X, Y)}{P(X)} = \frac{P(X) \cdot P(Y)}{P(X)} = P(Y)$$

= X와 Y가 연관이 없으면
X와 Y는 서로 독립적일 때도 예의 발생률은 같다.

- 따라서 itemset X와 Y가 서로 연관이 없을 때 $P(Y|X)$ 의 추정값은 다음과 같다.

X와 Y는 연관관계가 없다.

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

- Rule $X \rightarrow Y$ 의 lift는 X와 Y가 서로 연관이 없을 때와 비교해서 confidence 값이 얼마나 큰지를 평가하는 척도이다.

그것은 가능성이 있다.

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

연관이 없는 경우

가장 발생률을 높여 Y가 발생할 확률

연관이 있는 경우

- Rule $X \rightarrow Y$ 의 lift 값이 높을수록 더 큰 연관관계를 가진다고 볼 수 있다. 그리고 confidence와는 달리 $\text{lift}(X \rightarrow Y)$ 와 $\text{lift}(Y \rightarrow X)$ 는 같은 값을 가진다.

Rule	Lift
$\{\text{red, white}\} \rightarrow \{\text{green}\}$	$\text{confidence} / \text{support}(\text{green}) = 0.5 / 0.2 = 2.5$ $= \text{red, white} \rightarrow \text{green} \approx 2.5$ 그동안 green보다 2.5배 높다.
$\{\text{green}\} \rightarrow \{\text{red, white}\}$	$\text{confidence} / \text{support}(\text{red, white}) = 1 / 0.4 = 2.5$

The Process of Rule Selection

- Association rule 기법은 다음 두 단계로 이루어진다.

→ 1단계

- Apriori algorithm으로 최소 support 기준을 만족하는 frequent itemset들을 생성한다.

2단계 ex) red, white
w → R) 2M × bz | 2M

- 생성된 Frequent itemset으로 만들 수 있는 rule 중에서 최소 confidence 기준을 만족하는 rule을 선택한다.

→ 1단계는 이미가 많으므로
(→ 를 만들다.)
2단계 ex) R → W, B B → R, W R, B → W
B → R, B R, W → B B, W → R)

- 아래 목록은 핸드폰 케이스 예제에서 $|Y| = 1$ 인 rule 중에서 support 0.2 이상, confidence 0.5 이상인 rule들을 보여준다.

Rules	Support	Confidence
{red, white} → {green}	0.2	0.5
{green} → {red}	0.2	1
{white, green} → {red}	0.2	1
{orange} → {white}	0.2	1
{green} → {white}	0.2	1
{red, green} → {white}	0.2	1

제거의 Item이 동시에 발생하는 경우 20%

red, green → white

R, green을 선택할 때 W를 빼야

Identifying Frequently Purchased Groceries

Identifying Frequently Purchased Groceries

- “groceries.csv” 파일은 실제 grocery store에서의 한 달동안의 구매 데이터로부터 생성되었으며, 9,835개의 transaction을 포함하고 있다. 이것은 시간 당 대략 30건의 transaction이 발생한 것과 같다.
- 각 상품의 브랜드는 고려하지 않고 품목 간의 rule을 생성하기 위해 구매데이터에서 브랜드명은 삭제되었다. 따라서 transaction 데이터는 chicken, frozen meals, margarine, soda와 같은 169개의 품목으로 구성된다.
- Association rule 기법의 사용을 위해서 arules 패키지를 사용하자.
- 먼저 read.transaction() 함수를 사용하여 transaction 데이터를 sparse matrix 형태로 저장한다.

```
# arules 패키지 사용
library(arules)

# 데이터 읽기 (sparse matrix format 사용)
groceries <- read.transactions("groceries.csv", sep=",")
```

Sparse Matrix Format

	abrasive cleaner	artif. sweetner	baby cosmetics	baby food	bags	baking powder	bathroom cleaner	beef
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	1	1	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0

[Sparse matrix format]

```
summary(groceries)
```

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

- **density 0.026** : matrix의 전체 entry 중에 약 2.6%만이 0이 아닌 값을 가진다.

Sparse Matrix Format

```
## most frequent items:  
##      whole milk other vegetables      rolls/buns      soda  
##          2513             1903            1809           1715  
##      yogurt          (Other)  
##          1372            34055  
##  
## element (itemset/transaction) length distribution:  
## sizes  
##   1    2    3    4    5    6    7    8    9    10   11   12   13   14   15   16  
## 2159 1643 1299 1005  855  645  545  438  350  246  182  117  78  77  55  46  
##   17   18   19   20   21   22   23   24   26   27   28   29   32  
##   29   14   14    9   11    4    6    1    1    1    1    3    1  
##  
##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.  
##     1.000  2.000  3.000  4.409  6.000  32.000
```

- whole milk, other vegetables, rolls/buns, soda, yogurt의 순서대로 transaction 데이터에서 가장 빈번하게 발생한다.
- 전체 transaction 중에서 2159개는 하나의 item만 포함하며, 1643개는 2개의 item을 포함한다. 가장 많은 item을 포함하는 transaction은 32개의 item을 포함한다.
- 그리고 하나의 transaction은 평균 4.4개의 item으로 구성된다.

Sparse Matrix Format

```
# transaction의 세부 item 확인
```

```
inspect(groceries[1:3,])
```

```
##      items
## [1] {citrus fruit,
##       margarine,
##       ready soups,
##       semi-finished bread}
## [2] {coffee,
##       tropical fruit,
##       yogurt}
## [3] {whole milk}
```

```
# transaction과 item의 submatrix 확인
```

```
inspect(groceries[1:5, 1:50])
```

```
##      items
## [1] {citrus fruit}
## [2] {coffee}
## [3] {}
## [4] {cream cheese}
## [5] {condensed milk}
```

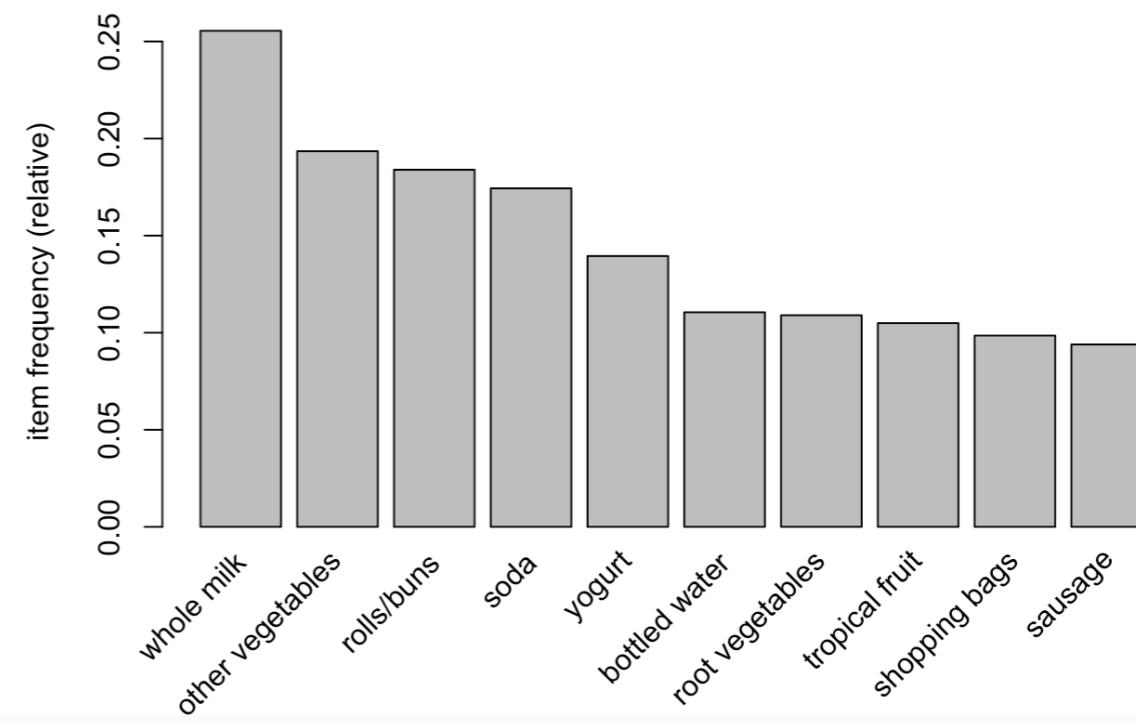
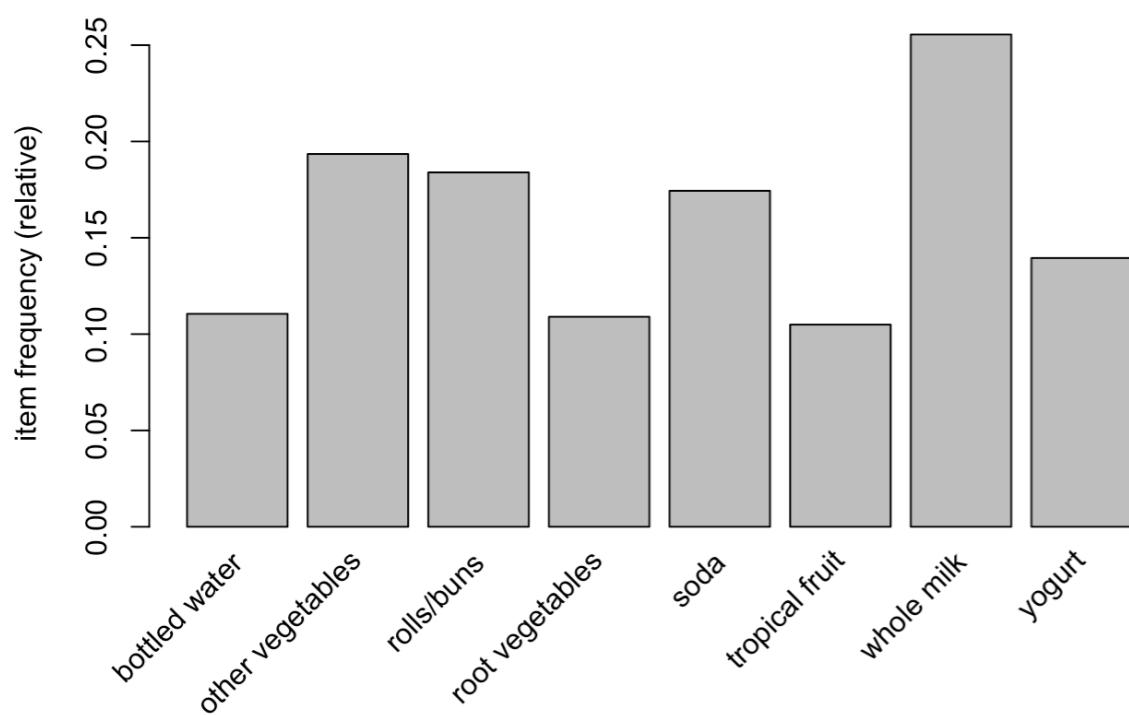
```
# item의 frequency (발생빈도) 확인
```

```
itemFrequency(groceries[, 1:4])
```

	abrasive	cleaner	artif.	sweetener	baby	cosmetics	baby food
##	0.0035587189	0.0032536858			0.0006100661		0.0001016777

Visualizing Frequencies

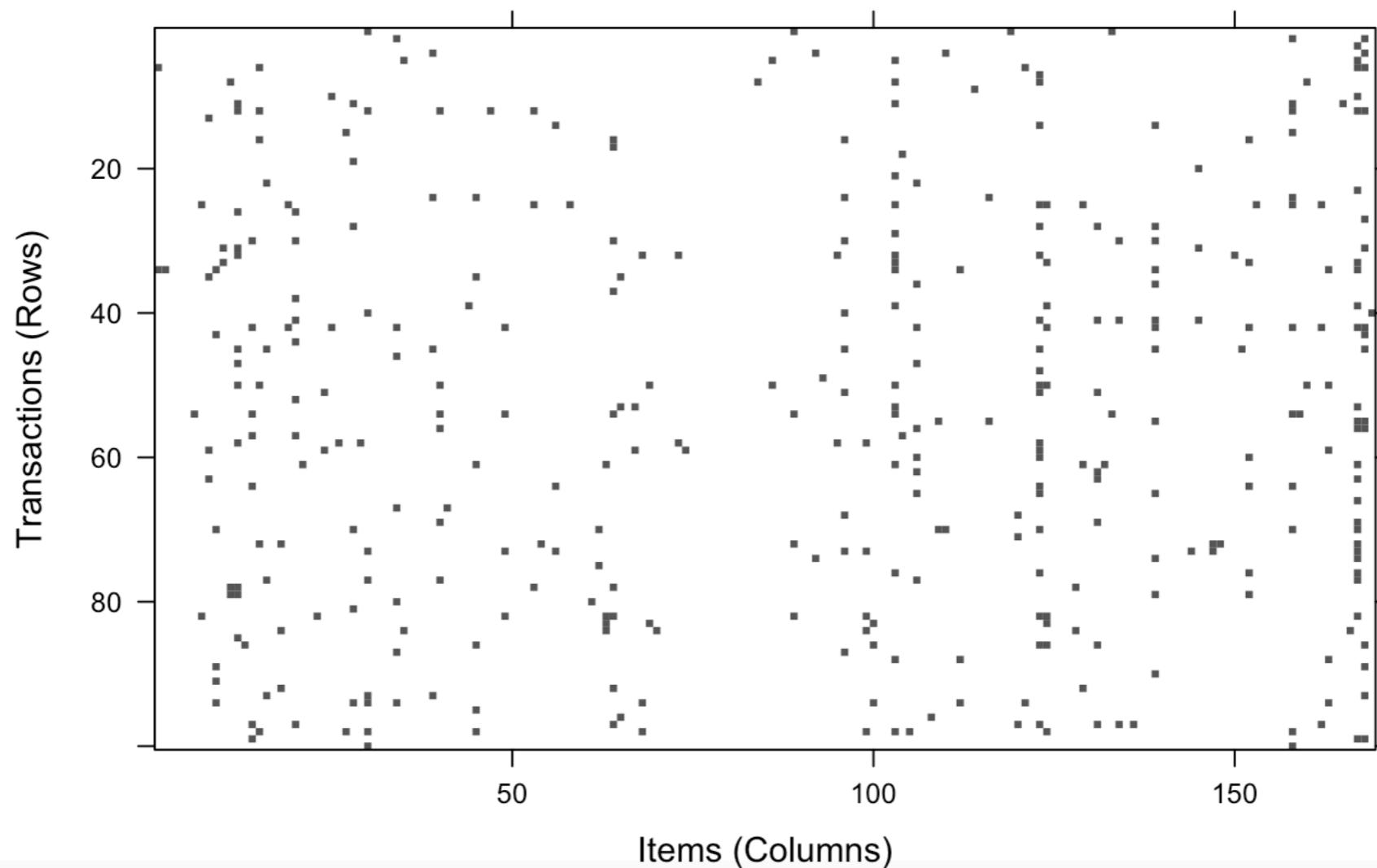
```
# support가 0.1 이상인 item 시각화  
itemFrequencyPlot(groceries, support=0.1)  
  
# 발생빈도가 가장 높은 10개 item 시각화  
itemFrequencyPlot(groceries, topN = 10)
```



Visualizing Sparse Matrix

```
# sparse matrix 시각화  
image(groceries[1:100])
```

Column Data 정렬!



apriori()

- apriori() 함수를 이용하여 association rule 분석을 수행할 수 있으며 다음 옵션을 설정할 수 있다.
 - support: itemset의 최소 support 값 (default = 0.1)
 - confidence: itemset의 최소 confidence 값 (default = 0.8)
 - minlen, maxlen: itemset에 포함되는 최소 및 최대 item 수 (default = 1 & 10)

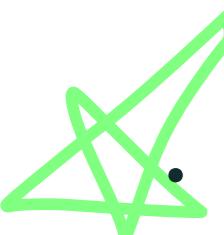
```
# default 옵션 사용
rules <- apriori(groceries)

rules
## set of 0 rules
```

default로 끄면 tm 발생

- default 옵션을 사용하여 apriori() 함수를 실행하는 경우, 생성되는 rule은 0개이다. support와 confidence 기준이 매우 높기 때문인 것으로 추측된다.
- 데이터의 도메인과 특성에 따라서 적절한 support와 confidence 옵션을 설정하여야 한다. 따라서 여러 support와 confidence 값에 대해 반복적으로 rule을 생성해보는 것이 필요하다.

Rule Generation



- 본 예제에서는 하루에 두 번 이상 구매되는 item을 중요하다고 가정해보자. 따라서 전체 30일동안 60번 이상 발생하는 itemset을 찾는 것이 합리적이므로, support 옵션을 0.006으로 설정하자 (전체 transaction은 9,835개이다.) $\frac{60}{9835} \approx 0.006$

- Confidence의 경우 너무 높은 값으로 설정하면, 상식적으로 매우 명확한 rule들만 생성될 가능성이 높다. 예를 들어 씨리얼은 대부분 우유와 함께 구매되는데, 이러한 rule들은 이미 알고 있기 때문에 새로운 정보로써의 가치가 없다. 본 예제에서는 confidence 옵션을 0.25로 설정해보자. ← 꽁꽁 날게 설정하면 의미를 알 수 있다.
- minlen = 2로 설정한다. item 하나로 구성되는 rule은 의미가 없다.

 ← 꽁꽁 날게 설정하면 의미를 알 수 있다!

```
rules <- apriori(groceries, parameter = list(support = 0.006, confidence = 0.25, minlen = 2))

rules
## set of 463 rules
```

- 위의 옵션으로 463개의 rule이 생성되었다.

Rule Generation

```
summary(rules)
```

```
## set of 463 rules
##
## rule length distribution (lhs + rhs):sizes
##   2   3   4
## 150 297 16
##
##      Min. 1st Qu. Median     Mean 3rd Qu.     Max.
##      2.000 2.000 3.000 2.711 3.000 4.000
##
## 질량
## summary of quality measures:
##       support      confidence      lift      count
##      Min. :0.006101  Min. :0.2500  Min. :0.9932  Min. : 60.0
## 1st Qu.:0.007117  1st Qu.:0.2971  1st Qu.:1.6229  1st Qu.: 70.0
## Median :0.008744  Median :0.3554  Median :1.9332  Median : 86.0
## Mean   :0.011539  Mean   :0.3786  Mean   :2.0351  Mean   :113.5
## 3rd Qu.:0.012303  3rd Qu.:0.4495  3rd Qu.:2.3565  3rd Qu.:121.0
## Max.   :0.074835  Max.   :0.6600  Max.   :3.9565  Max.   :736.0
##
## mining info:
##       data ntransactions support confidence
## groceries          9835      0.006         0.25
```

- Item 2개인 rule이 150개, 3개인 rule이 297개, 4개인 rule이 16개 생성되었다.
- 생성된 rule들에 대한 support, confidence, lift, count의 분포를 확인할 수 있다.

Rule Generation

- `inspect()` 함수를 사용하여 rule의 상세 정보를 확인할 수 있으며, `sort()` 함수를 사용하여 생성된 rule을 정렬시킬 수 있다.
누가 몇개 있나.
- `apriori()` 함수는 항상 rhs 파트가 item 1개인 rule만 생성한다.
구성 규칙 rule 놓았어

```
inspect(rules[1:3])
##      lhs                  rhs          support  confidence   lift    count
## [1] {potted plants} => {whole milk} 0.006914082 0.4000000 1.565460 68
## [2] {pasta}            => {whole milk} 0.006100661 0.4054054 1.586614 60
## [3] {herbs}             => {root vegetables} 0.007015760 0.4312500 3.956477 69

sorted_rules <- sort(rules, by = "lift")
inspect(sorted_rules[1:2])
##      lhs                  rhs          support  confidence   lift    count
## [1] {herbs}             => {root vegetables} 0.007015760 0.4312500 3.956477 69
## [2] {berries}           => {whipped/sour cream} 0.009049314 0.2721713 3.796886 89
```

- $\{berries\} \Rightarrow \{whipped\ cream\}$ rule은 89번 발생하며, 이것은 전체 transaction 중에서 0.9%에 해당한다.
- berries를 구매한 사람의 약 27%가 whipped cream을 구매하였다.
- berries를 구매한 사람은 그렇지 않은 사람에 비해 whipped cream을 구매할 가능성이 약 3.79배 더 높다고 할 수 있다.

Subsets of Rules

- $\{\text{berries}\} \Rightarrow \{\text{whipped cream}\}$ rule을 활용하여, 매장에서 berries와 whipped cream을 가까운 위치에 배치시킬 수 있으며, berries와 whipped cream을 동시에 구매할 경우 할인을 적용하는 프로모션을 진행할 수도 있을 것이다.
- 이번에는 `subset()` 함수를 사용하여 berries를 포함하는 rule을 모두 출력해보자.

```
berry_rules <- subset(rules, items %in% "berries")  
  
inspect(berry_rules)  
##      lhs            rhs          support    confidence     lift      count  
## [1] {berries} => {whipped/sour cream} 0.009049314 0.2721713 3.796886 89  
## [2] {berries} => {yogurt}                0.010574479 0.3180428 2.279848 104  
## [3] {berries} => {other vegetables}   0.010269446 0.3088685 1.596280 101  
## [4] {berries} => {whole milk}           0.011794611 0.3547401 1.388328 116
```

- $\{\text{berries}\} \Rightarrow \{\text{yogurt}\}$ rule의 경우, 아침식사나 디저트로 yogurt와 berries를 동시에 먹는 경우가 많기 때문으로 해석할 수 있다. 이 rule도 프로모션과 같은 마케팅에 활용될 수 있을 것이다.
- berries를 포함하는 나머지 rule들은 특별한 의미를 찾기 어렵다.

Subsets of Rules

- `subset()` 함수를 활용하여 특정 조건을 만족하는 rule들을 추출할 수 있다.

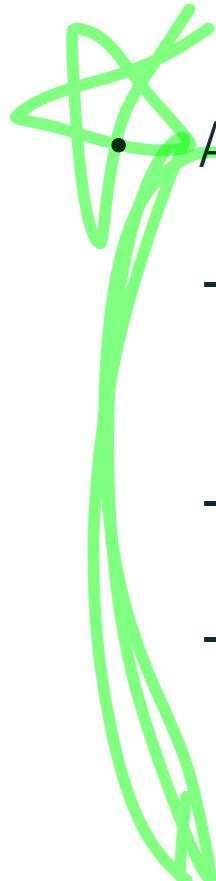
```
# 두 item 중 하나 이상을 포함하는 rule 추출
or_rules <- subset(rules, items %in% c("berries", "yogurt"))
or_rules
## set of 132 rules

# 두 item을 모두 포함하는 rule 추출
all_rules <- subset(rules, items %ain% c("berries", "yogurt"))
all_rules
## set of 1 rules

# confidence 값이 0.5 이상인 rule 추출
rules2 <- subset(rules, confidence > 0.5)
rules2
## set of 62 rules

# 발생 횟수가 500 이상인 rule 추출
rules3 <- subset(rules, count > 500)
inspect(rules3)
##      lhs                      rhs          support   confidence    lift     count
## [1] {yogurt}                => {whole milk} 0.05602440 0.4016035 1.571735 551
## [2] {rolls/buns}            => {whole milk} 0.05663447 0.3079049 1.205032 557
## [3] {other vegetables}     => {whole milk} 0.07483477 0.3867578 1.513634 736
## [4] {whole milk}             => {other vegetables} 0.07483477 0.2928770 1.513634 736
```

Rule Classification



Association rule 기법으로 생성한 rule은 다음과 같이 분류할 수 있다.

- **Actionable**: transaction data로부터 추출된 유용한 insight를 제공해주는 rule로써, market basket analysis에서 찾고자 하는 rule이다.
- **Trivial**: 분석 없이도 너무 명확한 rule이다. *ex) 쇠고기, 김치*
- **Inexplicable**: rule에 포함된 item들 간에 유의미한 관계를 찾기가 어려운 경우에 해당된다. 어떤 사람이 알 수 없는 이유로 특정한 item들을 매우 빈번하게 구매한 경우의 예를 들 수 있다.
- 알고리즘에 의해 생성된 rule 중에서 actionable rule을 찾기 위해서는 domain knowledge에 대한 이해가 필수적이다.