

# STA211 : Recherche de règles d'association

Vincent Audigier, Ndeye Niang

07 mars, 2025

- 1 Introduction
- 2 Quelques définitions et concepts de base
  - 2.1 Représentation et lecture des données
  - 2.2 Notions de support et de confiance
- 3 Recherche de règles d'association
  - 3.1 Algorithme Apriori
    - 3.1.1 Recherche des *itemsets* fréquents
    - 3.1.2 Extraction des règles
  - 3.2 Quelques algorithmes alternatifs
    - 3.2.1 Représentation horizontale des données
    - 3.2.2 Représentation verticale des données
    - 3.2.3 Bilan
  - 3.3 Indices de pertinence des règles
    - 3.3.1 Faiblesse de l'approche support-confiance
    - 3.3.2 Indices de pertinence
- 4 Exemples
  - 4.1 Supermarché
  - 4.2 German Credit
- 5 Conclusion
- Références

## 1 Introduction

Les méthodes de recherche de règles d'association ont été proposées pour découvrir quels produits étaient achetés conjointement et à quelle date dans les bases de données des ventes de supermarché (Agrawal, Imieliński, and Swami (1993)). Elles permettent d'extraire des règles de type : "lorsqu'un client achète du pain et du beurre alors 9 fois sur 10 il achète en même temps du lait". En effet, la technologie des codes-barres permet l'acquisition rapide et automatique des données relatives aux ventes. Trouver des ensembles d'articles achetés simultanément permet alors de mener des actions marketing pertinentes. Cependant, même si initialement la méthode de recherche de règles d'association a été développée dans un objectif marketing, elle peut être appliquée dans d'autres domaines si la structure des données s'y prête. Elle peut trouver des applications notamment en "webmining" dans l'analyse de la consultation des pages web sur un site Internet, ou encore en "text mining" dans la découverte de cooccurrences de termes dans les documents.

Cette section s'appuiera sur la thèse de Marie Plasse (Plasse (2006)) à laquelle le lecteur pourra se référer au besoin pour des approfondissements.

## 2 Quelques définitions et concepts de base

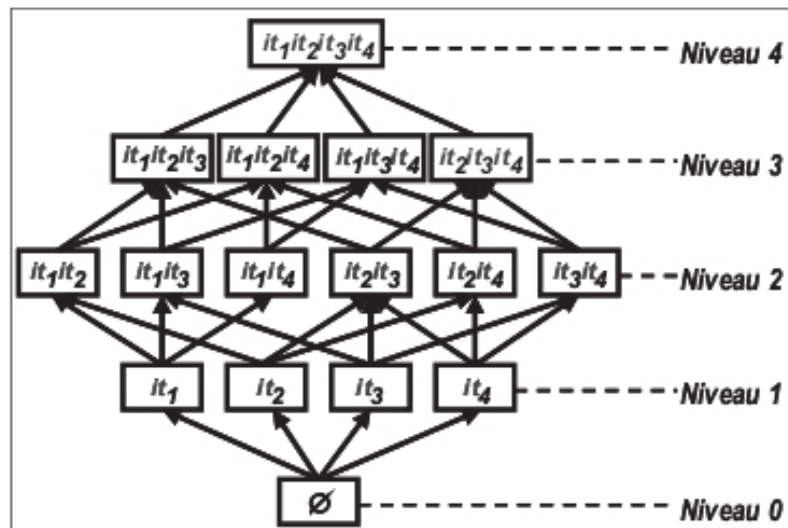
On définit l'ensemble  $I = \{it_1, it_2, \dots, it_i, \dots, it_p\}$  dont les éléments sont appelés **items**.  $I$  peut par exemple correspondre à l'ensemble des produits d'un supermarché. On appelle alors **transaction**, un sous-ensemble non-vide de  $I$ . Dans l'exemple précédent, une transaction correspond au contenu d'un caddie. On note

$T = \{tr_1, tr_2, \dots, tr_j, \dots, tr_n\}$  l'ensemble à  $n$  éléments des transactions.

Une règle d'association est une implication de la forme  $A \rightarrow B$  où  $A \subset I, B \subset I$  et  $A \cap B = \emptyset$ . Une règle comporte donc une partie **prémisse** (ou **antécédent**) composée d'un ensemble d'items  $A$  et une partie **conclusion** (ou **conséquent**) composée d'un ensemble d'items  $B$  disjoint de  $A$ . Les ensembles d'items sont appelés **itemsets**.

## 2.1 Représentation et lecture des données

Tous les *itemsets* des transactions peuvent être représentés par un treillis d'items formé de l'ensemble des parties de l'ensemble  $I$  muni de l'opération d'inclusion. La figure ci-dessous montre un exemple de treillis associé à un ensemble de 4 items.



Le treillis peut alors être lu en largeur ou en profondeur. Un parcours en largeur du treillis débute avec la lecture des *itemsets* de taille 1 puis continue avec la lecture des *itemsets* de taille 2 et ainsi de suite. Par contre, un parcours en profondeur s'intéresse d'abord à tous les *itemsets* commençant par le même sous-ensemble. Une fois arrivé à la profondeur maximale, la lecture revient à la racine et recommence avec un nouveau sous-ensemble (voir Figure animée 2.1).

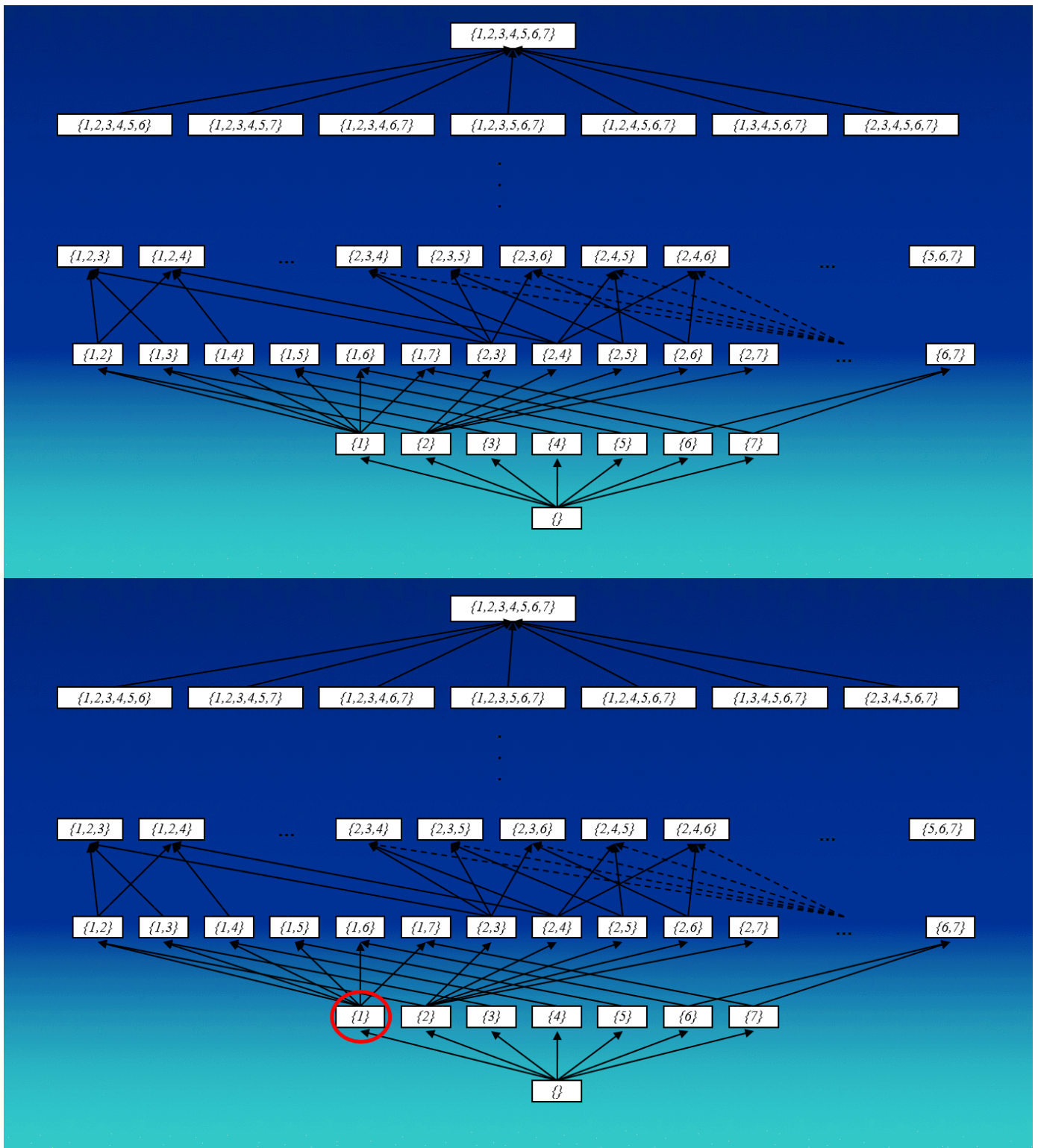


Figure 2.1: Mode de lecture du trellis : en largeur en haut, en profondeur en bas

Par ailleurs, les données peuvent être représentées sous la forme d'une matrice de données binaires de type présence absence où les transactions sont en ligne et les items en colonne. Une telle matrice peut être lue de plusieurs façons (cf Figure 2.2):

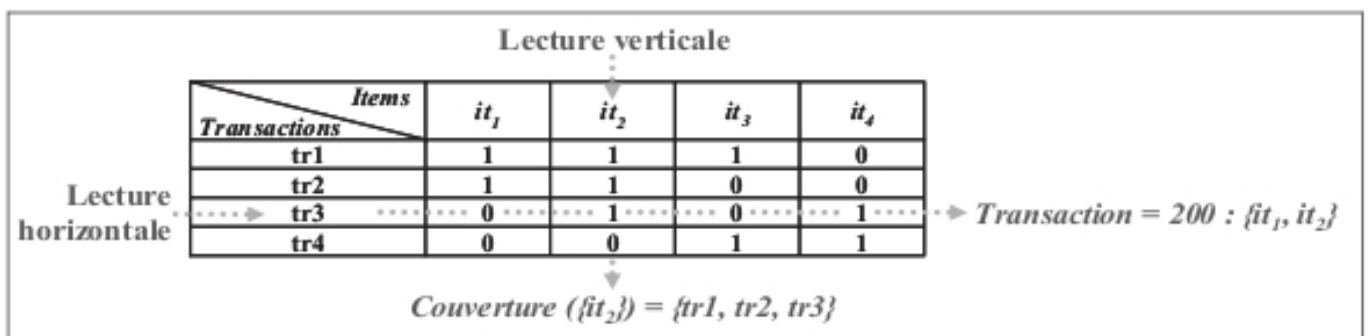


Figure 2.2: Représentation verticale et horizontale

- Lorsque la lecture est basée sur une représentation horizontale des données, on considère les transactions et à chacune d'elles correspond une liste d'items associés.
- Lorsque la lecture utilise une représentation verticale des données, on considère les items et la base de données est vue comme un ensemble d'items; à chacun d'eux correspond une liste de transactions associées. On appelle *couverture* d'un item l'ensemble des transactions qui le contiennent.

Les algorithmes de recherche de règles d'association diffèrent selon la manière dont ils parcourent le treillis et lisent (horizontalement ou verticalement) les données.

## 2.2 Notions de support et de confiance

Une règle d'association est attachée à deux informations indiquant sa pertinence : son support  $s$  et sa confiance  $c$ . Du point de vue ensembliste, le support correspond à la proportion de transactions contenant les items de  $A \cup B$  (le nombre de transactions contenant à la fois tous les items de  $A$  et tous les items de  $B$ ) par rapport à l'ensemble des transactions :

$$s = \text{supp}(A \longrightarrow B) = \frac{\text{card}(t \in T / A \cup B \subseteq t)}{\text{card}(T)}$$

Dans l'exemple des transactions d'un supermarché, dire que la règle {lait, pain} implique {beurre} possède un support de 0.5 signifie que 50% des transactions (caddies) contiennent à la fois du lait, du pain et du beurre.

La confiance indique la proportion de transactions contenant les items de  $A \cup B$  par rapport aux transactions contenant les items de  $A$

$$c = \text{conf}(A \longrightarrow B) = \frac{\text{card}(t \in T / A \cup B \subseteq t)}{\text{card}(t \in T / A \subseteq t)} = \frac{\text{supp}(A \longrightarrow B)}{\text{supp}(A)}$$

Par exemple, dire que la règle {lait, pain} implique {beurre} a une confiance de 1, signifie que 100% des caddies qui contenaient du lait et du pain contenaient aussi du beurre.

## 3 Recherche de règles d'association

La recherche de règles d'association se décompose en deux sous problèmes. Le premier est la recherche des *itemsets* fréquents dont le support est supérieur ou égal à un seuil minimum, noté *minsup*. A partir de ces sous ensembles fréquents, le second problème est l'extraction des règles d'association dont la confiance est supérieure ou égale à un second seuil minimum, noté *minconf*. Les deux seuils, *minsup* et *minconf*, sont spécifiés au préalable par l'utilisateur.

Dans cette section, nous présentons des algorithmes permettant de résoudre ces deux problèmes.

### 3.1 Algorithme Apriori

#### 3.1.1 Recherche des *itemsets* fréquents

L'algorithme fondateur pour la recherche de règles d'association *Apriori* (Agrawal, Srikant, et al. (1994)), repose sur la propriété selon laquelle *tout sous-ensemble d'un ensemble fréquent est nécessairement fréquent*.

Cette propriété permet en effet de réduire considérablement le nombre d'*itemsets* candidats générés, c'est à dire les *itemsets* potentiellement fréquents imposant un calcul du support. Les ensembles candidats sont générés *a priori* - d'où le nom de l'algorithme - et leur support est calculé à l'étape suivante.

L'algorithme Apriori utilise une représentation horizontale des données et un parcours en largeur du treillis des *itemsets*. Le principe est de rechercher de manière itérative les ensembles fréquents de cardinal  $k$  à partir des ensembles fréquents de cardinal  $k - 1$ , déterminés lors de l'itération précédente. Ainsi, à la lecture  $k$ , l'algorithme calcule le support des *itemsets* de taille  $k$ . Pour cela, le tableau des transactions est parcouru transaction par transaction et un compteur est tenu à jour pour chacun des *itemsets* de taille  $k$ . A la fin de ce parcours, seuls sont conservés les *itemsets* dont le support est supérieur ou égal à *minsup*. A partir de ces  $k$ -*itemsets* fréquents, l'algorithme génère les ensembles candidats de taille  $k + 1$  dont il calculera le support à la lecture  $k + 1$ . Lorsqu'il n'y a plus d'ensembles candidats, la procédure s'arrête (cf illustration ci-dessous)

A gorithme Apriori : etape  
1

Algorithmes

Déroulement pas à pas de Apriori

- Étape 1 : Recherche des sous-ensembles fréquents Support min = 40%

3 TRANSACTIONS

N°Transaction	Items
1	A B C

### 3.1.2 Extraction des règles

A partir des *itemsets* fréquents, l'étape suivante est l'extraction des règles d'association auxquelles est accordée une confiance au moins égale au seuil minimum *minconf* également fixé *a priori*. La méthode proposée par Agrawal, Srikant, et al. (1994) repose également sur la propriété selon laquelle tout sous-ensemble d'un ensemble fréquent est nécessairement fréquent.

Pour un *itemset* fréquent donné, l'algorithme génère toutes les règles ayant un conséquent de taille 1. Il calcule les confiances de ces règles et les compare à *minconf* pour sélectionner les règles à extraire. Ensuite, il génère les conséquents de taille 2 en utilisant seulement les conséquents de taille 1 des règles sélectionnées à l'étape précédente. Cette procédure continue jusqu'à ce qu'il n'y ait plus de conséquent à générer pour l'itemset. La procédure est illustrée ci-dessous.

# Algorithme Apriori : etape 2

# Algorithmes

## Déroulement pas à pas de Apriori

- Étape 1 : Recherche des sous-ensembles fréquents

Support min = 40%

3 TRANSACTIONS		TAILLE 1		TAILLE 2		TAILLE 3	
N°Transaction	Items	Itemset	Support	Itemset	Support	Itemset	Support
1	A B C	A	2	A B	2	A B D	1

En limitant le nombre d'ensembles candidats, l'algorithme Apriori économise des calculs inutiles pour le support, point déterminant dans le temps de calcul global. De plus, la base de données n'a pas besoin de tenir en mémoire car il est possible de lire le fichier transaction par transaction.

## 3.2 Quelques algorithmes alternatifs

L'algorithme Apriori a ensuite connu plusieurs extensions de ses fonctionnalités et performances. La recherche des *itemsets* fréquents est le point crucial de la recherche de règles d'association. De nombreux algorithmes ont, en effet, été proposés pour répondre à cette problématique. Quelques algorithmes de recherche d'*itemsets* fréquents proposés après Apriori sont présentés ci-après. Ces algorithmes visent principalement à limiter le nombre d'accès aux données et à accélérer les temps de calcul. Ils sont regroupés selon leur façon de représenter les données.

### 3.2.1 Représentation horizontale des données

#### 3.2.1.1 Algorithme Sampling

L'algorithme Sampling (Toivonen et al. (1996)) parcourt le treillis des *itemsets* en largeur (tout comme Apriori). Le principe de cet algorithme est de tirer un échantillon aléatoire de transactions permettant de chercher les *itemsets* fréquents. Cet échantillon étant de taille plus modeste que le nombre de transactions initial, il faut alors définir un seuil  $minsup' < minsup$  par rapport auquel sont déterminés les *itemsets* fréquents. La vérification des résultats et le calcul exact du support des *itemsets* fréquents (par rapport au seuil initial  $minsup$ ) sont ensuite réalisés avec le reste des transactions de la base.

#### 3.2.1.2 Algorithme DIC (Dynamic Itemset Counting)

Comme Apriori et Sampling, l'algorithme Dynamic Itemset Counting (DIC) parcourt le treillis des items en largeur. Il procède par niveaux dans le treillis des *itemsets* : dès qu'un *itemset* de niveau  $k$  s'avère fréquent, on examine tous les *itemsets* de taille  $k + 1$  générés par celui-ci (Brin et al. (1997)).

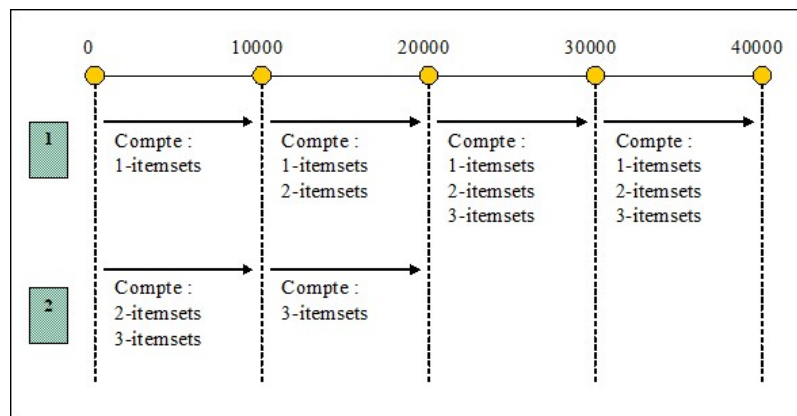


Figure 3.1: Schéma de fonctionnement de l'algorithme DIC

L'algorithme génère d'abord tous les *itemsets* de taille 1. Le support de ces *itemsets* candidats est calculé sur le premier intervalle de la base de données. A la fin du premier intervalle, les *itemsets* de taille 2 sont générés à partir des *itemsets* fréquents de taille 1. Le support de ces *itemsets* candidats de taille 2 commence alors à être calculé dès le second intervalle. Le procédé se répète jusqu'à l'obtention de tous les supports exacts de chaque *itemset* fréquent. Cet algorithme permet de réduire le nombre d'*itemsets* candidats et donc le nombre de calculs du support. Le nombre de lectures de la base est inférieur ou égal à celui d'Apriori, par contre il est nettement supérieur à celui de Sampling.

## 3.2.2 Représentation verticale des données

Cette représentation a pour avantage de permettre un calcul rapide du support d'un *itemset* donné en effectuant une simple opération d'intersection entre les couvertures des items concernés. Ainsi les algorithmes suivants sont plus rapides que Apriori.

### 3.2.2.1 Algorithme Partition

Comme Apriori, l'algorithme Partition (Savasere, Omiecinski, and Navathe (1995)) parcourt le treillis des items en largeur. Cet algorithme divise la base en parties disjointes de telle sorte que chacune, ainsi que les *itemsets* candidats qui seront générés, tiennent en mémoire. Ensuite, l'algorithme recherche les *itemsets* fréquents de chaque partie et détermine ainsi des *itemsets* fréquents locaux. Ces derniers sont alors réunis pour former l'ensemble des *itemsets* candidats globaux. Enfin, le support exact de tous les *itemsets* candidats globaux est calculé dans une seconde lecture de la base de données et les *itemsets* fréquents globaux sont ainsi obtenus.

Le partitionnement de la base soulève cependant plusieurs problèmes. Il est nécessaire de trouver un compromis entre la taille et le nombre des partitions. Elles doivent tenir en mémoire sans pour autant être trop nombreuses afin de garantir le principe de l'algorithme selon lequel "un itemset fréquent sur l'ensemble de la base doit être fréquent dans au moins une de ses partitions". Par ailleurs, dans l'idéal, chaque partition doit être représentative de la base entière pour éviter le risque de ne pas détecter un *itemset* fréquent.

### 3.2.2.2 Algorithme Eclat

L'algorithme Eclat "Equivalence CLASSE Transformation" (Zaki (2000)) ne requiert que de faibles besoins en mémoire. Il parcourt le treillis des items en profondeur et le décompose en sous parties qui sont résolues de manière indépendante en mémoire. Cette décomposition du treillis peut se faire en fonction du préfixe des ensembles candidats.

Eclat parcourt les  $k$ -*itemsets* ayant pour préfixe commun l'*itemset*  $a$  de taille  $k - 1$ . Il calcule la liste des transactions associées à chacun de ces *itemsets*. Il détermine ainsi les  $k$ -*itemsets* fréquents commençant par  $a$  et les place dans une nouvelle base de données  $D[a]$ . Ensuite, il s'agit d'une procédure récursive : l'algorithme recherche les *itemsets* fréquents commençant par  $b$  (avec  $a$  inclu dans  $b$ ).



Le fait de parcourir en profondeur d'abord le treillis des ensembles conduit à calculer le support d'*itemsets* supplémentaires par rapport à Apriori qui ne calcule que les supports nécessaires. Mais Eclat parvient quand même à être plus rapide qu'Apriori, grâce à la représentation verticale des données. Enfin, Eclat réalise une meilleure gestion de la mémoire que l'algorithme Partition qui utilise également une lecture verticale des données.

### 3.2.3 Bilan

En termes de résultats, les nombreux algorithmes pour la recherche des ensembles fréquents aboutissent tous aux mêmes ensembles fréquents puisque la recherche des ensembles fréquents est déterministe, tout comme la recherche de règles d'association. Cependant, tous ne sont pas équivalents en termes de performances sur les temps de calcul et la gestion de la mémoire. L'efficacité d'un algorithme dépend essentiellement de trois facteurs : la façon dont les *itemsets* candidats sont générés, les structures de données adoptées et l'implémentation de l'algorithme. Au vu de ces critères, Eclat est l'algorithme le plus rapide.

## 3.3 Indices de pertinence des règles

L'approche support-confiance a l'avantage d'être simple. En effet, le support et la confiance sont des concepts facilement compréhensibles et ils ont un sens concret. Cependant, cette approche montre plusieurs faiblesses. En effet, bien souvent, l'approche support-confiance précédemment décrite conduit à l'obtention de règles en trop grand nombre. Par conséquent, il est impossible de les faire valider par un expert. Dès lors, il est utile de les trier par ordre décroissant de leur intérêt au sens d'un indice de pertinence, tel que le lift (Brin et al., 1997), pour citer un des plus connus.

### 3.3.1 Faiblesse de l'approche support-confiance

Le choix de la valeur de *minsup* est difficile et est laissé à l'utilisateur. Un seuil élevé pour le support risque de disqualifier certaines règles très intéressantes, ayant un support faible mais une confiance très élevée. Par exemple, l'item "caviar" étant rare, il ne sera pas retenu alors qu'il peut cacher une règle du type "caviar → vodka" valable dans 85% des cas. D'un point de vue marketing, il est dommage de passer à côté d'une telle règle car le caviar est un produit de luxe. Sachant que l'achat de l'un entraîne de manière quasi sûre l'achat de l'autre, il pourrait être pertinent, par exemple, de faire une promotion sur la vodka lors de l'achat de caviar.

A l'inverse, fixer un seuil très bas pour le support peut entraîner l'extraction de règles sans intérêt du type "caviar → lait". En effet, le lait est un produit très commun qui figure dans un nombre élevé de transactions. Il n'est donc pas surprenant de le trouver dans le caddie de tous les clients qui achètent du caviar. Dans d'une application sur données avec des événements statistiquement rares, le support minimum doit être obligatoirement très faible. Par conséquent, les règles du type caviar → lait sont très nombreuses et doivent être sanctionnées.

Dans la règle "caviar → lait", le support mesure la fréquence globale des exemples de la règle ; il est donc faible étant donné la faible fréquence d'achat du produit caviar. La confiance est proche de 100% puisqu'elle s'intéresse à la répartition des achats de caviar entre les achats de lait et les "non achats" de lait. La répartition des "non achats" de caviar n'est pas du tout prise en compte alors qu'elle permettrait de discriminer ce type de règles. Ainsi l'approche support-confiance ne suffit pas pour discriminer les deux règles.



### 3.3.2 Indices de pertinence

La seule utilisation des seuils *minsup* et *minconf* présente donc quelques limites qui peuvent être levées grâce à l'utilisation d'autres critères pour mesurer l'intérêt statistique des règles. Ainsi, de nombreux indices de pertinence ont été présentés dans la littérature (voir par exemple ici (<https://mhahsler.github.io/arules/docs/measures>) pour différents exemples d'indices). Parmi eux, le lift (Brin et al., 1997) est un des plus connus et utilisés. Il est très facilement interprétable. C'est le rapport de la probabilité de trouver ensemble les items de l'antécédent et du conséquent sur la probabilité de les trouver ensemble alors qu'ils sont indépendants. Ainsi, les règles ayant un lift plus petit ou autour de 1 ne sont pas jugées intéressantes. Par contre, un lift égal à 2 montre que le nombre d'exemples de la règle  $A \rightarrow C$  est deux fois plus grand que celui attendu sous l'indépendance.

Il est donc possible de classer les règles par ordre décroissant d'un indice de pertinence, tel que le lift, et de faire analyser par l'expert seulement les plus intéressantes au sens de cet indice. Cependant, l'indice de pertinence optimal n'existe pas et le nombre élevé de tels indices est une difficulté supplémentaire pour l'utilisateur qui doit choisir le plus approprié à ses besoins. Parmi les travaux évaluant les indices de pertinence, nous pouvons citer ceux de Lallich et Teytaud (2004) qui proposent des critères d'évaluation. Aussi, Tan et al. (2002) ont réalisé une étude comparative d'une vingtaine d'indices symétriques, alors que Vaillant et al. (2004) s'intéressent aux indices dissymétriques.

Lenca et al. (2004) remarquent que certains indices sont équivalents, c'est à dire qu'ils classent les règles dans le même ordre. Par exemple, l'indice de Sebag-Schoenauer est une transformation monotone croissante de la confiance.

Afin d'évaluer les indices de pertinence, Lenca et al. (2004) définissent huit critères formels tels que :

1. le traitement non symétrique de A et de C : l'indice ne doit pas avoir la même valeur pour les règles  $A \rightarrow C$  et  $C \rightarrow A$
2. la décroissance avec le nombre d'occurrences de C dans la base de données (ce critère pénalise les règles du type caviar  $\rightarrow$  lait).

Cependant, aucun indice de pertinence ne satisfait tous ces critères. L'utilisateur est le seul juge de l'importance accordée au respect de chaque critère.

Sur la base de ces critères, les auteurs ont effectué une étude comparative des différents indices et aboutissent à une typologie en trois classes (voir Figure 3.2)

Mesures	Définition
<b>CLASSE 1</b>	
Piatetsky-Shapiro	$nP(A)(P(C A)-P(C)) = nP(A)P(C)(\text{lift} - 1)$ (avec $n$ le nombre de transactions)
Indice de qualité de Cohen	$2 \frac{P(AC) - P(A)P(C)}{P(A) + P(C) - 2P(A)P(C)}$
Gain Informationnel	$\log \frac{P(AC)}{P(A)P(C)} = \log(\text{lift})$
Confiance centrée	$P(C A) - P(C)$
Lift	$\frac{P(AC)}{P(A)P(C)}$
Coefficient de corrélation de Pearson	$\frac{P(AC) - P(A)P(C)}{\sqrt{P(A)P(C)P(\bar{A})P(\bar{C})}}$
Indice d'implication	$\sqrt{n} \frac{P(A\bar{C}) - P(A)P(\bar{C})}{\sqrt{P(A)P(\bar{C})}}$
Indice probabiliste discriminant	$P[N(0,1) > \text{indice d'implication}^{CR/B}]$ (La notation $CR/B$ signifie que l'indice d'implication est préalablement centré et réduit sur une base d'exemples.)
<b>CLASSE 2</b>	
Support	$P(AC)$
Confiance	$P(C A)$
Surprise	$\frac{P(AC) - P(A\bar{C})}{P(C)} = 2 \frac{P(A)}{P(C)} (\text{confiance} - 0.5)$
Laplace	$\frac{P(C A) + n / P(A)}{1 + (2n) / P(A)}$
Sebag et Schoenauer	$\frac{P(AC)}{P(A\bar{C})} = \frac{\text{confiance}}{1 - \text{confiance}}$
Taux d'exemples et de contre exemples	$1 - \frac{P(A\bar{C})}{P(AC)} = 1 - \frac{1}{\text{Sebag et Schoenauer}}$
<b>CLASSE 3</b>	
Multiplicateur de cotes	$\frac{P(AC)P(\bar{C})}{P(A\bar{C})P(C)} = \text{lift} \cdot \text{conviction}$
Loevinger	$\frac{P(C A) - P(C)}{P(\bar{C})} = \frac{1}{P(\bar{C})} \text{confiance centrée} = 1 - \frac{1}{\text{conviction}}$
Zhang	$\frac{P(AC) - P(A)P(C)}{\max \{P(AC)P(\bar{C}); P(C)(P(A) - P(AC))\}}$
Conviction	$\frac{P(A)P(\bar{C})}{P(A\bar{C})}$


Figure 3.2: Typologie des indices de pertinences.

où chacun de ces indices est une fonction des huit éléments suivants

	$C$	$\bar{C}$	Profils lignes
$A$	$P(AC)$	$P(A\bar{C})$	$P(A)$
$\bar{A}$	$P(\bar{A}C)$	$P(\bar{A}\bar{C})$	$P(\bar{A})$
Profils colonnes	$P(C)$	$P(\bar{C})$	1

A titre d'exemple, considérons 100 consommateurs : 8 ont acheté du caviar, 40 ont acheté du lait et 7 ont acheté les deux en même temps. Cette règle qui n'a, en réalité, aucun intérêt va tout de même présenter des indices de pertinence élevés, comme le montre la figure 3.3

	<i>C</i>	$\bar{C}$	Profils lignes
<i>A</i>	0,07	0,01	0,08
$\bar{A}$	0,33	0,59	0,92
Profils colonnes	0,4	0,6	1



<i>Indices</i>	<i>Valeur</i>
Confiance	0,88
Confiance centrée	0,48
Lift	2,19
Multiplicateur de cotes	10,5
Loevinger	0,79

Figure 3.3: Exemple de règle sans intérêt aux indices de pertinence élevés.

Selon le lift, le nombre d'exemples de {caviar → lait} est environ deux fois plus grand que sous l'indépendance de {caviar} et {lait}

Il faudrait des indices de pertinence qui pénalisent mieux les règles où le conséquent est fréquent par rapport à l'antécédent. C'est le cas de l'indice accords-désaccords proposé par Kulczynski (en 1927)

$$IAD = \frac{P(AC)}{P(\bar{A}C) + P(A\bar{C})} = \frac{\text{accords positifs}}{\text{désaccords}}$$

ou l'indice de Jaccard

$$Jaccard = \frac{P(AC)}{P(A) + P(C) - P(AC)} = \frac{P(AC)}{P(A \cup C)}$$

L'union représente un "ou" inclusif alors que la différence symétrique exprime un "ou" exclusif.

Les deux indices sont parfaitement équivalents dans la mesure où

$$\frac{1}{Jaccard} = \frac{1}{IAD} + 1$$

Ainsi, ils conduisent à un classement identique des règles d'association. L'indice de Jaccard présente l'intérêt d'être borné entre 0 et 1.

Sur l'exemple de la figure 3.3, ils se montrent plus sévères que les autres indices retenus tels que le lift, en effet : IAD=0.21 et Jaccard =0.17. Ce résultat n'est toutefois pas généralisable à toutes les applications.

On retrouvera dans Plasse (2006) une analyse graphique permettant de comparer ces différents indices.

## 4 Exemples

Le package R *arules* permet de rechercher des règles d'association selon les algorithmes Apriori ou Eclat. Nous proposons ici de l'utiliser afin d'illustrer la méthode de recherche d'associations sur un premier jeu de données ``jouet'' portant sur des achats de produits de consommation où les règles sont faciles à interpréter. Par la suite, nous rechercherons les règles d'association dans le jeu de données German Credit.

### 4.1 Supermarché

On commence par saisir les données et à les convertir dans un format adapté à l'application des algorithmes de recherche de règles.

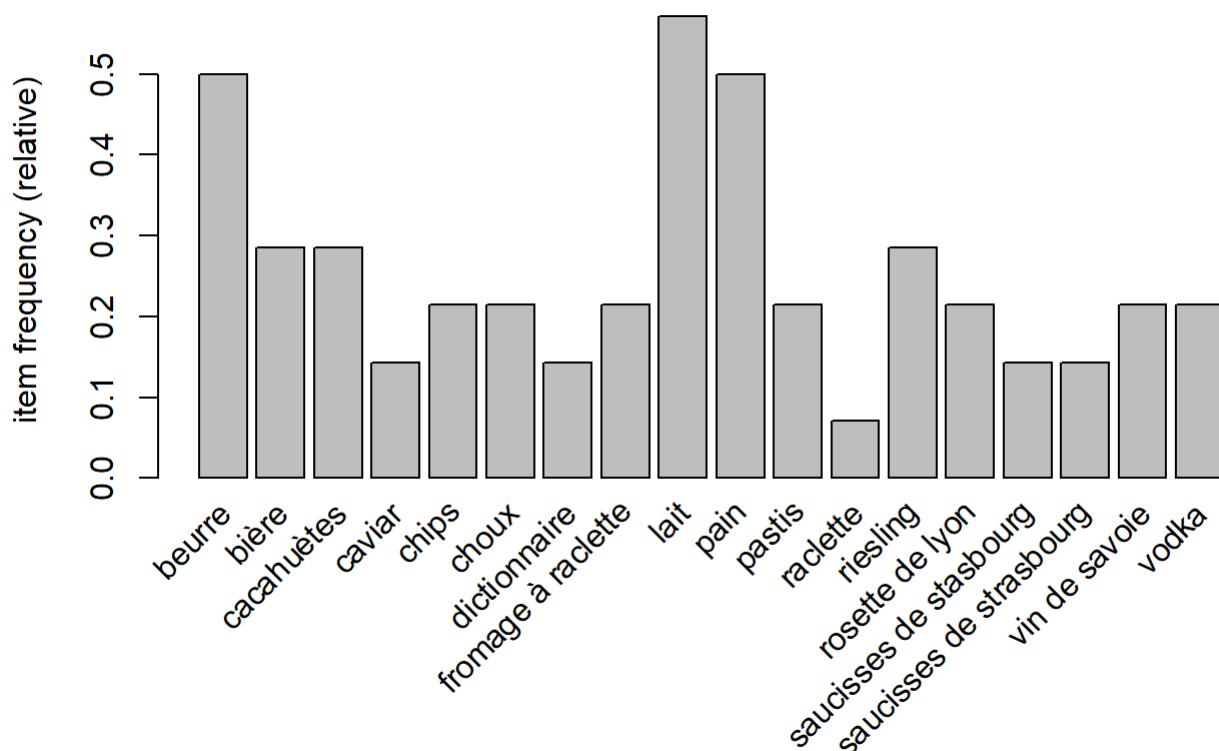
```
# charger Le package arules
library(arules)
```

```
# saisie des données
supermarche<-list(c("dictionnaire", "lait", "pain", "beurre"),
                  c("caviar", "vodka", "fromage à raclette", "rosette de lyon", "vin de savoie"),
                  c("saucisses de stasbourg","choux","riesling","pastis","bière","cacahuètes","chips"),
                  c("lait","pain","beurre"),
                  c("pastis","bière","cacahuètes","chips"),
                  c("lait","pain","beurre","pastis","bière","cacahuètes","chips"),
                  c("lait","pain","beurre","riesling","raclette"),
                  c("lait","pain","beurre","cacahuètes"),
                  c("saucisses de stasbourg","choux","riesling","fromage à raclette","rosette de lyon","vin de savoie"),
                  c("lait","pain","beurre","caviar","vodka"),
                  c("bière","vodka","saucisses de strasbourg"),
                  c("lait","pain","beurre"),
                  c("fromage à raclette","vin de savoie","rosette de lyon"),
                  c("dictionnaire","lait","saucisses de strasbourg","choux","riesling"))

# conversion au format transactions
supermarche.trans <- as(supermarche,"transactions")
```

Le jeu de données comporte 14 transactions et 18 items dont la fréquence est répartie de la façon suivante :

```
# Fréquences des items
itemFrequencyPlot(supermarche.trans)
```



Le beurre, le lait et le pain sont les items les plus fréquents dans les différentes transactions. On recherche les règles telles que le support soit supérieur à 3/18 selon l'algorithme Apriori.

```
# extraction des règles avec support supérieur à 3/18
supermarche.regles <- apriori(supermarche.trans,
                              parameter = list(supp = 3/18, conf = 0, minlen = 2, target = "rules"))

inspect(supermarche.regles)
```

##	lhs	rhs	support	confidence	coverage	lift
count						
## [1]	{fromage à raclette}	=> {rosette de lyon}	0.2142857	1.000	0.2142857	4.666667
3						
## [2]	{rosette de lyon}	=> {fromage à raclette}	0.2142857	1.000	0.2142857	4.666667
3						
## [3]	{fromage à raclette}	=> {vin de savoie}	0.2142857	1.000	0.2142857	4.666667
3						
## [4]	{vin de savoie}	=> {fromage à raclette}	0.2142857	1.000	0.2142857	4.666667
3						
## [5]	{rosette de lyon}	=> {vin de savoie}	0.2142857	1.000	0.2142857	4.666667
3						
## [6]	{vin de savoie}	=> {rosette de lyon}	0.2142857	1.000	0.2142857	4.666667
3						
## [7]	{pastis}	=> {chips}	0.2142857	1.000	0.2142857	4.666667
3						
## [8]	{chips}	=> {pastis}	0.2142857	1.000	0.2142857	4.666667
3						
## [9]	{pastis}	=> {bière}	0.2142857	1.000	0.2142857	3.500000
3						
## [10]	{bière}	=> {pastis}	0.2142857	0.750	0.2857143	3.500000
3						
## [11]	{pastis}	=> {cacahuètes}	0.2142857	1.000	0.2142857	3.500000
3						
## [12]	{cacahuètes}	=> {pastis}	0.2142857	0.750	0.2857143	3.500000
3						
## [13]	{choux}	=> {riesling}	0.2142857	1.000	0.2142857	3.500000
3						
## [14]	{riesling}	=> {choux}	0.2142857	0.750	0.2857143	3.500000
3						
## [15]	{chips}	=> {bière}	0.2142857	1.000	0.2142857	3.500000
3						
## [16]	{bière}	=> {chips}	0.2142857	0.750	0.2857143	3.500000
3						
## [17]	{chips}	=> {cacahuètes}	0.2142857	1.000	0.2142857	3.500000
3						
## [18]	{cacahuètes}	=> {chips}	0.2142857	0.750	0.2857143	3.500000
3						
## [19]	{bière}	=> {cacahuètes}	0.2142857	0.750	0.2857143	2.625000
3						
## [20]	{cacahuètes}	=> {bière}	0.2142857	0.750	0.2857143	2.625000
3						
## [21]	{beurre}	=> {pain}	0.5000000	1.000	0.5000000	2.000000
7						
## [22]	{pain}	=> {beurre}	0.5000000	1.000	0.5000000	2.000000
7						
## [23]	{beurre}	=> {lait}	0.5000000	1.000	0.5000000	1.750000
7						
## [24]	{lait}	=> {beurre}	0.5000000	0.875	0.5714286	1.750000
7						
## [25]	{pain}	=> {lait}	0.5000000	1.000	0.5000000	1.750000
7						
## [26]	{lait}	=> {pain}	0.5000000	0.875	0.5714286	1.750000
7						
## [27]	{fromage à raclette,					

##	rosette de lyon}	=> {vin de savoie}	0.2142857	1.000 0.2142857 4.666667
3				
## [28]	{fromage à raclette,			
##	vin de savoie}	=> {rosette de lyon}	0.2142857	1.000 0.2142857 4.666667
3				
## [29]	{rosette de lyon,			
##	vin de savoie}	=> {fromage à raclette}	0.2142857	1.000 0.2142857 4.666667
3				
## [30]	{chips,			
##	pastis}	=> {bière}	0.2142857	1.000 0.2142857 3.500000
3				
## [31]	{bière,			
##	pastis}	=> {chips}	0.2142857	1.000 0.2142857 4.666667
3				
## [32]	{bière,			
##	chips}	=> {pastis}	0.2142857	1.000 0.2142857 4.666667
3				
## [33]	{chips,			
##	pastis}	=> {cacahuètes}	0.2142857	1.000 0.2142857 3.500000
3				
## [34]	{cacahuètes,			
##	pastis}	=> {chips}	0.2142857	1.000 0.2142857 4.666667
3				
## [35]	{cacahuètes,			
##	chips}	=> {pastis}	0.2142857	1.000 0.2142857 4.666667
3				
## [36]	{bière,			
##	pastis}	=> {cacahuètes}	0.2142857	1.000 0.2142857 3.500000
3				
## [37]	{cacahuètes,			
##	pastis}	=> {bière}	0.2142857	1.000 0.2142857 3.500000
3				
## [38]	{bière,			
##	cacahuètes}	=> {pastis}	0.2142857	1.000 0.2142857 4.666667
3				
## [39]	{bière,			
##	chips}	=> {cacahuètes}	0.2142857	1.000 0.2142857 3.500000
3				
## [40]	{cacahuètes,			
##	chips}	=> {bière}	0.2142857	1.000 0.2142857 3.500000
3				
## [41]	{bière,			
##	cacahuètes}	=> {chips}	0.2142857	1.000 0.2142857 4.666667
3				
## [42]	{beurre,			
##	pain}	=> {lait}	0.5000000	1.000 0.5000000 1.750000
7				
## [43]	{beurre,			
##	lait}	=> {pain}	0.5000000	1.000 0.5000000 2.000000
7				
## [44]	{lait,			
##	pain}	=> {beurre}	0.5000000	1.000 0.5000000 2.000000
7				
## [45]	{bière,			
##	chips,			
##	pastis}	=> {cacahuètes}	0.2142857	1.000 0.2142857 3.500000



```

3
## [46] {cacahuètes,
##      chips,
##      pastis}      => {bière}      0.2142857      1.000 0.2142857 3.500000
3
## [47] {bière,
##      cacahuètes,
##      pastis}      => {chips}      0.2142857      1.000 0.2142857 4.666667
3
## [48] {bière,
##      cacahuètes,
##      chips}      => {pastis}      0.2142857      1.000 0.2142857 4.666667
3

```

Le nombre de règles obtenues est de 48. Par exemple, la règle 45 est {bière,chips,pastis} => {cacahuètes}. Son support vaut 21% ce qui signifie que parmi l'ensemble des transactions, 21% contiennent les items bière,chips,pastis,cacahuètes. On notera que la fonction renvoie aussi l'indice *coverage* correspondant au support de l'antécédent de la règle, indiquée en ligne. Ici, il vaut 21% ce qui signifie que parmi l'ensemble des transactions, 21% contiennent les items bière, chips, pastis. Aussi, sa confiance vaut 1 ce qui signifie que tous les gens qui achètent bière, chips et pastis achètent aussi des cacahuètes. On peut remarquer que le caviar ou le dictionnaire n'apparaissent pas dans ces différentes règles. En effet, ces produits sont plutôt rares, il faudrait abaisser le support minimum pour pouvoir les observer. Au contraire, le lait est souvent présent, mais comme ce produit est très fréquent, les règles qui le contiennent ne sont pas forcément très intéressantes.

Parmi ces 48 règles, on propose de s'intéresser à celles donc le lift est le plus élevé.

```

# afficher les 10 règles avec le lift le + élevé
regles.triees <- sort(supermarche.regles, by = "lift")
inspect(regles.triees[1:10])

```

##	lhs	rhs	support	confidence	coverage	lift
count						
## [1]	{fromage à raclette}	=> {rosette de lyon}	0.2142857	1	0.2142857	4.666667
3						
## [2]	{rosette de lyon}	=> {fromage à raclette}	0.2142857	1	0.2142857	4.666667
3						
## [3]	{fromage à raclette}	=> {vin de savoie}	0.2142857	1	0.2142857	4.666667
3						
## [4]	{vin de savoie}	=> {fromage à raclette}	0.2142857	1	0.2142857	4.666667
3						
## [5]	{rosette de lyon}	=> {vin de savoie}	0.2142857	1	0.2142857	4.666667
3						
## [6]	{vin de savoie}	=> {rosette de lyon}	0.2142857	1	0.2142857	4.666667
3						
## [7]	{pastis}	=> {chips}	0.2142857	1	0.2142857	4.666667
3						
## [8]	{chips}	=> {pastis}	0.2142857	1	0.2142857	4.666667
3						
## [9]	{fromage à raclette, rosette de lyon}	=> {vin de savoie}	0.2142857	1	0.2142857	4.666667
3						
## [10]	{fromage à raclette, vin de savoie}	=> {rosette de lyon}	0.2142857	1	0.2142857	4.666667
3						

Selon ce critère, les règles {fromage à raclette} => {rosette de lyon}, ou {rosette de lyon} => {vin de savoie}, ou {fromage à raclette, rosette de lyon} => {vin de savoie} font partie des plus intéressantes, à titre d'exemple. A la lecture du tableau, on voit que tous les individus achetant du fromage à raclette achètent systématiquement de la rosette de Lyon (confiance à 1), que ceux qui achètent de la rosette achètent systématiquement du vin de Savoie (confiance à 1), et que donc, ceux qui achètent fromage à raclette et rosette de Lyon achètent systématiquement du vin de Savoie.

Pour approfondir cet exemple, on pourra faire l'exercice disponible sur la plateforme moodle dont la correction est disponible en ligne.

## 4.2 German Credit

Nous appliquons à présent la recherche de règles sur le jeu German Credit dans lequel les variables quantitatives ont été recodées en facteur via une discrétisation). L'illustration ci-dessous est issue d'une étude de cas de R. Rakotomalala ([http://eric.univ-lyon2.fr/ricco/tanagra/fichiers/fr\\_Tanagra\\_Assoc\\_Rules\\_Comparison.pdf](http://eric.univ-lyon2.fr/ricco/tanagra/fichiers/fr_Tanagra_Assoc_Rules_Comparison.pdf))

```
# charger le fichier de données
german.quali <- read.table(file = "credit-german.txt", header = T, dec = ".", sep = "\t", stringsAsFactors = TRUE)
```

On recherche les règles d'association selon l'algorithme Apriori, pour un support de 0.25 et une confiance de 0.75. On spécifie également que seuls les *itemsets* de taille comprise entre 2 et 10 sont inspectés.

```
# transformer les données attributs-variables en données transactionnelles
german.trans <- as(german.quali, "transactions")

# extraction des règles
german.regles <- apriori(german.trans, parameter = list(supp = 0.25, conf = 0.75, minlen = 2,
maxlen = 10, target = "rules"))
```

On obtient alors 1928 règles dont voici un extrait

```
# afficher les 10 premières règles parmi les 1928
inspect(german.regles[1:10])
```

##	lhs	rhs	support	confi
	dence coverage lift count			
## [1]	{checking_status=0<=X<200}	=> {foreign_worker=yes}	0.264	0.98
14126	0.269 1.0191201 264			
## [2]	{checking_status=<0}	=> {foreign_worker=yes}	0.259	0.94
52555	0.274 0.9815737 259			
## [3]	{purpose=radio/tv}	=> {num_dependents=one}	0.250	0.89
28571	0.280 1.0566357 250			
## [4]	{purpose=radio/tv}	=> {foreign_worker=yes}	0.275	0.98
21429	0.280 1.0198784 275			
## [5]	{property_magnitude=real estate}	=> {foreign_worker=yes}	0.262	0.92
90780	0.282 0.9647747 262			
## [6]	{credit_history=critical/other existing } => {other_payment_plans=none}		0.251	0.85
66553	0.293 1.0524021 251			
## [7]	{credit_history=critical/other existing } => {other_parties=none}		0.268	0.91
46758	0.293 1.0084628 268			
## [8]	{credit_history=critical/other existing } => {foreign_worker=yes}		0.279	0.95
22184	0.293 0.9888042 279			
## [9]	{class=bad}	=> {num_dependents=one}	0.254	0.84
66667	0.300 1.0019724 254			
## [10]	{class=bad}	=> {other_parties=none}	0.272	0.90
66667	0.300 0.9996325 272			

On peut ensuite trier les règles selon le lift et s'intéresser aux règles les plus pertinentes au sens de ce critère

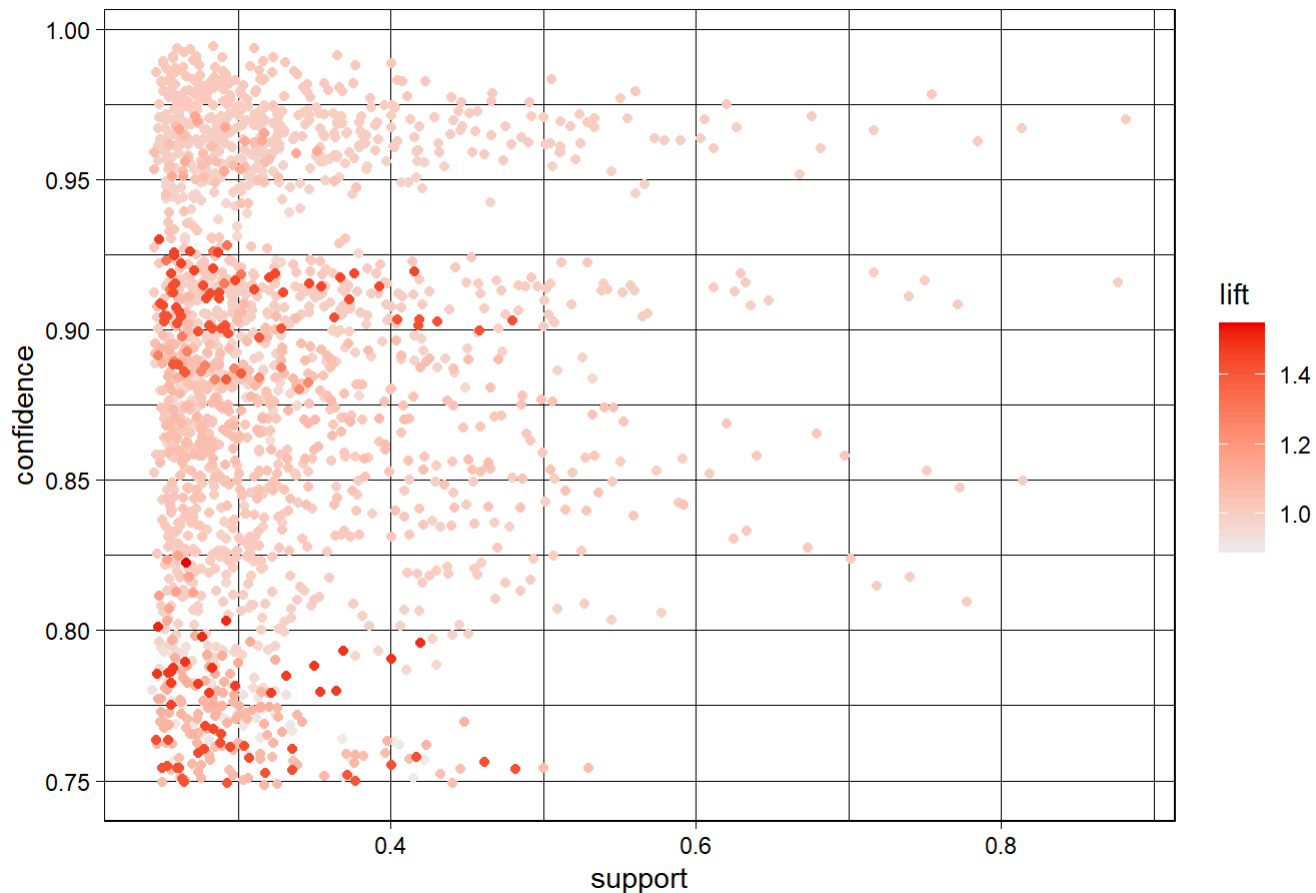
```
# afficher les 5 règles avec le lift le + élevé
regles.triees <- sort(german.regles, by = "lift")
inspect(regles.triees[1:5])
```

##	lhs	rhs	support	confidence	cover
age	lift count				
## [1]	{other_payment_plans=none, existing_credits=one, own_telephone=none}	=> {credit_history=existing paid}	0.263	0.8218750	0.
320	1.550708 263				
## [2]	{other_payment_plans=none, housing=own, existing_credits=one, num_dependents=one}	=> {credit_history=existing paid}	0.253	0.8031746	0.
315	1.515424 253				
## [3]	{other_payment_plans=none, housing=own, existing_credits=one}	=> {credit_history=existing paid}	0.287	0.8016760	0.
358	1.512596 287				
## [4]	{other_payment_plans=none, housing=own, existing_credits=one, foreign_worker=yes}	=> {credit_history=existing paid}	0.271	0.7970588	0.
340	1.503885 271				
## [5]	{other_payment_plans=none, existing_credits=one}	=> {credit_history=existing paid}	0.415	0.7950192	0.
522	1.500036 415				

Une connaissance précise des données sera nécessaire pour exploiter ces règles. Par la suite, nous présentons comment le package *arulesViz* peut être utilisé pour fournir des sorties graphiques facilitant l'exploitation des règles. La fonction `plot` permet de visualiser chacune des règles en fonction de son support, de sa confiance et du lift associé.

```
# chargement du package
library("arulesViz")
plot(german.regles)
```

Scatter plot for 1928 rules



L'argument `engine = 'interactive'` permettra d'inspecter à la main certaines règles dignes d'intérêt.

```
sel <- plot(german.regles, engine = 'interactive')
```

On retrouve dans Ong et al. (2002), une autre visualisation des règles à partir d'une représentation matricielle. Le principe est de créer une matrice dont les colonnes et les lignes correspondent respectivement aux antécédents et conséquents. La matrice contient les valeurs d'un indice de pertinence sélectionné par l'analyste (par défaut, le *lift*) et est visualisée à l'aide du coloriage de la matrice. Pour améliorer la lisibilité des résultats, *arulesViz* réorganise la matrice de manière à ce que les valeurs de l'indice choisi diminuent de la gauche vers la droite et du haut vers le bas. Ainsi, les règles présentant les indices les plus élevés sont placées en haut à gauche du graphique.

```
plot(german.regles, method = "matrix")
```

## Matrix for 1928 rules

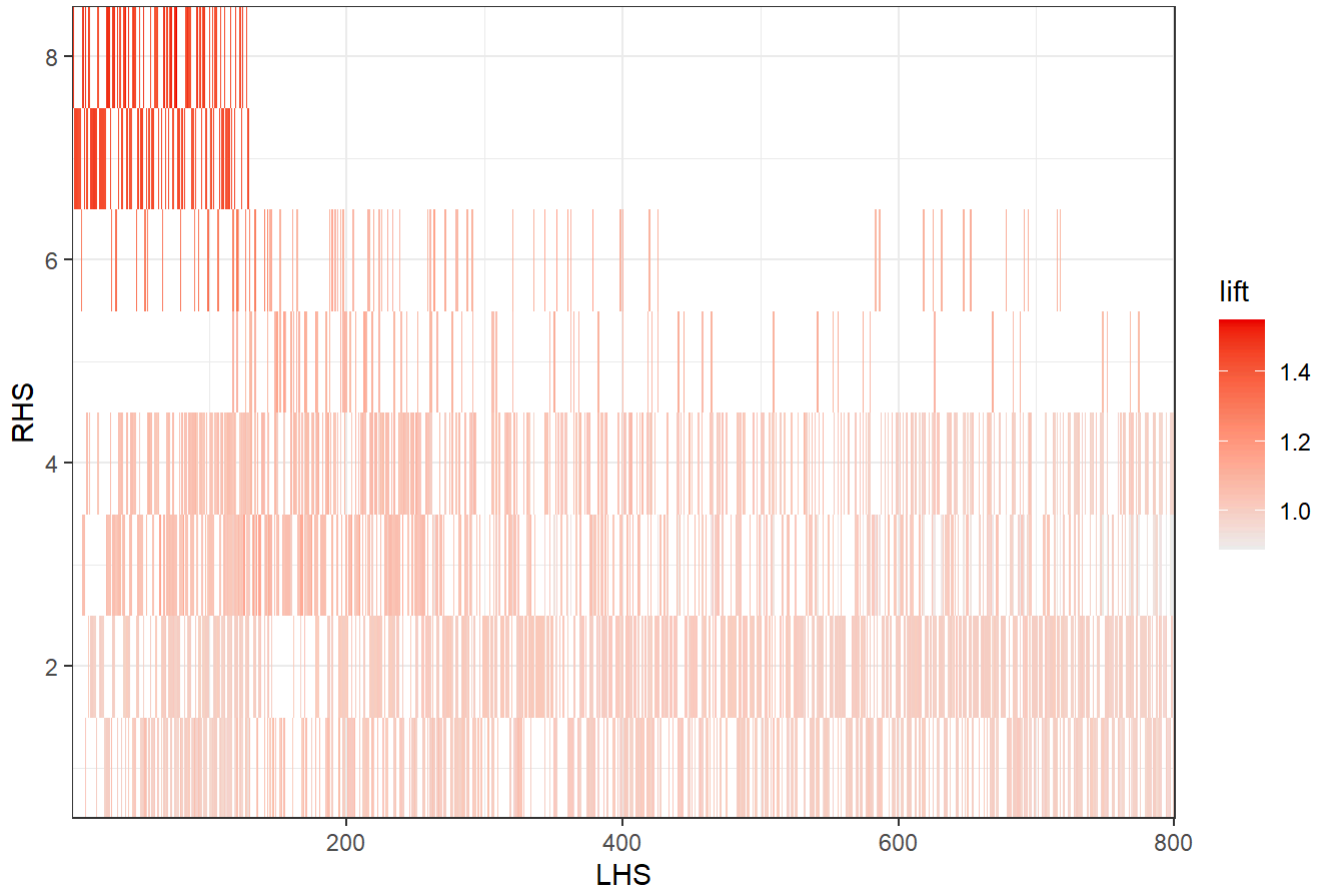


Figure 4.1: Représentation matricielle des règles. En abscisses, les antécédents (LHS), en colonne les conséquents (RHS). Le coloriage correspond à la valeur du lift.

### Itemsets in Antecedent (LHS)

- [1] "{other\_parties=none,other\_payment\_plans=none,existing\_credits=one,num\_dependents=one,foreign\_worker=yes}"
- [2] "{credit\_history=existing paid,other\_parties=none,num\_dependents=one,foreign\_worker=yes,class=good}"
- [3] "{credit\_history=existing paid,other\_parties=none,other\_payment\_plans=none,num\_dependents=one,foreign\_worker=yes}"
- [4] "{credit\_history=existing paid,other\_parties=none,own\_telephone=none,foreign\_worker=yes}"
- [5] "{credit\_history=existing paid,num\_dependents=one,own\_telephone=none,foreign\_worker=yes}"
- [6] "{credit\_history=existing paid,other\_parties=none,housing=own,num\_dependents=one,foreign\_worker=yes}"
- [7] "{checking\_status=no checking,other\_parties=none,housing=own,foreign\_worker=yes}"
- [8] "{other\_parties=none,other\_payment\_plans=none,existing\_credits=one,foreign\_worker=yes}"
- [9] "{credit\_history=existing paid,other\_parties=none,other\_payment\_plans=none,foreign\_worker=yes}"
- [10] "{other\_payment\_plans=none,existing\_credits=one,num\_dependents=one,foreign\_worker=yes}"
- ...

### Itemsets in Consequent (RHS)

- |                              |                                  |
|------------------------------|----------------------------------|
| [1] "{other_parties=none}"   | "{foreign_worker=yes}"           |
| [3] "{num_dependents=one}"   | "{other_payment_plans=none}"     |
| [5] "{housing=own}"          | "{class=good}"                   |
| [7] "{existing_credits=one}" | "{credit_history=existing paid}" |

Ce type de représentation présente l'intérêt de pouvoir comparer les règles à antécédent (ou conséquent) fixé. Néanmoins, il présente aussi une limite évidente : il n'est pas adapté pour un grand nombre d'items (la représentation graphique précédente ne permet pas l'affichage des items représentés en ligne et colonne directement sur le graphique). Ainsi, il est possible d'améliorer la lisibilité du graphique en effectuant un extrait du graphe précédent (par exemple de façon interactive ou selon un indice), ou encore en effectuant des regroupements des colonnes (`argument method = "grouped"` ).

```
# extraction des règles selon un indice de qualité
subrègles <- german.regles[quality(german.regles)$confidence > 0.9]
plot(subrègles, method = "matrix")
# extraction de façon interactive
plot(german.regles, method = "matrix", engine = "htmlwidget")
# regroupements
plot(german.regles, method = "grouped")
```

Il existe d'autres types de représentation des règles. Pour aller plus loin, on pourra se reporter à Hahsler (2017).

## 5 Conclusion

D'un point de vue technique, les méthodes de recherche de règles d'association reposent sur une optimisation algorithmique pour identifier toutes les règles au support et confiance élevés. Il faut pour cela limiter le nombre de lectures de la base et limiter la quantité de données stockée en mémoire. La plupart des algorithmes de recherche de règles d'association procèdent comme Apriori, l'algorithme fondateur. Ils commencent par rechercher les sous-ensembles fréquents d'items, puis ils extraient les règles d'association. Ces deux étapes supposent de fixer des seuils minimums pour le support et la confiance, paramètres qui ne sont pas toujours suffisants pour évaluer la pertinence d'une règle d'association. De nombreux indices de pertinence ont donc été proposés dans la littérature pour combler les lacunes de l'approche support - confiance. Ainsi, en aval de la recherche de règles d'association, il est possible de classer les règles par ordre décroissant de leur pertinence au sens d'un indice donné. Cependant, le choix d'un indice particulier dépend beaucoup des données analysées et de ce que cherche l'utilisateur. Il a été précisé que les données sont stockées sous un format présence/absence. Quand les variables sont qualitatives (à plusieurs modalités), il est alors nécessaire de les recoder par leurs indicatrices (c'est d'ailleurs ce qui a été appliqué lors de la mise en oeuvre sur le jeu German Credit en Section 3.2). Aussi, il peut parfois être utile de regrouper des items, de façon à diminuer le nombre de règles, typiquement, on pourra regrouper des produits similaires (e.g. des chaussettes de pointure différente) sous un item générique (e.g. chaussette). En pratique, rechercher des associations sur un grand ensemble d'évènements rares conduit à une profusion de règles difficiles à interpréter de par leur nombre et leur complexité. Dans ce cas, il peut être pertinent de constituer des groupes de variables plus restreints et plus homogènes via une stratégie de classification (Section 2). Les règles obtenues sont moins nombreuses et plus simples (Plasse (2006)).

Pour finir, on rappellera que la recherche de règles n'est pas une technique supervisée, qui serait à comparer à des techniques telles que la régression logistique ou les arbres de décision. Elle reste en effet une technique descriptive, qui peut être très utile pour mettre en évidence des associations entre modalités, y compris quand celles-ci portent sur une variable cible qualitative. Elle peut également trouver sa place dans l'exploration des données incomplètes afin de mettre en évidence les associations entre les modalités observées et certaines données manquantes.

## Références

Agrawal, Rakesh, Tomasz Imieliński, and Arun Swami. 1993. "Mining Association Rules Between Sets of Items in Large Databases." In *Proceedings of the 1993 ACM SIGMOD International Conference on Management*



- of Data*, 207–16. SIGMOD '93. New York, NY, USA: ACM. <http://doi.acm.org/10.1145/170035.170072> (<http://doi.acm.org/10.1145/170035.170072>).
- Agrawal, Rakesh, Ramakrishnan Srikant, et al. 1994. "Fast Algorithms for Mining Association Rules." In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, 1215:487–99.
- Brin, Sergey, Rajeev Motwani, Jeffrey D Ullman, and Shalom Tsur. 1997. "Dynamic Itemset Counting and Implication Rules for Market Basket Data." *Acm Sigmod Record* 26 (2): 255–64.
- Hahsler, Michael. 2017. "arulesViz: Interactive Visualization of Association Rules with R." *The R Journal* 9 (2): 163–75. <https://doi.org/10.32614/RJ-2017-047> (<https://doi.org/10.32614/RJ-2017-047>).
- Ong, Hian-Huat, Kok-Leong Ong, Wee-Keong Ng, and Ee Peng Lim. 2002. "CrystalClear: Active Visualization of Association Rules." In *Proceedings of the ICDM'02 International Workshop on Active Mining*, 9–12. Maebashi City, Japan.
- Plasse, Marie. 2006. "Utilisation Conjointe Des méthodes de Recherche de règles d'association Et de Classification: Contribution à l'amélioration de La Qualité Des véhicules En Production Grâce à l'exploitation Des Systèmes d'information." PhD thesis, Paris, CNAM.
- Savasere, Ashok, Edward Robert Omiecinski, and Shamkant B Navathe. 1995. "An Efficient Algorithm for Mining Association Rules in Large Databases." Georgia Institute of Technology.
- Toivonen, Hannu et al. 1996. "Sampling Large Databases for Association Rules." In *VLDB*, 96:134–45.
- Zaki, Mohammed Javeed. 2000. "Scalable Algorithms for Association Mining." *IEEE Transactions on Knowledge and Data Engineering* 12 (3): 372–90.