Chapitre 2

Fouille des motifs séquentiels dans les bases de données transactionnelles

Les motifs séquentiels représentent des informations extraites des bases de données séquentielles extraites à leur tour des bases de données transactionnelles. Contrairement au séries temporelles représentant le séquencement de données numériques, les données séquentielles représentent le séquencement des données symboliques. Elle peuvent être utilisées pour représenter des données tel que : les codes des protéines dans un génome, les actions effectués par un internaute dans un site, les opérations effectuées par un client d'une banque, le séquencement des phénomènes météorologiques, ...

2.1 Concepts de base

2.1.1 Définition d'une séquence

Étant donnée un alphabet A fini d'éléments, une séquence de longueur k est une liste ordonnée de k éléments choisis successivement dans A.

A peut être par exemple un ensemble de lettres, de mots, de codes de protéines, d'articles achetés, d'évènements d'un systèmes, ...etc

Les éléments de A peuvent avoir une dimension temporelle ou représentent seulement des transitions ou des changements.

2.1.2 Visualisation de séquences

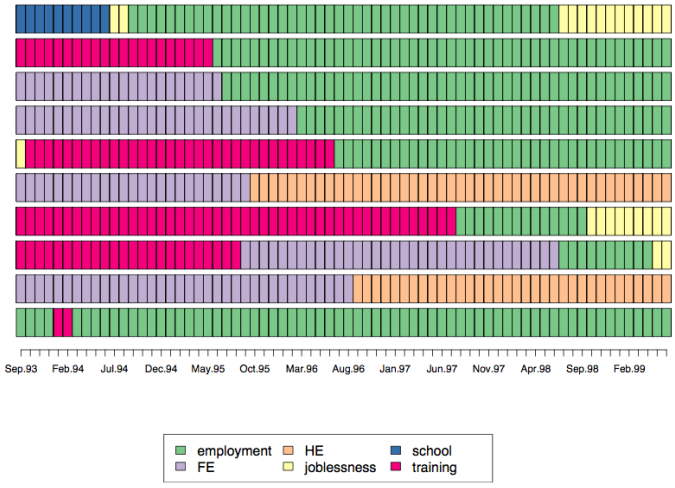
Pour visualiser les séquences, trois graphiques de base sont utilisés i-plot, f-plot et d-plot. Nous allons prendre comme exemple les données mvad utilisées par McVicar et

Anyadike-Danes (2002) sur la transition entre formation et emploi en Irlande du Nord. Ces données proviennent d'une enquête auprès 712 jeunes irlandais et indiquent les états mensuels successifs de chaque individu entre septembre 1993 et juin 1999. Les états sont :

- EM : en emploi
- FE : formation secondaire
- HE : formation supérieure
- JL : au chômage
- SC : école
- TR : en stage ou apprentissage.

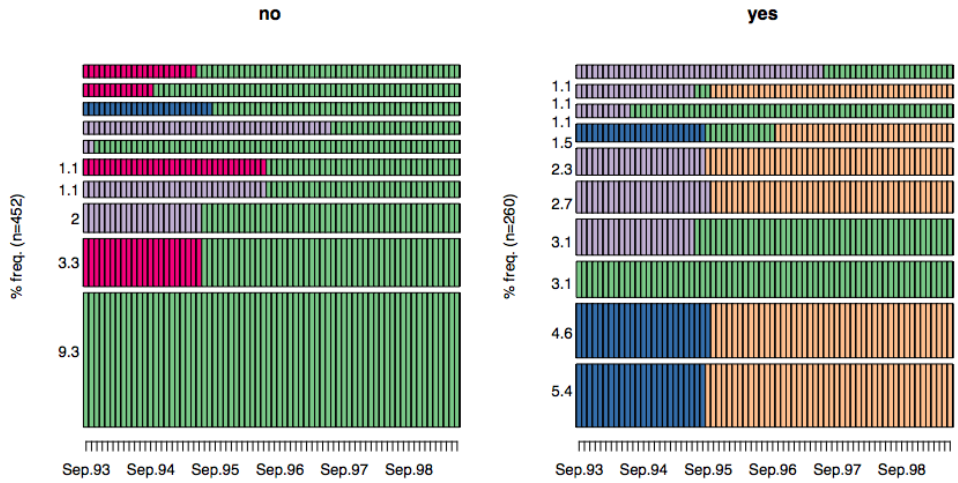
2.1.2.1 Plot de séquences individuelles (i-plot)

Le plot de séquences individuelles (i-plot) visualise chaque séquence par une barre horizontale. La figure suivante montre le i-plot de 10 séquences représentant l'évolution des carrières de 10 personnes.



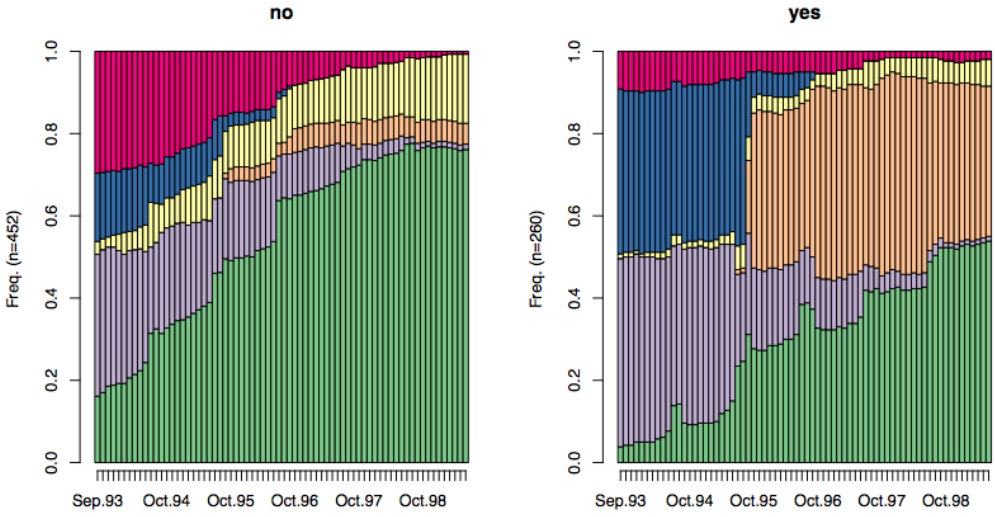
2.1.2.2 Plot de séquences selon la fréquence (f-plot)

Semblable au précédent mais la hauteur des séquences est relative à leur fréquence.



2.1.2.3 Séquences des distributions transversales (d-plot)

Représente les distributions des états selon la position (le temps).



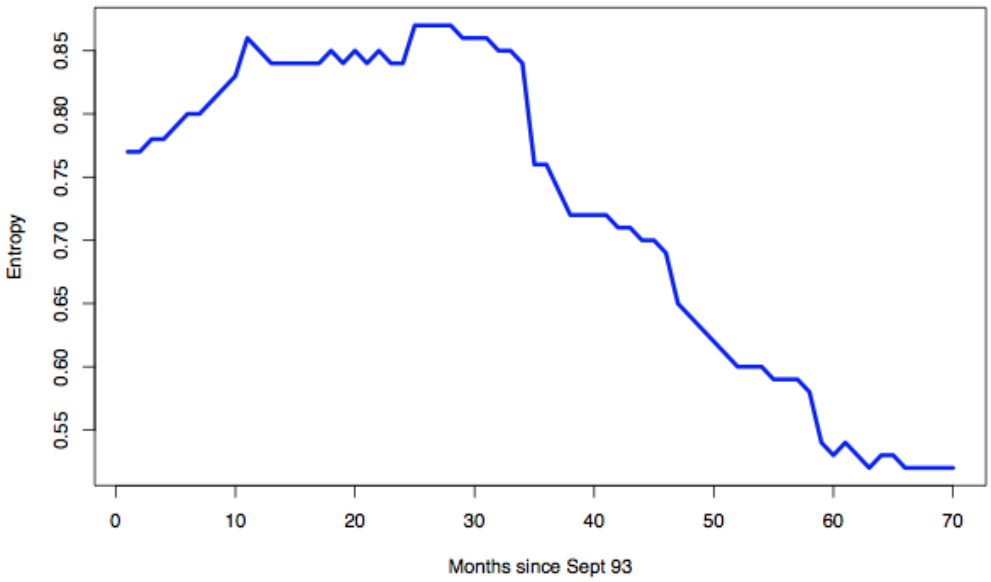
2.1.2.4 Graphique de la série des entropies

Représente l'entropie des séquence pour chaque état temporel.

$$h(p_1, \ldots, p_a) = -\sum_{i=1}^{a} p_i \; log_2(p_i)$$

Où a représente la taille de l'alphabet A et p_i représente la fréquence de l'élément i de l'alphabet dans l'état temporel considéré.

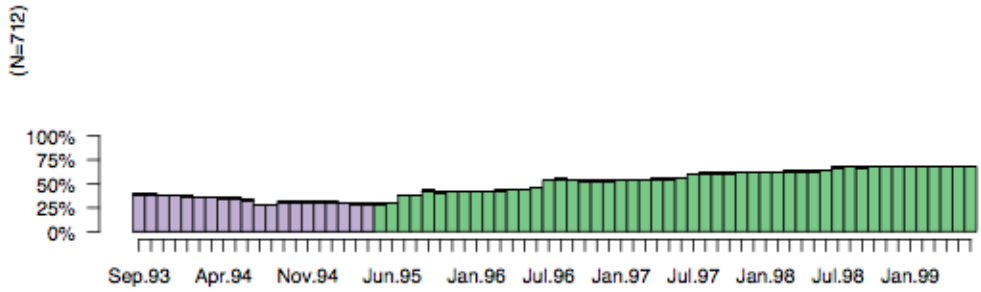
La figure suivante représente la série de l'entropie d'une séquence de données pour des intervalles de temps donnés (des mois) :



2.1.3 Caractéristiques des séquences

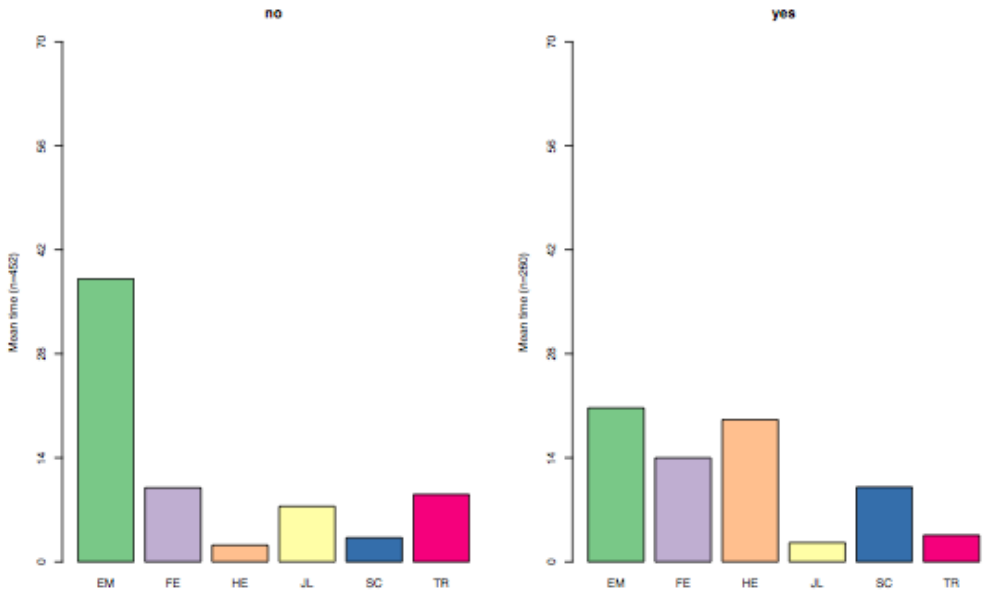
2.1.3.1 Séquence des états modaux

Représente les états les plus fréquents dans les séquences et leurs fréquences.



2.1.3.2 Durée moyenne passée dans chaque état

Représente la durée moyenne passée dans chacun des états pour chaque séquence individuelle.



2.1.3.3 Taux de transition

C'est une caractéristique longitudinale individuelle (concerne chaque séquence). Elle Représente l'estimation des probabilités de transition d'un état à l'autre pour une séquence donnée : $p(x_{it}|x_{j(t?1)})$

	[-> EM]	[-> FE]	[-> HE]	[-> JL]	[-> SC]	[-> TR]
[EM ->]	0.9864	0.0020	0.0025	0.0065	0.0004	0.0022
[FE ->]	0.0279	0.9514	0.0066	0.0090	0.0010	0.0041
[HE ->]	0.0102	0.0002	0.9872	0.0019	0.0000	0.0005
[JL ->]	0.0418	0.0084	0.0023	0.9387	0.0005	0.0084
[SC ->]	0.0142	0.0081	0.0182	0.0056	0.9509	0.0029
[TR ->]	0.0383	0.0036	0.0000	0.0136	0.0004	0.9442

2.1.3.4 Entropie longitudinale

C'est une caractéristique longitudinale individuelle (concerne chaque séquence). Elle est nulle si la séquence garde le même état durant toute la période d'observation (A-A-A-A-A-A-A par exemple) et elle est maximale si elle contient le même nombre pour chacun des états de l'alphabet (par exemple A-A-B-B-C-C-D-D)

$$h_l(p_1, \ldots, p_a) = \frac{-\sum_{i=1}^{a} p_i \ log_2(p_i)}{h(A)}$$

avec p_i proportion de positions dans l'état i.

2.1.3.5 Turbulence

L'entropie considère uniquement la fréquence des états et néglige leur ordre. La **Turbulence** est une mesure alternative proposée par Elzinga and Liefbroer en 2007 qui tient compte indirectement de l'ordre. Elle est basée sur :

- le nombre $\phi(x)$ de sous-séquences distinctes contenues dans la suite des états distincts composant la séquence. Par exemple $x = S - U - M - C$ (16 sous-séquences : $\Phi, S, SU, SM, SC, SUM, SUC, SMC, SUMC, U, UM, UC, UMC, M, MC, C$) plus turbulente que $y = S - U - S - C$ (15 sous-séquences).
- la variance des durées passées dans chacun des états distincts. Par exemple la séquence S/10-U/2-M/132 représente une trajectoire moins turbulente que S/48-U/48-M/48.

Pour calculer la turbulence d'une séquence x, on commence par calculer la série des durées passées dans chacun des états distincts. Par exemple la série correspondant à la séquence $x = S - S - S - S - U - U - M - M - M$ est la série $(4, 2, 3)$.

La turbulence est alors calculée pour une séquence x par la formule :

$$T(x) = log_2(\phi(x)\frac{s^2_{t,max}(x) + 1}{s^2_t(x) + 1})$$

tel que :

- s^2_t est la variance des durées passées dans chacun des états distincts et $s^2_{t,max}$ est la valeur maximum que peut atteindre cette variance compte tenu de la durée totale de la séquence. Ce maximum est

$$s^2_{t,max}(x) = (n-1)(1-\bar{t})$$

où $\bar{t}$ est la moyenne des durées consécutives passées dans les états distincts.

$\bar{t}$ = durée de la séquence / nombre de ses états distincts

2.1.4 Mesures de similarité entre séquences

Les mesures de similarité ont une importance particulière pour les séquences parce qu'elles sont la base de plusieurs tâches effectuées dans ce domaine. On trouve plusieurs

mesures dont l'utilisation et l'importance varient selon l'application visée.

2.1.4.1 La distance sac-de-caractères

La distance sac de caractères (bag of characters) permet de calculer la distance entre deux séquences en les considérant comme des sacs d'états c-à-d sans tenir compte de l'ordre des états dans ces séquences. L'importance est donnée plutôt à la fréquence des états. On représente chaque séquence par un vecteur des fréquences d'apparition des états de l'alphabet dans cette séquence. Une séquence S_i est représentée par un vecteur $V_i = (V_{i1}, \cdots, V_{im})$ où chaque valeur V_{ik} représente la fréquence d'apparition de l'état d'indice k de l'alphabet dans la séquence S_i. La distance entre deux séquences S_1 et S_2 est donnée par le cosinus de l'angle θ entre les deux vecteurs représentant les deux séquences .

$$D(S_1, S_2) = cos(\theta) \quad = \frac{\langle V_1, V_2 \rangle}{\|V_1 \cdot V_2\|} \quad = \frac{\sum_{k=1}^{m} V_{1k} V_{2k}}{\sqrt{(\sum_{k=1}^{m} V_{1k}^2) \cdot (\sum_{k=1}^{m} V_{2k}^2)}}$$

2.1.4.2 La distance *p-Spectrum*

Elle permet de calculer le nombre de sous-séquences contigües que peuvent avoir deux séquences en commun. Cette distance tire sont principe du fait que plus le nombre de sous-séquences communes entre deux séquences est important, moins est la distance entre ces deux séquences.

Soit l'alphabet A, l'espace associé à la distance p-spectrum est de dimension $card(A)^p$, c-à-d le nombre de séquences de longueur p pouvant être composées de l'alphabet A.

Pour une distance *p-Spectrum*, chaque séquence s est représentée, donc, par un vecteur $V^p = \{Card(V_u^p)/u \in A^p\}$, où chaque élément V_u^p est une sous-séquence de s de longueur p. Plus formellement :

$$\Phi_u^p(s) = card(\{(v_1, v_2)|s = v_1 u v_2\}, \quad u \in A^p)$$

Où $v_1 u v_2$ désigne la concaténation des séquences v_1, u et v_2, et $\Phi_u^p(s)$ désigne l'image de s dans l'espace de caractéristiques. La distance *p-Spectrum* est alors :

$$\begin{aligned} K_p(s_1, s_2) &= \langle \Phi^p(s_1), \Phi^p(s_2) \rangle \\ &= \sum_{u \in \Sigma^p} \Phi_u^p(s_1) \Phi_u^p(s_2) \end{aligned}$$

La distance compte, donc, toutes les sous-séquences de longueur p dans les deux séquences puis calcule la somme des produits. Un cas particulier où $p = 1$ est le *bag of*

characters.

2.1.4.3 Longest Common Prefix (LCP)

Le LCP donne le plus long préfixe commun entre deux séquences. Le LLCP est sa longueur. La distance est donnée par

$$D_{LCP}(S_1, S_2) = 1 - \frac{LLCP(1,1)}{\sqrt{|x| \bullet |y|}}$$

$|x|$ est la longueur de la séquence x.

$$LLCP(i,j) = \begin{cases} S_1[i] + LLCP(i+1, j+1) & si\ S_1[i] = S_2[j] \\ "" & sinon \end{cases}$$

2.1.4.4 Longest Common Subsequence (LCS)

Cette mesure calcule la taille des sous-séquences partagées par 2 séquences. Par exemple pour les deux séquences $x = 1-1-1-2-2-3-3$ et $y = 1-1-1-4-4-3-3$, la LCS est $1-1-1-3-3$ et la LLCS sa longueur est de 5 .

La distance associée est :

$$D_{LCS}(i,j) = 1 - \frac{LLCS(i,i)}{\sqrt{ij}}$$

$$LLCS(i,j) = \begin{cases} 1 + LLCS(i-1, j-1) & si\ S_1[i] = S_2[j] \\ max\{LLCS(i-1,j), LLCS(i,j-1)\} & sinon \end{cases}$$

2.2 Méthodes de fouille des données séquentielles

La fouille des données séquentielles représente une part très importante du domaine de data mining. La tâche la plus active est celle de recherche des motifs fréquents séquentiels.

2.2.1 Recherche des motifs fréquents séquentiels

La fouille des motifs séquentiels est le processus permettant d'extraire certains motifs séquentiels dont le support dépasse un minimum de support prédéfini. Contrairement aux motifs fréquents indiquant les relations intra-transactions, les motifs fréquents séquentiels représentent les corrélations entre les transactions.

2.2.1.1 Définitions et propriétés

- Une transaction constitue, pour un client C, l'ensemble des items achetés par C à une même date. Dans une base de données client, une transaction s'écrit sous la forme d'un ensemble : <id-client, id-date, itemset>.
- Une base de données séquentielle est composée d'éléments ou événements ordonnés.

SID	séquences
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

- Une séquence est une liste ordonnée, non vide, d'itemsets notée $< s_1 s_2 ... sn >$ où s_j est un itemset (une séquence est donc une suite de transactions qui apporte une relation d'ordre entre les transactions).
- Une séquence de données est une séquence représentant les achats d'un client. Soit $T_1, T_2, ..., T_n$ les transactions d'un client, ordonnées par dates d'achat croissantes et soit $itemset(T_i)$ l'ensemble des items correspondants à T_i, alors la séquence de données de ce client est $< itemset(T_1) itemset(T_2) ... itemset(T_n) >$.

  Par exemple, soit C un client et $S =< (C)(DE)(H) >$, la séquence de données représentant les achats de ce client. S peut être interprétée par "C a acheté l'item C, puis en même temps les items D et E et enfin l'item H"
- Soit $s_1 =< a_1 a_2 ... a_n >$ et $s_2 =< b_1 b_2 ... b_m >$ deux séquences de données. s_1 est une sous séquence de s_2 si et seulement si il existe $i_1 < i_2 < ... < i_n$ des entiers tels que $a_1 \subset b_{i1}, a_2 \subset b_{i2}, ... a_n \subset bi_n$.

  Par exemple, La séquence $s_1 =< (C)(DE)(H) >$ est une sous-séquence de la séquence $s_2 =< (G)(CH)(I)(DEF)(H) >$ car $(C) \subset (CH), (DE) \subset (DEF)$ et $(H) \subset (H)$. En revanche $< (C)(E) >$ n'est pas une sous séquence de $< (CE) >$ (et vice versa).
- Un client supporte une séquence s (fait partie du support pour s) si s est une sous séquence de la séquence de données de ce client.
- Le support d'une séquence s est calculé comme étant le pourcentage des clients qui supportent s.
- Soit $MinSupp$ le support minimum fixé par l'utilisateur. Une séquence qui vérifie le support minimum (i.e. dont le support est supérieur à $minSup$) est une séquence fréquente.

– Soit une base de données DB, l'ensemble L^{DB} des séquences fréquentes maximales (également notées motifs séquentiels) est constitué de toutes les séquences fréquentes telles que pour chaque séquence s dans L^{DB}, s n'est est une sous séquence d'aucune autre séquence de L^{DB}. Le problème de la recherche de séquences maximales consiste à trouver l'ensemble L^{DB}.
– Si une séquence s n'est pas fréquente, aucune de ses sur-séquences n'est fréquente (décroissance de support)
– Si une séquence s est fréquente, alors toutes ses sous-séquences le sont aussi.

Exemple :

Considérons la base des transactions suivante :

Client	Date	Items
C_1	01/01/2004	B,F
C_1	02/02/2004	B
C_1	04/02/2004	C
C_1	18/02/2004	H,I
C_2	11/01/2004	A
C_2	12/01/2004	C
C_2	29/01/2004	D,F,G
C_3	05/01/2004	C,E,G
C_3	12/02/2004	A,B
C_4	06/02/2004	B,C
C_4	07/02/2004	D,G
C_4	08/02/2004	I

Avec un support minimum de 50% (i.e. pour qu'une séquence soit retenue, il faut que deux clients dans la base de données supportent cette séquence), les séquences fréquentes maximales sont alors les suivantes : $< (A) >, < (F) >, < (B)(I) >, < (C)(I) >$ et $< (C)(D, G) >$. La première fait partie des achats de C2 et C3, alors que la dernière apparaît dans les séquences de données des clients C2 et C4.

En pratique, les motifs séquentiels candidats sont beaucoup plus nombreux que motifs traditionnels et la méthode de leur recherche est d'une grande importance.

Beaucoup d'algorithmes ont été proposée pour réduire l'espace de recherche et extraire les motifs séquentiels les plus significatifs.

2.2.1.2 Algorithme AprioriAll

L'algorithme AprioriAll est une adaptation de l'algorithme de base Apriori pour les séquences, où la génération des candidats et le calcul de support sont faits d'une manière différentes. Il utilise une base transactionnelle sur laquelle il effectue les étapes suivantes :

1. Étape de tri : consiste à trier la base selon l'identificateur du client et la date de la transaction. L'objectif est de mapper la base transactionnelle en une base séquentielle.
2. Étape Litemset : consiste à trouver toutes les séquences fréquentes de longueur 1 en utilisant l'algorithme Apriori. Attention, le support est le nombre de clients et non pas le nombre de transactions.
3. Étape de transformation : consiste à mapper chaque transaction à tous les motifs fréquents séquentiels contenus dans la transaction puis mapper chaque motif fréquent séquentiel à un entier.
4. Étape de séquence : consiste à trouver toutes les séquences fréquentes à l'aide d'un algorithme ressemblant à l'algorithme Apriori :

Algorithme Étape Séquence

$L_1 \leftarrow$ {Séquences fréquentes de longueur 1 } ;
$k \leftarrow 2$;
tantque $L_k \neq \phi$ **faire**
  $C_k \leftarrow$ GénérerCandidats(L_{k-1})
  pour chaque séquene c d'un client dans la base séquentielle **faire**
    Incrémenter les compteurs de tous les candidats dans C_k contenus dans c
  fin pour
  $L_k \leftarrow$ {Candidats dans C_k ayant le support minimum}
  $k \leftarrow k + 1$;
fin tantque
Retourner $\cup_k L_k$

La procédure GénérerCandidats(L_{k-1}) génère l'ensemble C_k de motifs séquentiels candidats de longueur k à partir de l'ensemble L_{k-1} en suivant les étapes suivantes :

(a) Jointure : Lier L_{k-1} avec L_{k-1} avec la condition d'avoir les mêmes sous-séquences préfixes de longueur $k-2$,

(b) Lier les séquences avec elles-mêmes,

(c) Élagage : Supprimer toutes les séquences ayant des sous-séquences non fréquentes.

Autrement dit :

insert into C_k

select $p.litemset_1, ..., p.litemset_{k-1}, q.litemset_{k-1}$

from $L_{k-1}\ p, L_{k-1}\ q$

where $p.litemset_1 = q.litemset_1, ..., p.litemset_{k-2} = q.litemset_{k-2}$

Exemples de génération :

Frequent 3-sequences	Candidate 4-sequences after join	Candidate 4-sequences after pruning
123	1233	1234
	1234	
124	1243	
	1244	
134	1344	
234	2344	

⟨ {1 5} {2} {3} {4} ⟩
⟨ {1} {3} {4} {3 5} ⟩
⟨ {1} {2} {3} {4} ⟩
⟨ {1} {3} {5} ⟩
⟨ {4} {5} ⟩

Base séquentielle

Sequence	Support
⟨ 1 ⟩	4
⟨ 2 ⟩	2
⟨ 3 ⟩	4
⟨ 4 ⟩	4
⟨ 5 ⟩	2

1-Séquence

Sequence	Support
⟨ 1 2 ⟩	2
⟨ 1 3 ⟩	4
⟨ 1 4 ⟩	3
⟨ 1 5 ⟩	3
⟨ 2 3 ⟩	2
⟨ 2 4 ⟩	2
⟨ 3 4 ⟩	3
⟨ 3 5 ⟩	2
⟨ 4 5 ⟩	2

2-Séquence

Sequence	Support
⟨ 1 2 3 ⟩	2
⟨ 1 2 4 ⟩	2
⟨ 1 3 4 ⟩	3
⟨ 1 3 5 ⟩	2
⟨ 2 3 4 ⟩	2

3-Séquence

Sequence	Support
⟨ 1 2 3 4 ⟩	2

4-Séquence

Exemple de ApprioriAll :

Considérons la base suivante et supposons un support minimum de 40% (0.4)

Transaction Time	Customer Id	Items Bought
June 10 '93	2	10, 20
June 12 '93	5	90
June 15 '93	2	30
June 20 '93	2	40, 60, 70
June 25 '93	4	30
June 25 '93	3	30, 50, 70
June 25 '93	1	30
June 30 '93	1	90
June 30 '93	4	40, 70
July 25 '93	4	90

L'algorithme AprioriAll procède comme suit :

1. À la première étape de tri, la base est trié selon le code du client "Custumer Id" puis transformée en une base séquentielle :

Customer Id	TransactionTime	Items Bought
1	June 25 '93	30
1	June 30 '93	90
2	June 10 '93	10, 20
2	June 15 '93	30
2	June 20 '93	40, 60, 70
3	June 25 '93	30, 50, 70
4	June 25 '93	30
4	June 30 '93	40, 70
4	July 25 '93	90
5	June 12 '93	90

Customer Id	Customer Sequence
1	⟨ (30) (90) ⟩
2	⟨ (10 20) (30) (40 60 70) ⟩
3	⟨ (30 50 70) ⟩
4	⟨ (30) (40 70) (90) ⟩
5	⟨ (90) ⟩

2. À la deuxième étape de Litemset, les motifs de longueur 1 sont trouvé :

1-Itemset	Support
(30)	4
(90)	3
(30 90)	1
(10)	1
(20)	1
(10 20)	1
(40)	2
(60)	1
(70)	3
(40 60)	1
(40 70)	2
(60 70)	1
:	1

Donc les séquences fréquentes de longueur 1 sont {(30), (90), (40), (70), (40 70) }

3. À la troisième étape, les séquences de longueur 1 sont mappées à des entiers :

Large Itemsets	Mapped To
(30)	1
(40)	2
(70)	3
(40 70)	4
(90)	5

Puis mapper chaque transaction en les séquences fréquentes qu'elle contient :

Customer Id	Original Customer Sequence	Transformed Customer Sequence	After Mapping
1	⟨ (30) (90) ⟩	⟨ {(30)} {(90)} ⟩	⟨ {1} {5} ⟩
2	⟨ (10 20) (30) (40 60 70) ⟩	⟨ {(30)} {(40), (70), (40 70)} ⟩	⟨ {1} {2, 3, 4} ⟩
3	⟨ (30 50 70) ⟩	⟨ {(30), (70)} ⟩	⟨ {1, 3} ⟩
4	⟨ (30) (40 70) (90) ⟩	⟨ {(30)} {(40), (70), (40 70)} {(90)} ⟩	⟨ {1} {2, 3, 4} {5} ⟩
5	⟨ (90) ⟩	⟨ {(90)} ⟩	⟨ {5} ⟩

4. À la quatrième étape, on génère les séquences candidates de longueur 2 à partir des séquences de longueur 1 en utilisant la procédure GénérerCandidats

2-Itemset	Support	2-Itemset	Support	2-Itemset	Support
11	0	31	0	51	0
12	2	32	0	52	0
13	2	33	0	53	0
14	2	34	0	54	0
15	2	41	0	55	0
21	0	42	0		
23	0	43	0		
24	0	44	0		
25	1	45	1		

Les séquences fréquentes de longueur 2 et candidats de longueur 3 :

2-Itemset fréquent	Support	Candidat 3-itemset après jointure	Candidat 3-itemset après élagage
12	2	122	
		123	
		124	
		125	
13	2	132	
		133	
		134	
		135	
14	2	142	
		143	
		144	
		145	
15	2	152	
		153	
		154	
		155	

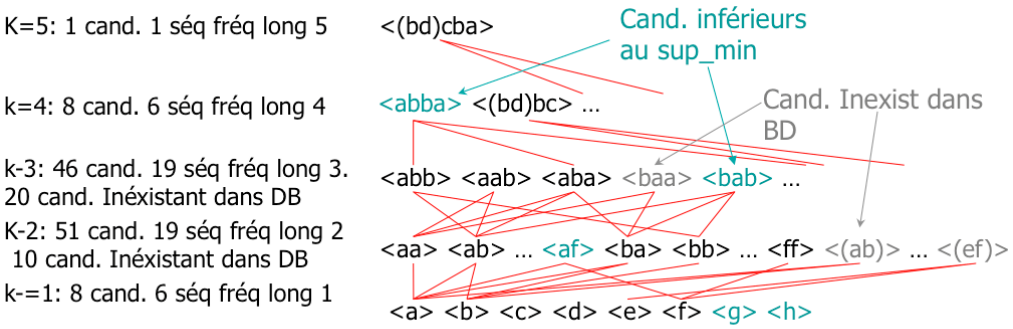
Les séquences fréquentes : (1,2,3,4,5,12,13,14,15) c'est à dire { (30), (40), (70), (40 70), (90), { (30) (40) }, { (30) (70) }, { (30) (40 70) }, { (30) (90) } }

2.2.1.3 Algorithme GSP

La méthode GSP (Generalized Sequential Patterns) a été l'une des premières propositions pour résoudre la problématique des motifs séquentiels, elle repose sur l'algorithme Apriori. Elle se résume dans les étapes suivantes :

1. Initialement, chaque item dans la base est un candidat de longueur 1
2. Pour chaque niveau (i.e., sequences de longueur k) faire
   - Scanner la base pour déterminer le support de chaque séquence dans la base
   - Garder les séquences fréquentes
   - Générer les séquences candidates de longueur k+1 à partir des séquences fréquentes de longueur k en utilisant la méthode de jointure et élagage.
3. Répéter jusqu'à ce qu'aucune séquence candidate ne peut être trouvé.

Exemple :



min_sup =2

Seq. ID	Sequence
10	<(bd)cb(ac)>
20	<(bf)(ce)b(fg)>
30	<(ah)(bf)abf>
40	<(be)(ce)d>
50	<a(bd)bcb(ade)>

2.2.1.4 Algorithme SPADE

SPADE (Sequential PAttern Discovery using Equivalence classes) utilise les listes TIDs. Lors de la génération des séquences candidates, les jointures se font de la façon suivante :

- Élément contre élément : (P B) jointe avec (P D) en (P B D).
- Élément contre séquence :(P B) jointe avec (P) (A) en (P B) (A).
- Séquence contre séquence : (P) (A) jointe avec (P) (F) en (P) (A F), (P) (A) (F) et (P) (F) (A).

Le principal avantage de SPADE par rapport à GSP est l'utilisation de listes verticales temporaires pour les jointures. Ces listes, bornées, sont beaucoup plus petites et rapides à générer que celles utilisées par GSP ce qui permet d'améliorer grandement le temps de calcul nécessaire ainsi que de réduire les besoins de mémoire.

2.3 Fouille des motifs séquentiels sous contraintes