Mineração de itemsets e regras de associação

Prof. Marcelo Keese Albertini
Faculdade de Computação
Universidade Federal de Uberlândia

Conteúdo

- Itemsets, Tidsets, Base binária, Base Vertical
- Suporte e confiança
- Regras de associação
- Força bruta
- A-priori
- Eclat
- FP-Growth

- Ler capítulos 8 e 9 de Zaki e Meira
- `http://www.dataminingbook.info/`

Problema: análise de cestas de compras

- Uma cadeia de supermercados vende 100000 itens diferentes
- Compras contém diferentes variações de itens
- Como identificar quais são os produtos mais frequentemente associados?
  - Linguiça e carvão
  - Fralda e cerveja

Exemplos

- Sugestão de itens para grávidas: link
- *Market Basket Analysis*: quais itens são comprados juntos em supermercado?
- Análise de sites/interfaces: quais conjuntos de páginas/janelas são acessadas na mesma sessão?
- Análise de tópicos: quais palavras são usadas juntas em documentos?
- Detecção de plágio: quais sentenças são usadas em documentos diferentes?

Exemplo: Market Basket Analysis

- Conjunto de itens: $\mathcal{I}$ pode ser todos produtos vendidos no supermercado
- Itemset: qualquer combinação de produtos em $\mathcal{I}$: $\{carne, cerveja, fralda, leite\}$

Exemplo: análise de sites/interfaces

- Conjunto de itens:
  $\mathcal{I} = \{x_1, x_2, \ldots, x_m\} = \{\text{todas as páginas do site}\}$
- Itemsets $X \subseteq \mathcal{I}$:
  - Usuário que visita páginas `main`, `laptop` e `desconto` também visita `carrinho-de-compra` e `pagamento`.
  - $X = \{\textit{main}, \textit{laptop}, \textit{carrinhodecompra}, \textit{pagamento}, \textit{desconto}\}$

Exemplo: análise de tópicos

- Conjunto de itens:
  $\mathcal{I}$ = todas as palavras de todos os documentos em análise
- Itemset = "conjunto de palavras em um documento em análise"

Exemplo: detecção de plágio

- Conjunto de itens:
  $\mathcal{I}$ = todas as **sentenças** de todos os documentos em análise
- Itemsets = "conjunto de **sentenças** em um documento em análise"

Mineração de conjuntos de itens - Itemsets

- Análise de cesta de itens (= itemset)
- Objetivo: obter regras de associação
- Exemplos de regras:
  - Fralda $\rightarrow$ Cerveja
  - Política, Imigrantes $\rightarrow$ Trump, Mexicanos

Exemplo: MovieLens.org

- Avaliações sobre 164979 filmes
  - `https://grouplens.org/datasets/movielens/`
  - `http://files.grouplens.org/datasets/movielens/ml-latest-small.zip`
- Cada usuário avalia diversos filmes
- Pré-processamento:
  - `ratings.csv` – filtrar avaliações de usuários
  - `movies.csv` – filmes duplicados
- Objetivo: descobrir associações interessantes entre filmes

Exemplo MovieLens.org: Pré-processamento

```
require(arules) # códigos de Borgelt
require(dplyr)

movies  <- read.csv("ml-latest-small/movies.csv")
titles <- as.character(movies$title)

for (t in which(duplicated(titles))) {
   titles[t] = paste(titles[t],"(dup)",sep="")
}

movies$title <- titles # 2 titulos são deduplicados
```

Exemplo MovieLens.org: Pré-processamento

```
ratings <- read.csv("ml-latest-small/ratings.csv")
ratings5 <- filter(ratings, rating ==5) %>%
              select(userId, movieId)

# filtra os titulos de filmes com rating == 5
movies5  <- filter(movies, movieId %in% ratings5$movieId)

tableRatings5 <- table(ratings5)
colnames(tableRatings5) <- as.character(movies5$title)

# tipo "transactions" é usado para representar base
transacoes5 <- as(tableRatings5 > 0, "transactions")
```

Transações

- Transação $\langle t, X \rangle$ é um registro de um id t, ou tid, com um itemset X na base
- O conjunto $\mathcal{T}$ contém todos os *id* de transações
- Um conjunto $T \subseteq \mathcal{T}$ é um tidset
- A função $\mathbf{i}(tid) = \{x | \forall tid \in T, t \text{ contém } x\}$ retorna o conjunto de cada item x que a transação com id t possue

tid	$\mathbf{i}(tid)$
1	ABDE
2	BCE
3	ABDE
4	ABCE
5	ABCDE
6	BCD

Tidsets

- Um subconjunto de tids $T \subseteq \mathcal{T}$ de transações é um tidset
- A função $\mathbf{t}(X) = \{t | t \in \mathcal{T}$ e t contém $X\}$ retorna o conjunto de *tids* de transações que contém todos os itens no itemset X

	x				
	A	B	C	D	E
$\mathbf{t}(\{x\})$	1 3 4 5	1 2 3 4 5 6	2 4 5 6	1 3 5 6	1 2 3 4 5

- $\mathbf{t}(\{A\}) = \{1, 3, 4, 5\}$
- $\mathbf{t}(\{A, B\}) = \{1, 3, 4, 5\}$
- $\mathbf{t}(\{B, C\}) = \{2, 4, 5\}$
- $\mathbf{t}(\{A, B, C, D, E\}) = \{5\}$

Representação de bases de dados

- Uma base binária é uma relação de itens e transações

$$\mathbf{D} \subseteq \mathcal{T} \times \mathcal{I}$$

- Uma transação t tem um item x se $(t, x) \in \mathbf{D}$

D	A	B	C	D	E
1	1	1	0	1	1
2	0	1	1	0	1
3	1	1	0	1	1
4	1	1	1	0	1
5	1	1	1	1	1
6	0	1	1	1	0

Base binária

t	$\mathbf{i}(t)$
1	ABDE
2	BCE
3	ABDE
4	ABCE
5	ABCDE
6	BCD

Transações

	x				
	A	B	C	D	E
$\mathbf{t}(\{x\})$	1 3 4 5	1 2 3 4 5 6	2 4 5 6	1 3 5 6	1 2 3 4 5

Tidsets

Base vertical

Definições: Itemset

k-itemset : conjunto com k itens (cardinalidade k)

- $X = \{\textit{fralda}, \textit{cerveja}\}$ é um 2-itemset
- $X = \{\text{política}, \text{imigrantes}, \text{trump}, \text{mexicanos}\}$ é um 4-itemset

$\mathcal{I}$: conjunto de todos itemsets

$\mathcal{I}^{(k)}$: conjunto de todos k-itemsets

Avaliação de Itemsets

- Suporte: número de transações contendo um itemset X

$$sup(X) = |\{tid | X \subseteq \mathbf{i}(tid)\}| = |\mathbf{t}(X)|$$

- Suporte relativo:

$$rsup(X) = \frac{sup(X)}{|\mathbf{D}|}$$

  onde $|\mathbf{D}|$ é o número de transações na base
- Limiar de Suporte Mínimo: *minsup*
  - Se $sup(X) \geq minsup$ então X é frequente!
- $\mathcal{F}$ é o conjunto de todos itemsets frequentes
- $\mathcal{F}^{(k)}$ é o conjunto dos k-itemsets frequentes

k-itemsets frequentes: $\mathcal{F}^{(k)}$

tid	$\mathbf{i}(tid)$
1	ABDE
2	BCE
3	ABDE
4	ABCE
5	ABCDE
6	BCD

$$\mathcal{F}^{(1)} = \{A, B, C, D, E\}$$
$$\mathcal{F}^{(2)} = \{AB, AD, AE, BC, BD, BE, CE, DE\}$$
$$\mathcal{F}^{(3)} = \{ABD, ABE, ADE, BCE, BDE\}$$
$$\mathcal{F}^{(4)} = \{ABDE\}$$

suporte mínimo $minsup = 3$

Itemsets frequentes com suporte mínimo $minsup = 3$

tid	$\mathbf{i}(tid)$
1	ABDE
2	BCE
3	ABDE
4	ABCE
5	ABCDE
6	BCD

Table 8.1. Frequent itemsets with $minsup = 3$

sup	itemsets
6	B
5	E, BE
4	$A, C, D, AB, AE, BC, BD, ABE$
3	$AD, CE, DE, ABD, ADE, BCE, BDE, ABDE$

Regra de associação

- $X \rightarrow Y$, com X e Y sendo itemsets disjuntos $X \cap Y = \emptyset$
- Exemplos de regras:
  - Fralda $\rightarrow$ Cerveja
  - Política, Imigrantes $\rightarrow$ Trump, Mexicanos

Suporte de uma regra de associação

Uma regra de associação $X \longrightarrow Y$ tem suporte

$$sup(X \rightarrow Y) = |t(XY)| = sup(X \cup Y) = sup(XY)$$

E tem suporte relativo

$$rsup(X \rightarrow Y) = \frac{sup(XY)}{|\mathbf{D}|}$$

Usuário pode definir um suporte mínimo *minsup* para as regras desejadas.

Confiança de uma regra de associação

$$conf(X \rightarrow Y) = \frac{sup(XY)}{sup(X)}$$

- Regra é forte se tem no mínimo confiança de *minconf*

tid	$\mathbf{i}(tid)$
1	ABDE
2	BCE
3	ABDE
4	ABCE
5	ABCDE
6	BCD

- $s = sup(BC \rightarrow E) = sup(BCE) = 3$
- $c = conf(BC \rightarrow E) = \frac{sup(BCE)}{sup(BC)} = 3/4$

ALGORITHM 8.1. Algorithm BRUTEFORCE

```
BRUTEFORCE (D, I, minsup):
1  F ← ∅ // set of frequent itemsets
2  foreach X ⊆ I do
3      sup(X) ← COMPUTESUPPORT (X, D)
4      if sup(X) ≥ minsup then
5          F ← F ∪ {(X, sup(X))}
6  return F

COMPUTESUPPORT (X, D):
7  sup(X) ← 0
8  foreach ⟨t, i(t)⟩ ∈ D do
9      if X ⊆ i(t) then
10         sup(X) ← sup(X) + 1
11 return sup(X)
```

- $\mathcal{I}$ é o conjunto de itens que formam itemsets
- $\mathbf{D}$ é a base binária
- $\mathbf{i}(tid)$ é o conjunto de itens da transação tid

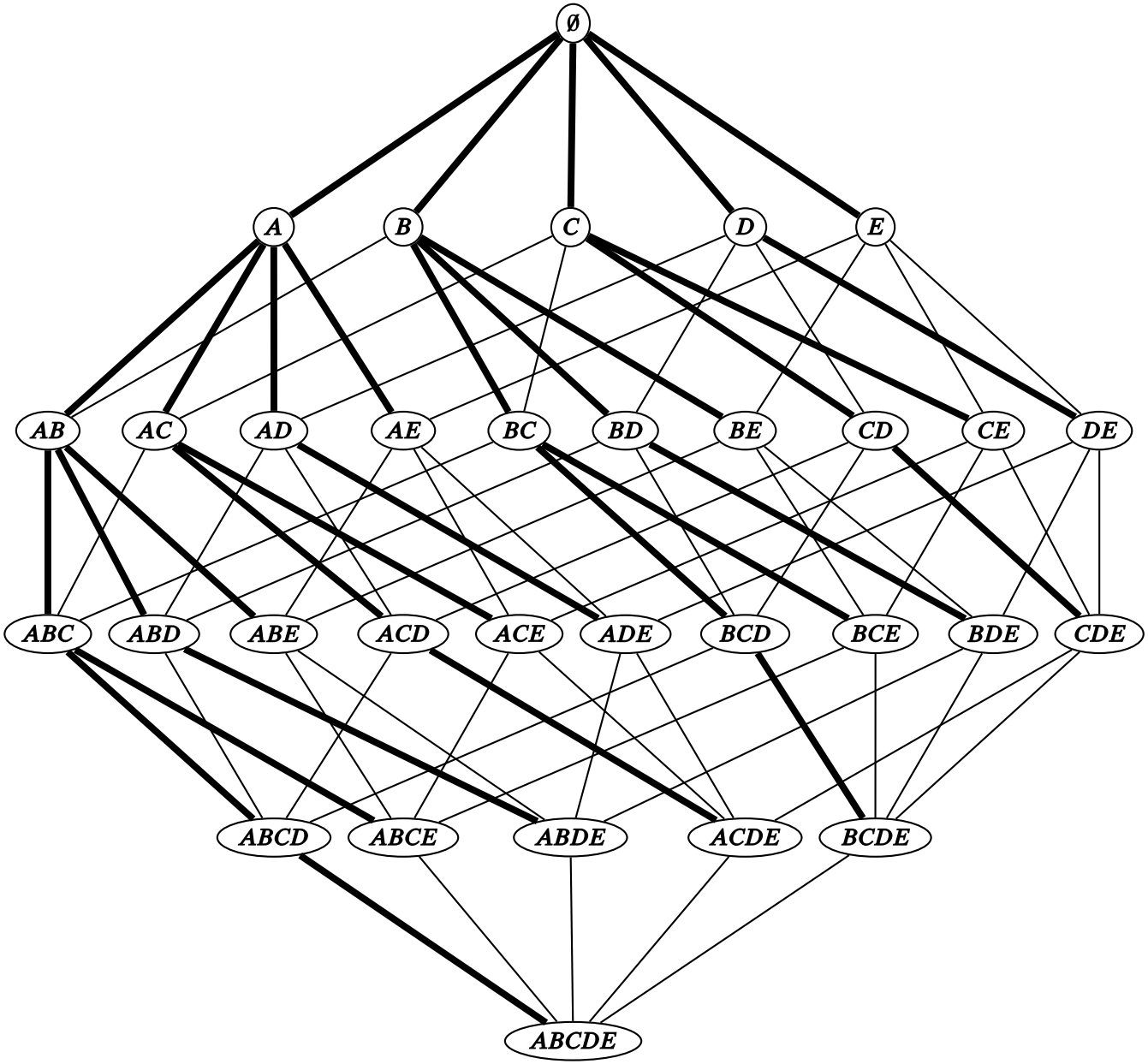


Figure 8.2. Itemset lattice and prefix-based search tree (in bold).

Força bruta explora todo o grafo.

Avaliação do algoritmo de força bruta

- Geração dos candidatos: $O(2^{|\mathcal{I}|})$
  - Avaliar todas as combinações de itens
- Computação do suporte: $O(|\mathcal{I}| \cdot |\mathbf{D}|)$
  - Comparar cada itemset candidato com cada transação
- Total de tempo: $O(|\mathcal{I}| \cdot |\mathbf{D}| \cdot 2^{|\mathcal{I}|})$
- Total de I/O: $O(2^{|\mathcal{I}|})$

Busca em largura: algoritmo Apriori

Propriedades para reduzir espaço de busca

- Se $X \subseteq Y \subseteq \mathcal{I}$, então $sup(X) \geq sup(Y)$
  - $sup(\{A, B, C\}) \geq sup(\{A, B, C, D\})$
- Se Y é frequente, então $\forall X \subseteq Y$ são frequentes
- Se X não é frequente, então $\forall Y \supseteq X$ não são frequentes

- Exploração de árvore de prefixos em nível (busca em largura)
- Evita (poda) ramos de regras infrequentes
- Encontra todos k-itemsets usando árvore até altura k

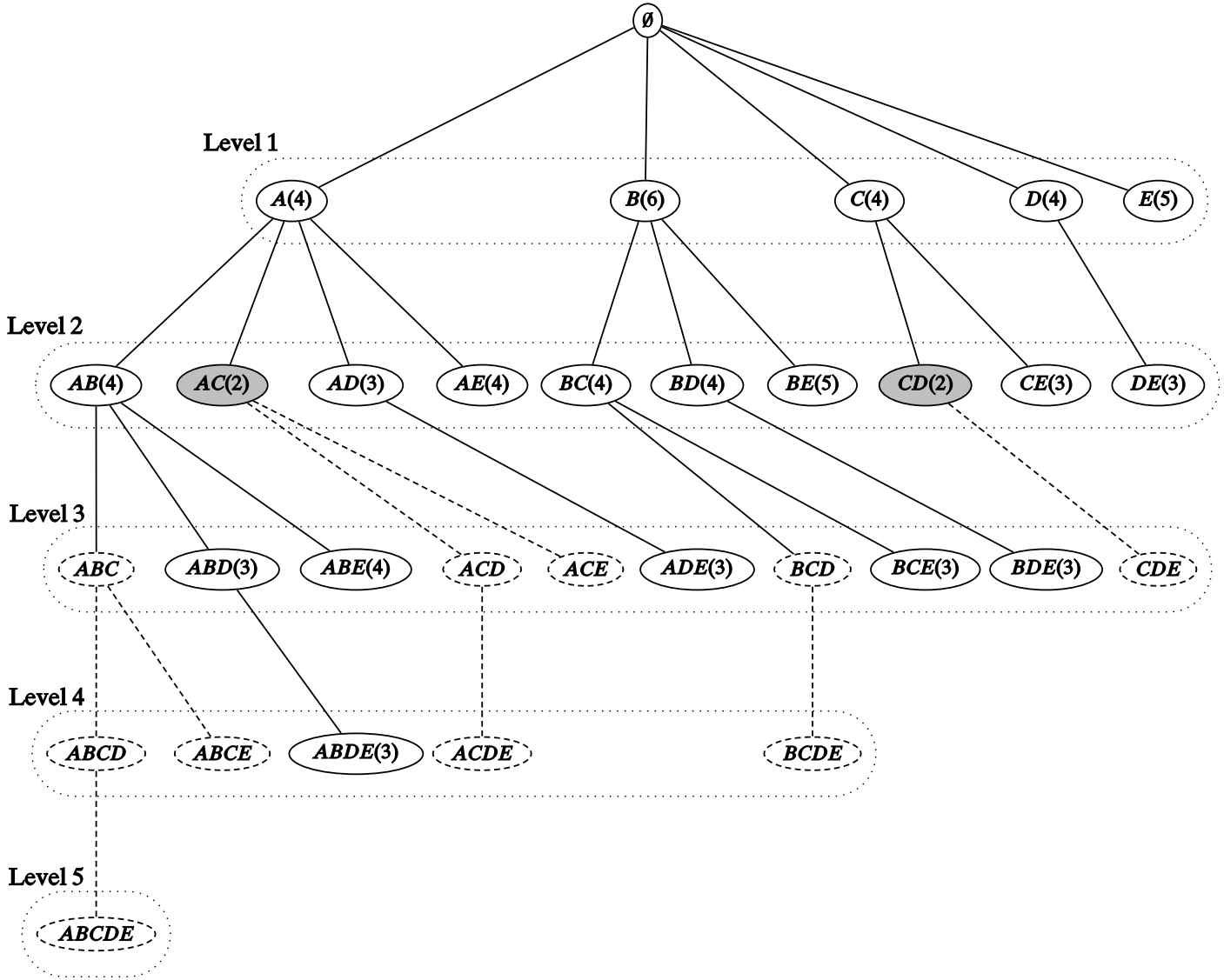


Figure 8.3. Apriori: prefix search tree and effect of pruning. Shaded nodes indicate infrequent itemsets, whereas dashed nodes and lines indicate all of the pruned nodes and branches. Solid lines indicate frequent itemsets.

Algoritmo Apriori

- Manutenção de uma árvore de prefixos para buscar subconjuntos mais frequentes
- Algoritmo em três partes
  - Algoritmo com árvore de prefixos para achar itemsets mais frequentes
  - Sub-algoritmo para computar o suporte de regras
  - Sub-algoritmo para extender a árvore de prefixos
    - Dois k-itemsets X_a e X_b com prefixo em comum de tamanho $k-1$ dão origem a um $k+1$-itemset $X_{ab} = X_a \cup X_b$
    - X_{ab} é mantido se nenhum subconjunto nele é infrequente
    - X_a e X_b são removidos

APRIORI $(\mathbf{D}, \mathcal{I}, minsup)$:

1 $\mathcal{F} \leftarrow \emptyset$ `//itemsets frequentes`

2 $\mathcal{C}^{(1)} \leftarrow \{\emptyset\}$ `//prefixo inicial da árvore`

3 **foreach** $i \in \mathcal{I}$ **do**
  Adiciona i como filho de $\emptyset$ em $\mathcal{C}$ com $sup(i) \leftarrow 0$

4 $k \leftarrow 1$ `//denota o nível atual`

5 **while** $\mathcal{C}^{(k)} \neq \emptyset$ **do**

6 COMPUTESUPPORT $(\mathcal{C}^{(k)}, \mathbf{D})$

7 **foreach** folha $X \in \mathcal{C}_{(k)}$ **do**

8 **if** $sup(X) \geq minsup$ **then**
  $\mathcal{F} \leftarrow \mathcal{F} \cup \{(X, sup(X))\}$

9 **else** remove X de $\mathcal{C}^{(k)}$

10 $\mathcal{C}^{(k+1)} \leftarrow$ EXTENDPREFIXTREE $(\mathcal{C}^{(k)})$

11 $k \leftarrow k + 1$

12 **return** $\mathcal{F}^{(k)}$

COMPUTESUPPORT $(\mathcal{C}^{(k)}, \mathbf{D})$:

13 **foreach** $\langle t, \mathbf{i}(t) \rangle \in \mathbf{D}$ **do**

14 **foreach** *k-subset* $X \subseteq \mathbf{i}(t)$ **do**

15 **if** $X \in \mathcal{C}^{(k)}$ **then** $sup(X) \leftarrow sup(X) + 1$

- Objetivo: Computar o suporte de cada k-itemset candidato em $C^{(k)}$
- Como: obter todos os k-itemsets possíveis para cada itemset na base $\mathbf{D}$ e contar repetições/suporte de cada k-itemset em $C^{(k)}$

EXTENDPREFIXTREE ($C^{(k)}$):

16 **foreach** folha $X_a \in C^{(k)}$ **do**

17 **foreach** folha X_b irmã de X_a tal que $b > a$ **do**

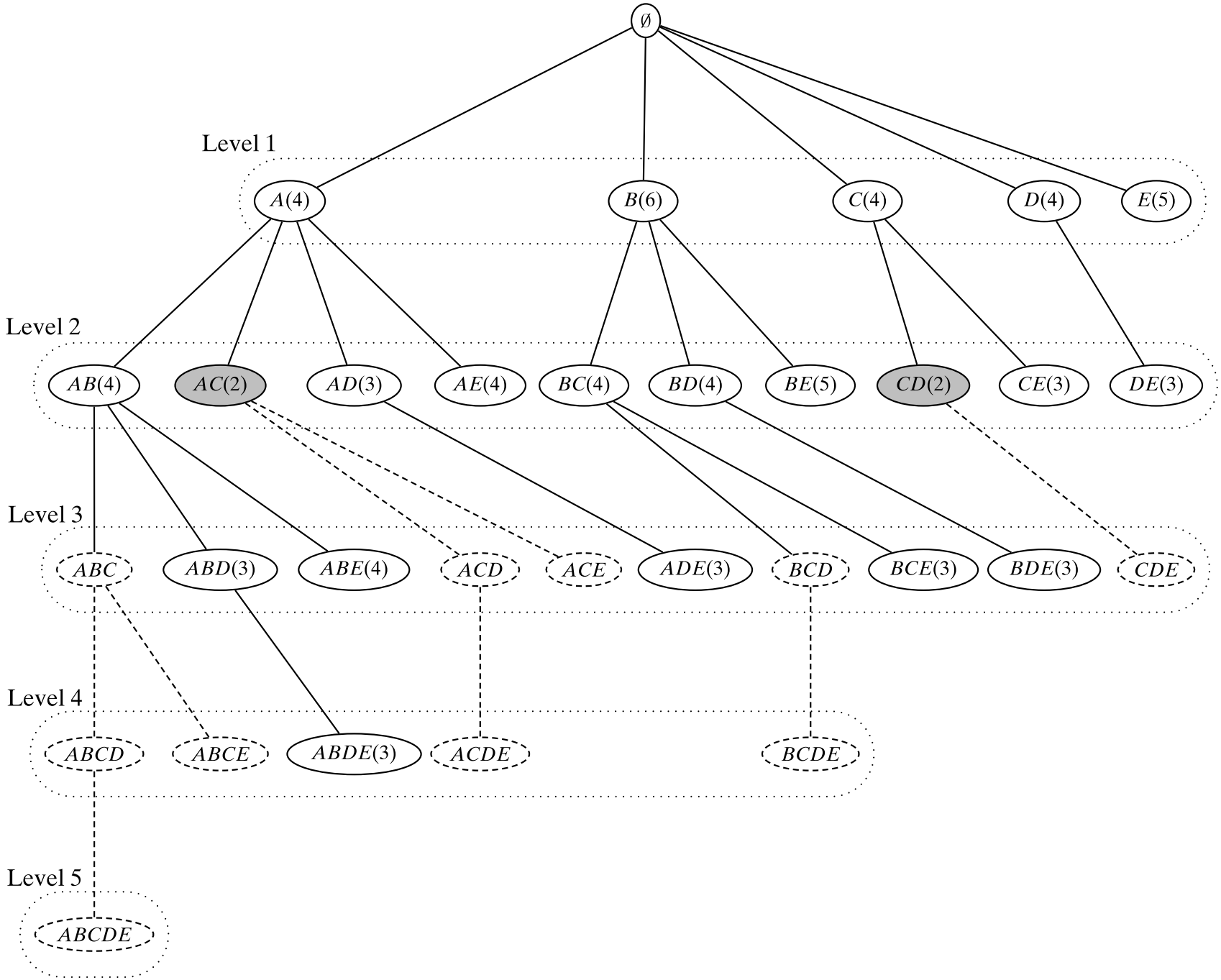
18 $X_{ab} \leftarrow X_a \cup X_b$

// podar candidato se tiver subconjunto infrequente

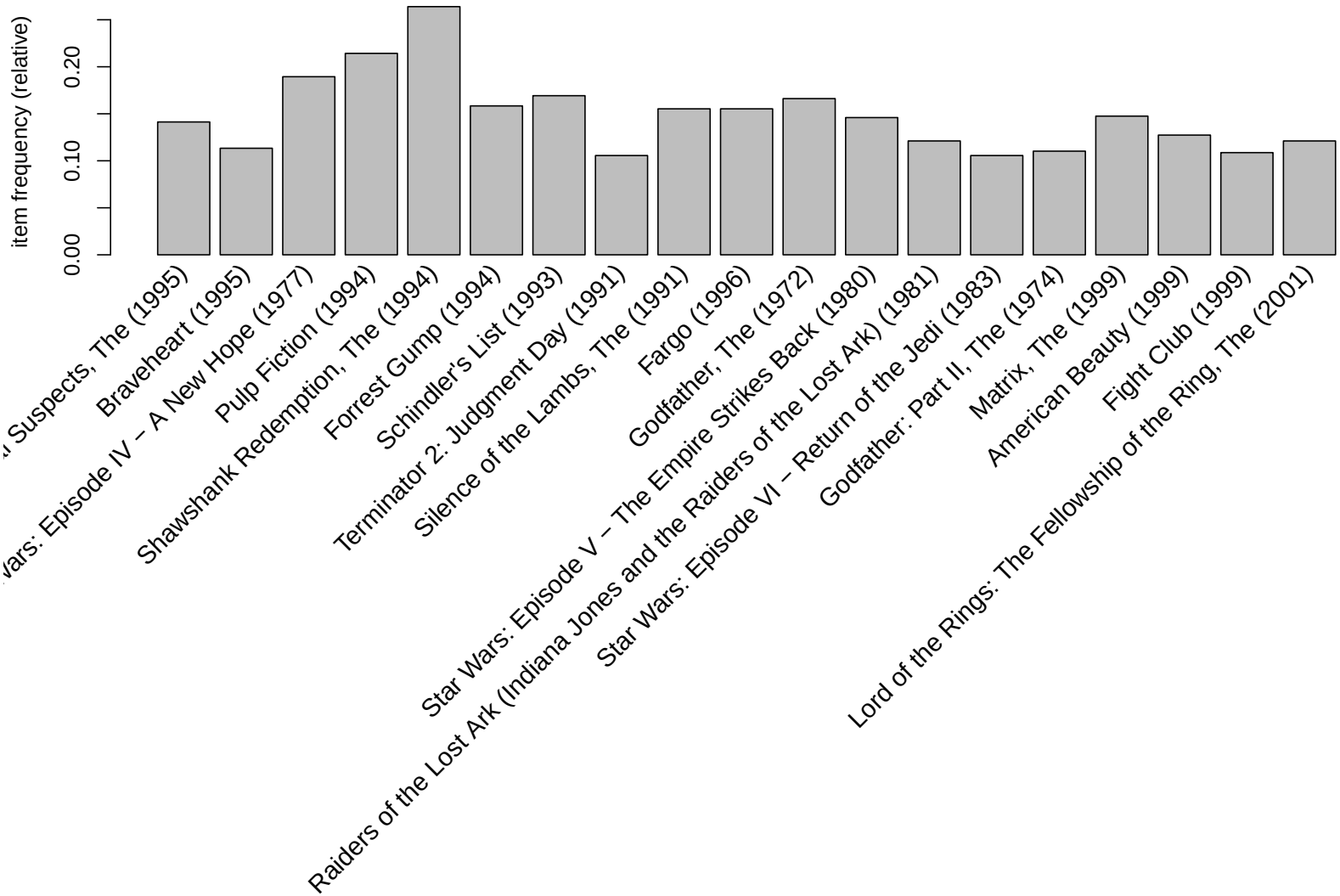
19 **if** $X_j \in C^{(k)}$, ***for all*** $X_j \subset X_{ab}$, **tal que** $|X_j| = |X_{ab}| - 1$ **then**

20 Adiciona X_{ab} como filho de X_a com $sup(X_{ab}) \leftarrow 0$

21 **return** $C^{(k)}$


∅
Level 1
A(4)
B(6)
C(4)
D(4)
E(5)
Level 2
AB(4)
AC(2)
AD(3)
AE(4)
BC(4)
BD(4)
BE(5)
CD(2)
CE(3)
DE(3)
Level 3
ABC
ABD(3)
ABE(4)
ACD
ACE
ADE(3)
BCD
BCE(3)
BDE(3)
CDE
Level 4
ABCD
ABCE
ABDE(3)
ACDE
BCDE
Level 5
ABCDE

```
itemFrequencyPlot(transacoes5,support=0.1,cex.names=1.2)
```



Exemplo MovieLens.org

```
regras <-apriori(transacoes5,
          par=list(supp=0.10, conf=.2, target="rules"),
          control=list(verbose=FALSE))
```

```
summary(regras)

## set of 4 rules
##
## rule length distribution (lhs + rhs):sizes
## 1 2
## 2 2
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     1.0     1.5     1.5     2.0     2.0
##
## summary of quality measures:
##     support         confidence          lift
##  Min.   :0.1071   Min.   :0.2143   Min.   :1.000
##  1st Qu.:0.1071   1st Qu.:0.2516   1st Qu.:1.000
##  Median :0.1607   Median :0.4148   Median :2.437
##  Mean   :0.1731   Mean   :0.4445   Mean   :2.437
##  3rd Qu.:0.2267   3rd Qu.:0.6077   3rd Qu.:3.875
##  Max.   :0.2640   Max.   :0.7340   Max.   :3.875
```

```
inspect(regras)
```

```
##     lhs
## [1] {}
## [2] {}
## [3] {Star Wars: Episode V - The Empire Strikes Back (198
## [4] {Star Wars: Episode IV - A New Hope (1977)}
```

Custo algoritmo Apriori

- Custo computacional é $O(|\mathcal{I}| \cdot |\mathbf{D}| \cdot 2^{|\mathcal{I}|})$
  - Pois todos itemsets podem ser frequentes
- Com a poda, custo computacional costuma ser mais baixo
- Custo de I/O do Apriori é $O(|\mathcal{I}|)$ e força-bruta é $O(2^{|\mathcal{I}|})$
- Custo de I/O é dependente do comprimento da sequência mais longa na base de dados

Exemplo MovieLens.org: custo Apriori

```
titles <- filter(movies, movieId %in% ratings$movieId)
tableRatings <- table(select(ratings,userId, movieId))
colnames(tableRatings) <- as.character(titles$title)
transacoes <- as(tableRatings > 0, "transactions")

# demora muito e não consegue completar
# time(apriori(transacoes,par=list(sup=0.05, conf=0.9)))
```

Melhoria do algoritmo Apriori: Algoritmo Eclat

- Problema do Apriori:
  - Cada nível, para computar suporte, na linha 19, geramos subconjunto de itens de cada transação para verificar existência na árvore
  - Mas muitos subconjuntos não existem na árvore
- Objetivo do Eclat: melhoria do custo de calcular o suporte de regras

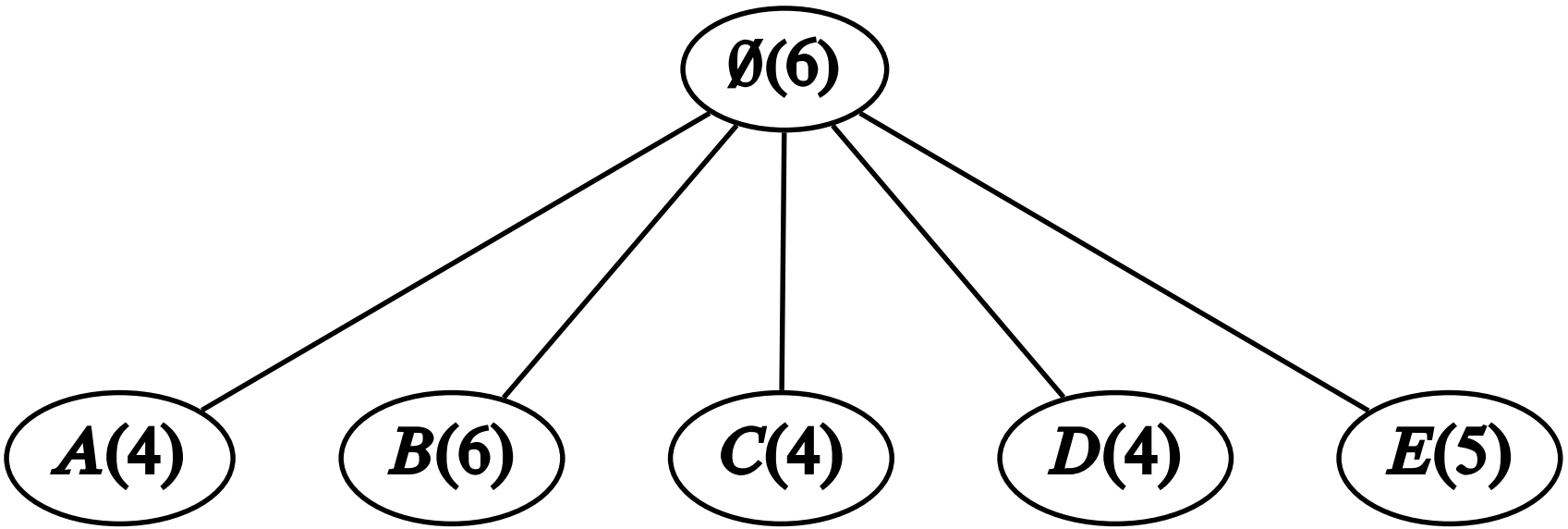
Algoritmo Eclat

- tidset $t(A)$ = conjunto de ids de transações contendo o item A
- Para os tidsets $t(X)$ e $t(Y)$, temos
  $t(X \cup Y) = t(XY) = t(X) \cap t(Y)$
- O suporte de XY é a cardinalidade de $t(XY)$:
  $sup(XY) = |t(XY)|$
- Exemplo:
  - Se $t(A) = 2345$ e $t(B) = 2456$ então
    $sup(XY) = |t(A) \cap t(B)| = |245| = 3$
- Útil na busca em profundidade na árvore de prefixos
- P é classe de equivalência de prefixos
  - Exemplo: $P_A = \{AB, AC, AD, AE\}$

```
// Chamada inicial F ← ∅, P ← {⟨i, t(i)⟩ | i ∈ I, |t(i)| ≥ minsup}
ECLAT (P, minsup, F):
1 foreach ⟨X_a, t(X_a)⟩ ∈ P do
2     F ← F ∪ {(X_a, sup(X_a))} // adiciona nível atual aos frequentes
3     P_a ← ∅ // guarda novos candidatos
4     foreach ⟨X_b, t(X_b)⟩ ∈ P, with X_b > X_a do
5         X_ab = X_a ∪ X_b
6         t(X_ab) = t(X_a) ∩ t(X_b) // tidset de Xab é intersecção dos de Xa Xb
7         if sup(X_ab) ≥ minsup then //sup(XY) = |t(XY)|
8             P_a ← P_a ∪ {⟨X_ab, t(X_ab)⟩}
9     if P_a ≠ ∅ then ECLAT (P_a, minsup, F)
```

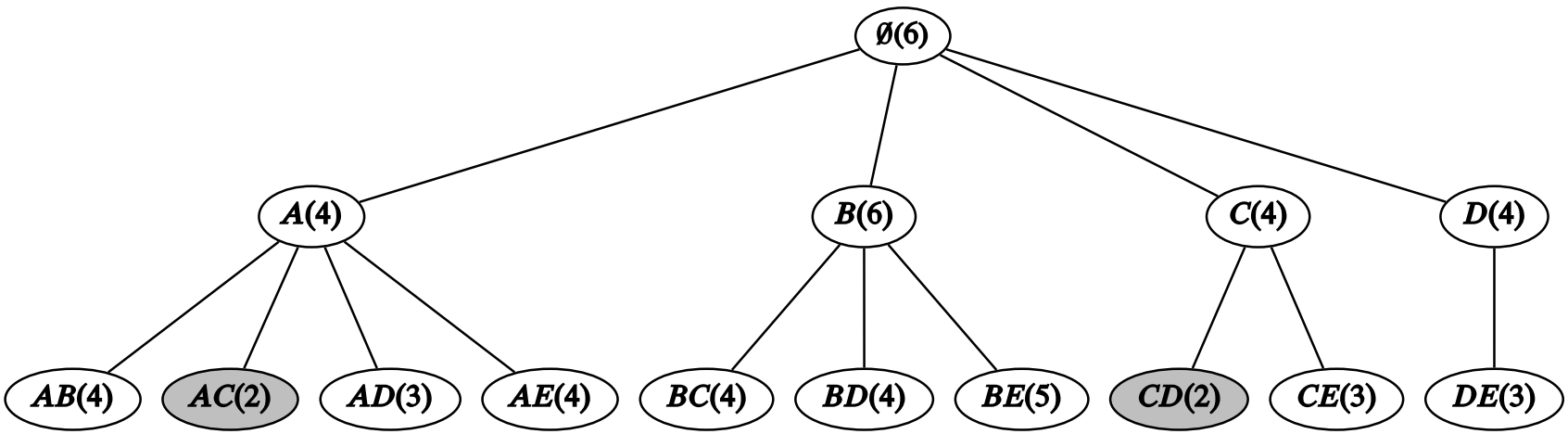
- Inicia recursão com os itens com suporte mínimo

Exemplo Eclat



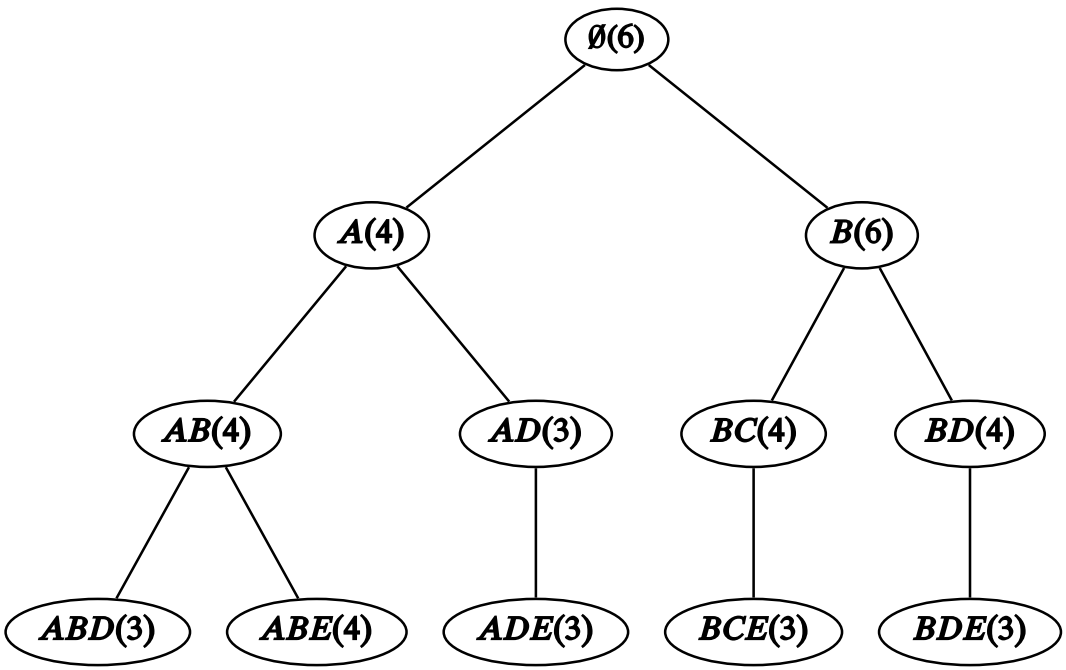
(a) $C^{(1)}$

Exemplo Eclat

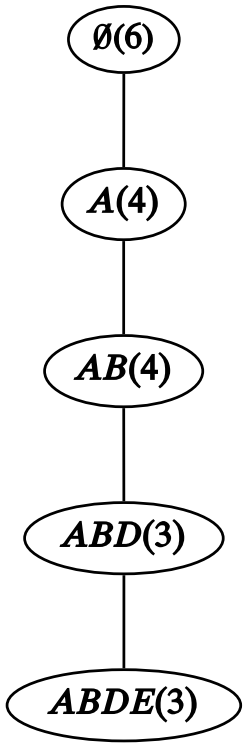


(b) $C^{(2)}$

Exemplo Eclat



(c) $C^{(3)}$



(d) $C^{(4)}$

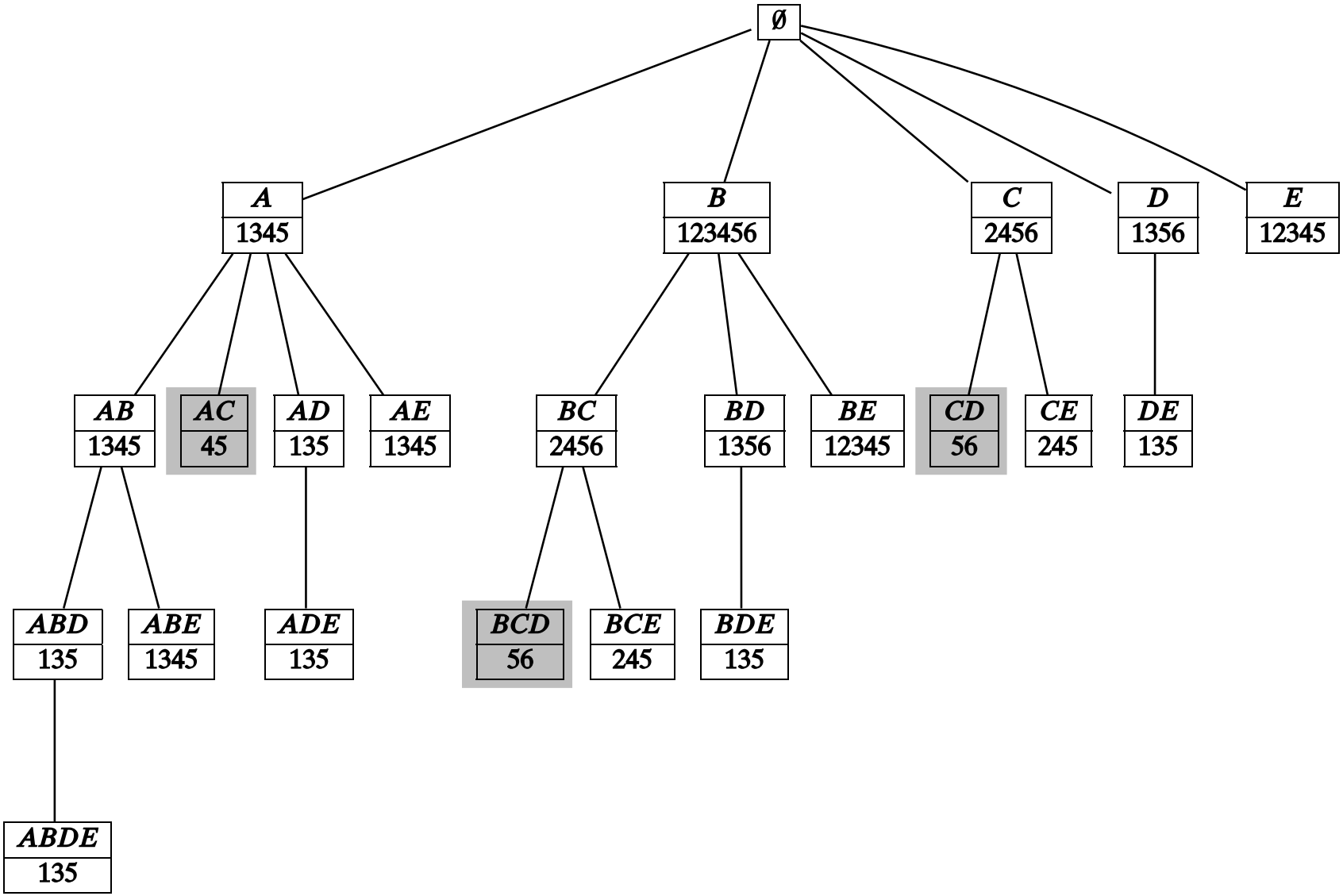


Figure 8.5. Eclat algorithm: tidlist intersections (gray boxes indicate infrequent itemsets).

Eclat: custos

- Complexidade computacional $O(|\mathbf{D}| \cdot 2^{|\mathcal{I}|})$
  - Podem haver $2^{|\mathcal{I}|}$ itemsets frequentes
  - Intersecção entre tidset é $O(\mathbf{D})$
  -
- I/O: $O(2^{|\mathcal{I}|}/|\mathcal{I}|)$, mas requer até l varreduras na base onde l é o comprimento do maior itemset

Exemplo MovieLens.org: custo Eclat

```
titles <- filter(movies, movieId %in% ratings$movieId)
tableRatings <- table(select(ratings,userId, movieId))
colnames(tableRatings) <- as.character(titles$title)
transacoes <- as(tableRatings > 0, "transactions")

# muito melhor que apriori:
#time(ruleInduction(
#      eclat(transacoes,par=list(sup=0.05))
#      ,conf=0.9))
```

Melhoria da computação de suporte: FP-Tree e FPGrowth

- Computação de Suporte com indexação com a *frequent pattern tree*: FP-Tree
- FP-tree
  - Cada nó representa um item e suporte da raiz até o próprio nó

Construção de uma FP-tree

- Início: raiz contém item vazio $\emptyset$
- Inserção na árvore com itens em ordem decrescente pelo suporte
  - Em vez de ABED, inserir na ordem BEAD (pois sup(6) ¿ sup(E))
- Para cada tupla $(t, X) \in \mathbf{D}$ onde $X = \mathbf{i}(tid)$, inserir X na árvore incrementando contagem dos nós no caminho de X
- Se X compartilha prefixo com outro itemset, X vai ser mesmo caminho até formar o prefixo.
- Contadores iniciam com 1
- Após construção da FP-tree, ela é usada como índice no lugar de **D**

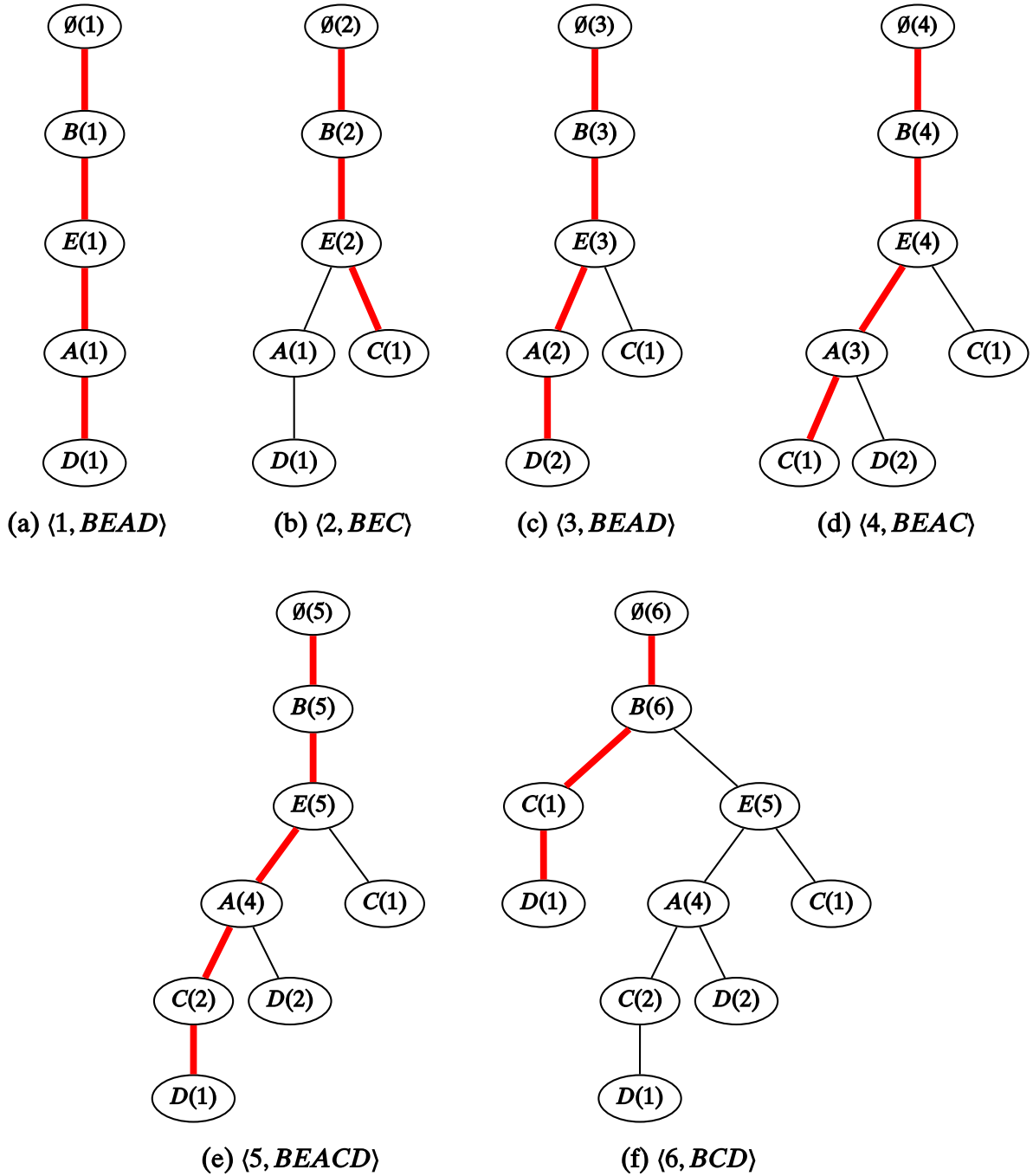


Figure 8.7. Frequent pattern tree: bold edges indicate current transaction.

```
//Chamada inicial: R ← FP-tree(D), P ← ∅, F ← ∅
FPGrowth (R, P, F, minsup):
1  Remove itens infrequentes de R
2  if IsPath(R) then //insere subsets de R em F
3      foreach Y ⊆ R do
4          X ← P ∪ Y
5          sup(X) ← min_{x∈Y}{cnt(x)}
6          F ← F ∪ {(X, sup(X))}
7  else // processa FP-tree projetada p/ cada item frequente i
8      foreach i ∈ R em ordem crescente de sup(i) do
9          X ← P ∪ {i}
10         sup(X) ← sup(i) // soma de cnt(i) para todos nós com "i"
11         F ← F ∪ {(X, sup(X))}
12         R_X ← ∅ //FP-tree projetada para X
13         foreach path ∈ PathFromRoot(i) do
14             cnt(i) ← contador de i em path
15             Insere path, excluindo i, em FP-tree R_X com contador cnt(i)
16         if R_X ≠ ∅ then FPGrowth (R_X, X, F, minsup)
```

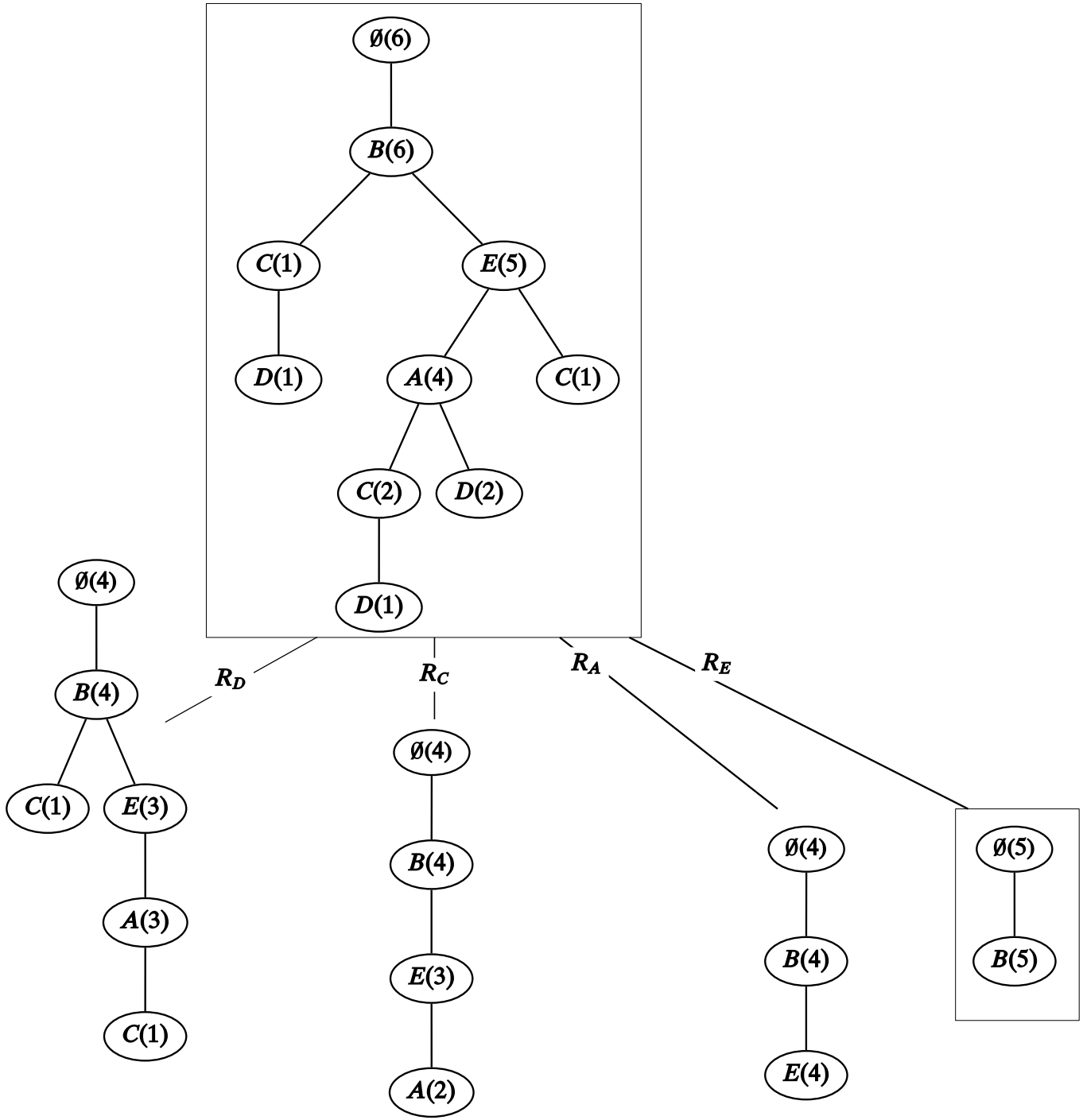


Figure 8.9. FPGrowth algorithm: frequent pattern tree projection.

Geração de regras de associação

- Gerar itemsets frequentes é o primeiro passo para obter regras de associação
- Regra de associação

$$X \rightarrow Y$$

  onde XY deve ter suporte mínimo
- A regra é forte se tem confiança mínima:

$$c = \frac{sup(X \cup Y)}{sup(X)}$$

ALGORITHM 8.6. Algorithm AssociationRules

```
AssociationRules (F, minconf):
1  foreach Z ∈ F, such that |Z| ≥ 2 do
2      A ← {X | X ⊂ Z, X ≠ ∅}
3      while A ≠ ∅ do
4          X ← maximal element in A
5          A ← A \ X // remove X from A
6          c ← sup(Z)/sup(X)
7          if c ≥ minconf then
8              print X ⟶ Y, sup(Z), c
9          else
10             A ← A \ {W | W ⊂ X} // remove all subsets of X from A
```

- $\mathcal{A}$ são itemsets que formarão antecedentes de regras
- Elemento maximal tem o maior número de itens
- $Y = Z \backslash X$
- Linha 10: poda subconjuntos contendo antecedente com baixa confiança

Comparação entre algoritmos (Goethals e Zaki, 2004)

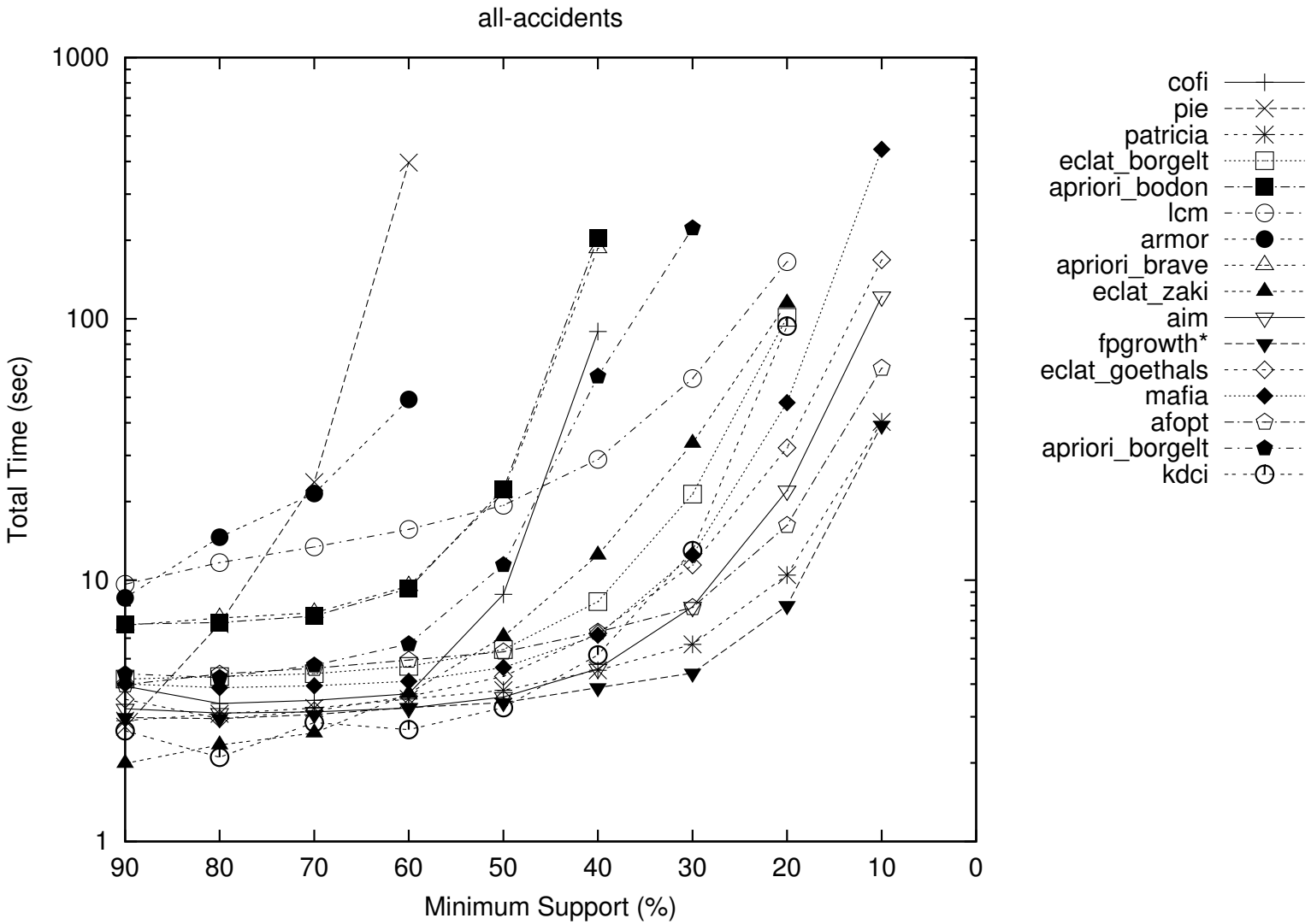


Figura: Accidents: $|\mathcal{I}| = 468$, $E(|\mathbf{t}(x)|) = 33.8$, $|\mathbf{t}(x)| = 340\,183$

Sumarização de Itemsets

- Espaço de itemsets é grande
- Baixo *min_sup* torna problema intratável
- Busca por representações para itemsets frequentes
  - Itemset maximal
  - Itemset fechado
  - Itemset não-derivável

Itemsets maximais

- Itemset X é maximal se não existe $Y \supset X$ tal que $sup(Y) \geq min\_sup$
- Exemplo: ABDE e BCE

Tid	Itemset
1	$ABDE$
2	BCE
3	$ABDE$
4	$ABCE$
5	$ABCDE$
6	BCD

(a) Transaction database

sup	Itemsets
6	B
5	E, BE
4	$A, C, D, AB, AE, BC, BD, ABE$
3	$AD, CE, DE, ABD, ADE, BCE, BDE, ABDE$

(b) Frequent itemsets ($minsup = 3$)

Figure 9.1. An example database.

Geradores minimais (mínimos)

- Todos subsets de X tem suporte maior que $sup(X)$.
- $\mathcal{G} = \{X | X \in \mathcal{F}$ e $\nexists Y \subset X$ tal que $sup(X) = sup(Y)\}$

GenMax

- Definição conjunto maximais:
  $\mathcal{M} = \{X | X \in \mathcal{F} \text{ e } \nexists Y \in \mathcal{F} \text{ tal que } Y \supset X\}$
- Construir $\mathcal{M}$ passo a passo
- Usar um algoritmo para encontrar itemset frequente X e verificar:
  - Um itemset X está contido em $\mathcal{M}$?
  - Um itemset $Y \in \mathcal{M}$ está contido em X ?

ALGORITHM 9.1. Algorithm GENMAX

```
// Initial Call: M ← ∅, P ← {⟨i, t(i)⟩ | i ∈ I, sup(i) ≥ minsup}
GENMAX (P, minsup, M):
1  Y ← ⋃ X_i
2  if ∃Z ∈ M, such that Y ⊆ Z then
3      return // prune entire branch
4  foreach ⟨X_i, t(X_i)⟩ ∈ P do
5      P_i ← ∅
6      foreach ⟨X_j, t(X_j)⟩ ∈ P, with j > i do
7          X_ij ← X_i ∪ X_j
8          t(X_ij) = t(X_i) ∩ t(X_j)
9          if sup(X_ij) ≥ minsup then P_i ← P_i ∪ {⟨X_ij, t(X_ij)⟩}
10     if P_i ≠ ∅ then GENMAX (P_i, minsup, M)
11     else if ∄Z ∈ M, X_i ⊆ Z then
12         M = M ∪ X_i    // add X_i to maximal set
```

- Linha 2: se verdade, ramo inteiro junto não é maximal
- Linha 10: se não existem mais candidatos com X_i ele pode ser maximal

Execução GenMax

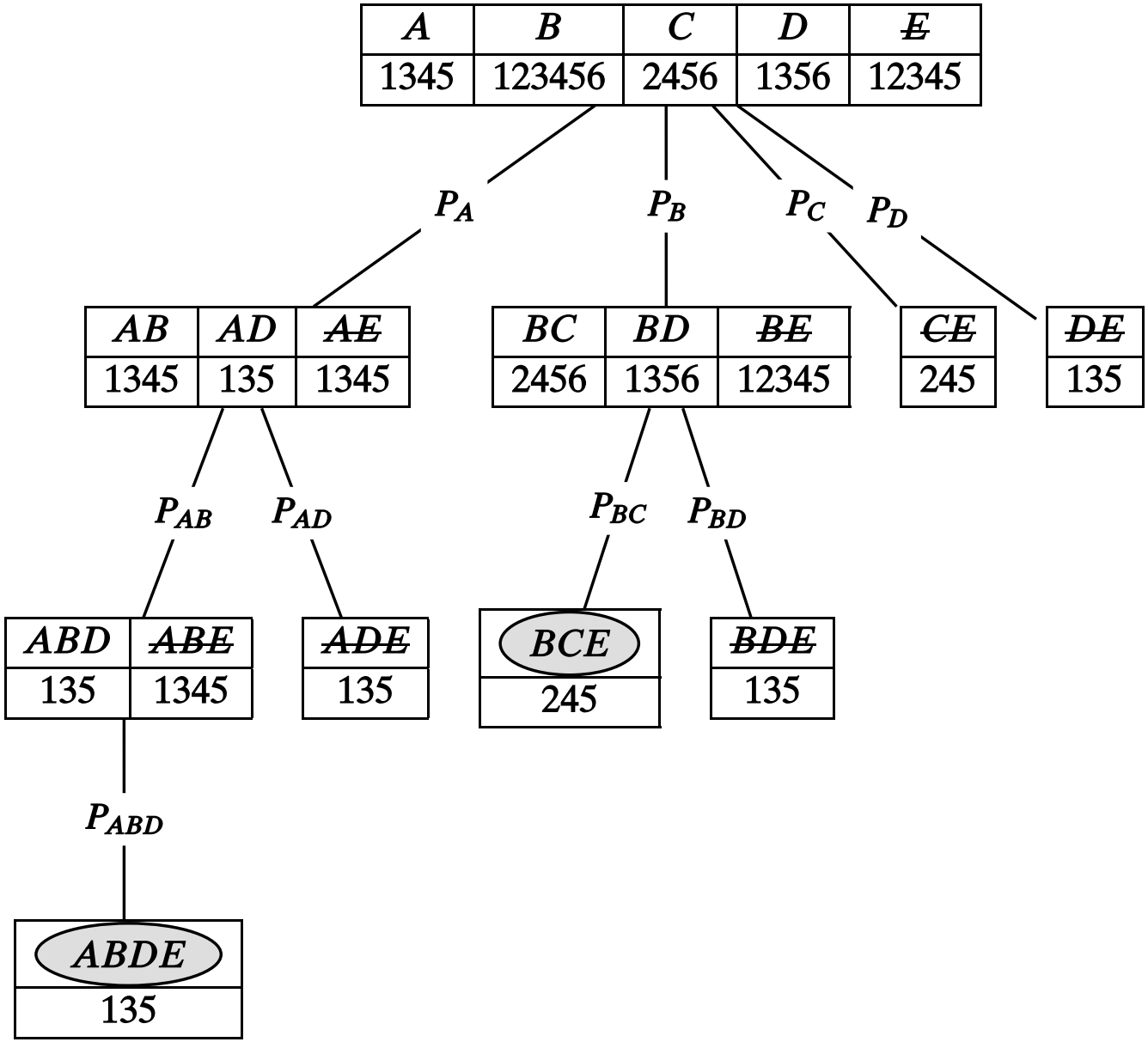
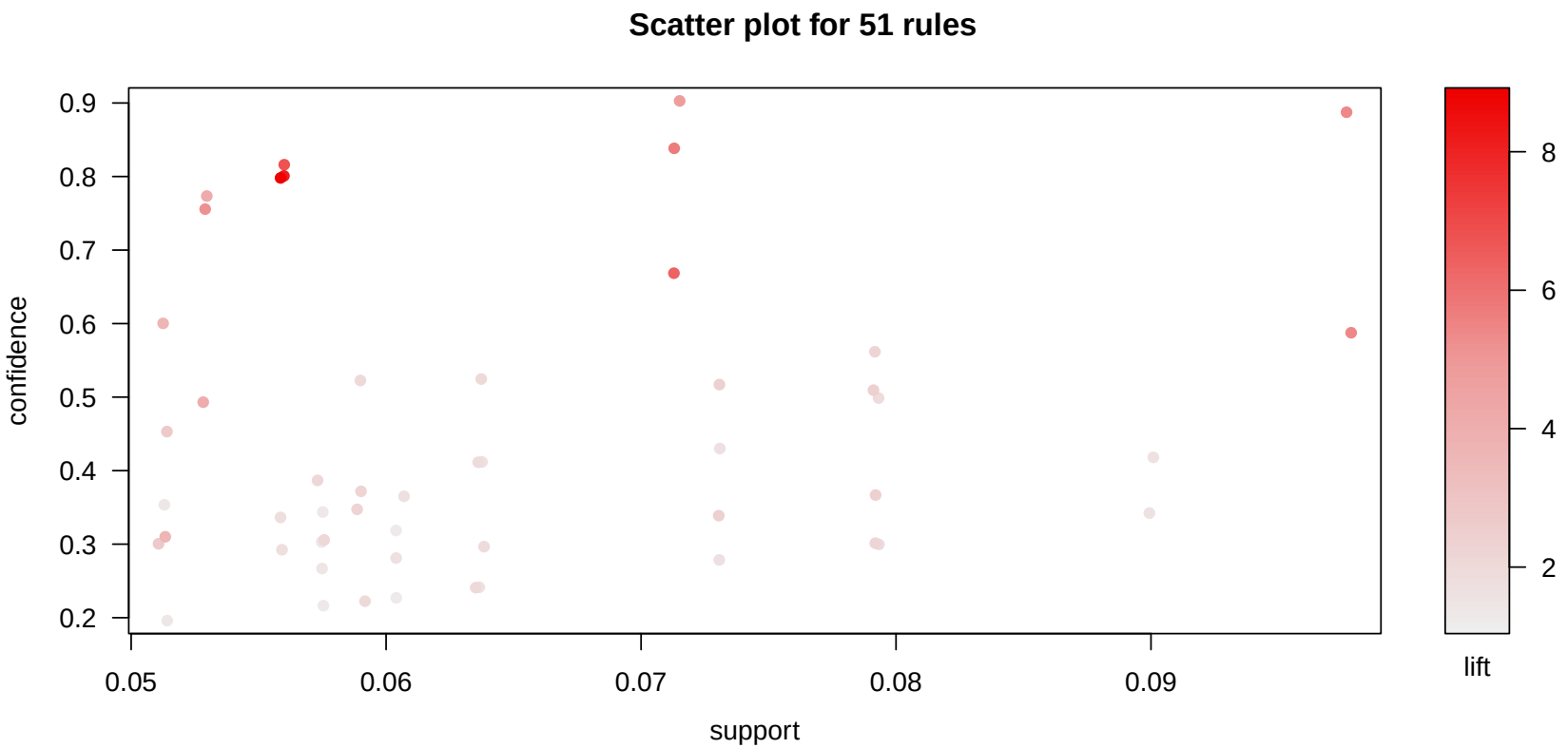


Figure 9.3. Mining maximal frequent itemsets. Maximal itemsets are shown as shaded ovals, whereas pruned

Itemsets Maximais

```
require(arulesViz)
maximais <- apriori(transacoes5,
                    par=list(target="max",sup=0.05),
                    control=list(verbose=FALSE))
plot(ruleInduction(maximais,transacoes5,conf=0.01))
```



Itemsets fechados

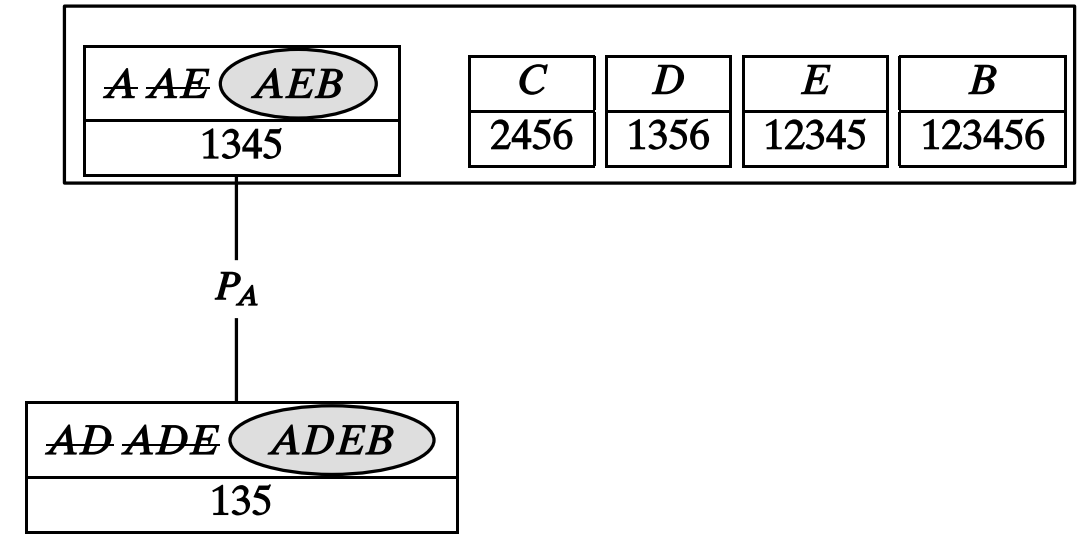
- Itemset X é fechado se não existe $Y \supset X$ tal que $sup(Y) = sup(X)$
  - $\mathbf{t}(X) = \{t \in \mathcal{T} | t \supset X\}$
  - $\mathbf{i}(id) = \{x \in \mathcal{I} | \forall tid \in T, t \supset x\}$
- Operador de fechamento $\mathbf{c} : 2^{\mathcal{I}} \rightarrow 2^{\mathcal{I}}$:
  - $\mathbf{c}(X) = \mathbf{i} \circ \mathbf{t}(X) = \mathbf{i}(\mathbf{t}(X))$
  - Propriedade "extensivo": $X \subset \mathbf{c}(X)$
  - Propriedade "monotônico": se $X_i \subseteq X_J$ então $\mathbf{c}(X_i) \subseteq \mathbf{c}(X_j)$
  - Propriedade "idempotente": $\mathbf{c}(\mathbf{c}(X)) = \mathbf{c}(X)$
- Itemset X é fechado se $\mathbf{c}(X) = X$
- Cálculo de suporte: $sup(X) = \max\{sup(Y) | X \subseteq Y = c(X)\}$

Mineração de Itemsets Fechados: Charm

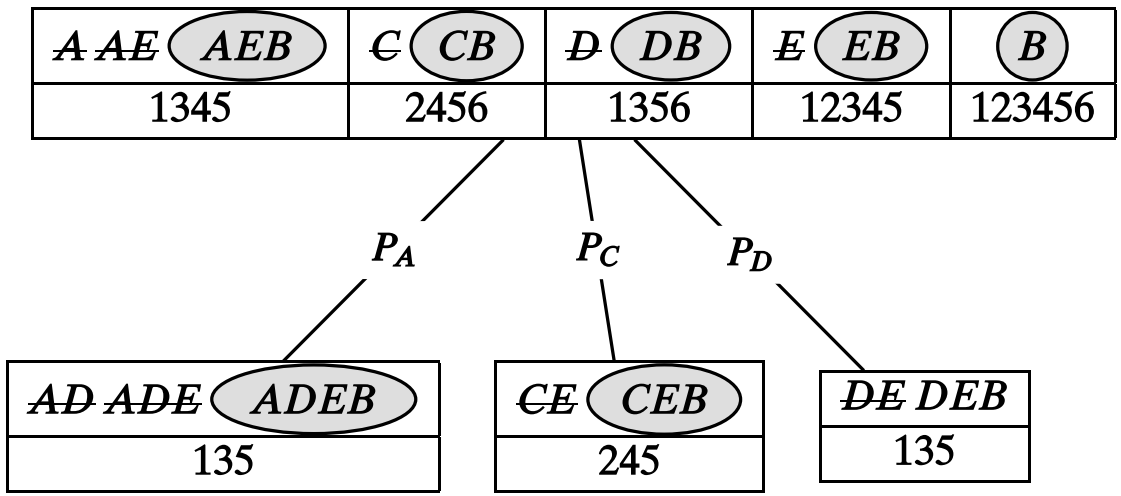
- Operador fechamento: $c(X) = i(t(X))$
- $t(X_i) = t(X_j) \Rightarrow c(X_i) = c(X_j) = c(X_i \cup X_j)$
  - Substituir X_i por $X_i \cup X_j$
  - Podar X_j pois fará itemset fechado idêntico ao de X_i
- $t(X_i) \subset t(X_j) \Rightarrow c(X_i) = c(X_i \cup X_j) \neq c(X_j)$
  - Substituir X_i por $X_i \cup X_j$
  - Não podar para X_j pois fechamento será diferente
- $t(X_i) \not\subset t(X_j)$
  - Sem podas

ALGORITHM 9.2. Algorithm CHARM

```
// Initial Call: C ← ∅, P ← {⟨i, t(i)⟩ : i ∈ I, sup(i) ≥ minsup}
CHARM (P, minsup, C):
1  Sort P in increasing order of support (i.e., by increasing |t(X_i)|)
2  foreach ⟨X_i, t(X_i)⟩ ∈ P do
3      P_i ← ∅
4      foreach ⟨X_j, t(X_j)⟩ ∈ P, with j > i do
5          X_ij = X_i ∪ X_j
6          t(X_ij) = t(X_i) ∩ t(X_j)
7          if sup(X_ij) ≥ minsup then
8              if t(X_i) = t(X_j) then // Property 1
9                  Replace X_i with X_ij in P and P_i
10                 Remove ⟨X_j, t(X_j)⟩ from P
11             else
12                 if t(X_i) ⊂ t(X_j) then // Property 2
13                     Replace X_i with X_ij in P and P_i
14                 else // Property 3
15                     P_i ← P_i ∪ {⟨X_ij, t(X_ij)⟩}
16     if P_i ≠ ∅ then CHARM (P_i, minsup, C)
17     if ∄Z ∈ C, such that X_i ⊆ Z and t(X_i) = t(Z) then
18         C = C ∪ X_i   // Add X_i to closed set
```



(a) Process A



(b) Charm

Figure 9.4. Mining closed frequent itemsets. Closed itemsets are shown as shaded ovals. Strike-through represents itemsets X_i replaced by $X_i \cup X_j$ during execution of the algorithm. Infrequent itemsets are not shown.

Itemsets fechados

```
data("Adult")
closed <- apriori(Adult, # busca itemsets fechadas
   parameter = list(target = "closed", support = 0.4),
   control = list(verbose=FALSE))
regras_closed <- ruleInduction(closed, Adult)
summary(regras_closed)

## set of 165 rules
##
## rule length distribution (lhs + rhs):sizes
##  2  3  4  5
## 40 74 43  8
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   3.000   3.000   3.115   4.000   5.000
##
## summary of quality measures:
```

Itemsets fechados

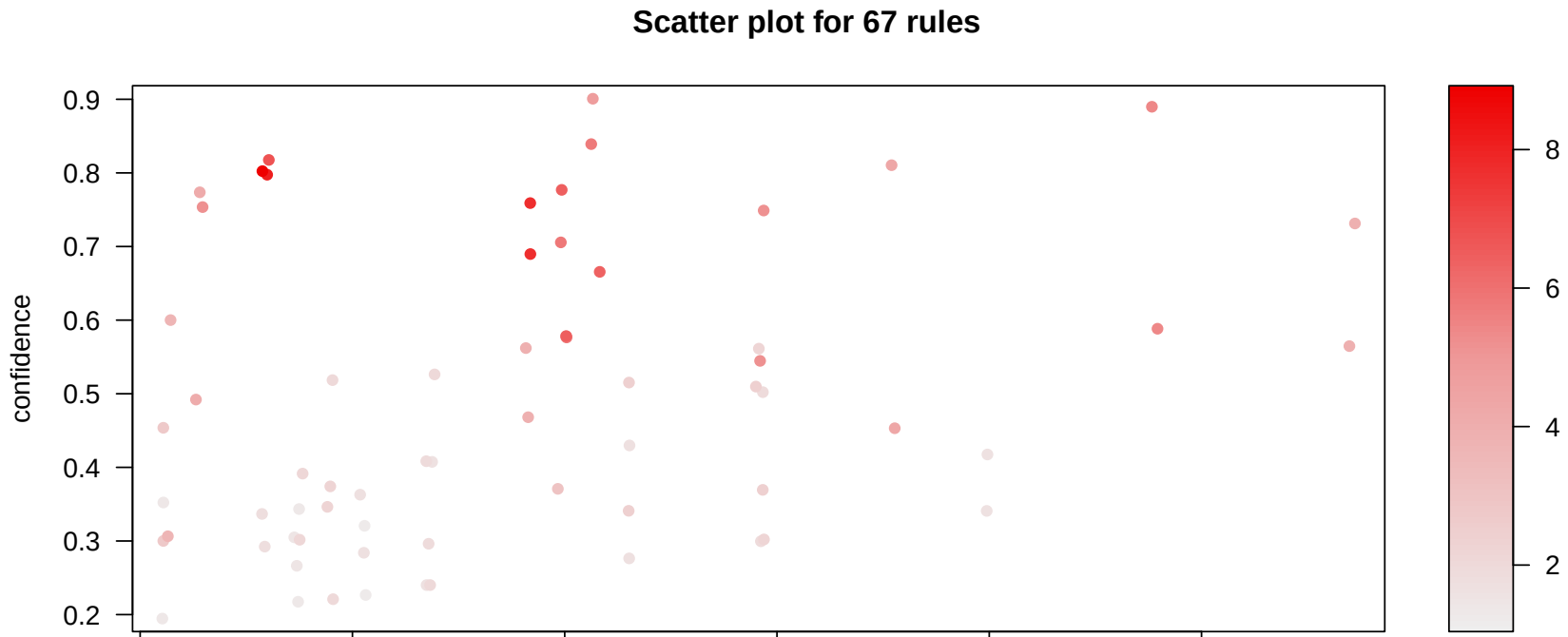
```
summary(regras_closed)

## set of 165 rules
##
## rule length distribution (lhs + rhs):sizes
##  2  3  4  5
## 40 74 43  8
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   3.000   3.000   3.115   4.000   5.000
##
## summary of quality measures:
##     support         confidence          lift
##  Min.   :0.4013   Min.   :0.8209   Min.   :0.9594
##  1st Qu.:0.4341   1st Qu.:0.8871   1st Qu.:0.9912
##  Median :0.4994   Median :0.9129   Median :0.9984
##  Mean   :0.5333   Mean   :0.9121   Mean   :1.0409
##  3rd Qu.:0.5697   3rd Qu.:0.9471   3rd Qu.:1.0169
##  Max.   :0.8707   Max.   :0.9999   Max.   :2.4529
##     itemset
##  Min.   :12.00
##  1st Qu.:47.00
##  Median :70.00
##  Mean   :65.33
##  3rd Qu.:87.00
##  Max.   :99.00
##
## mining info:
##   data ntransactions support confidence
##  Adult         48842     0.4        0.8
```

Itemsets fechados

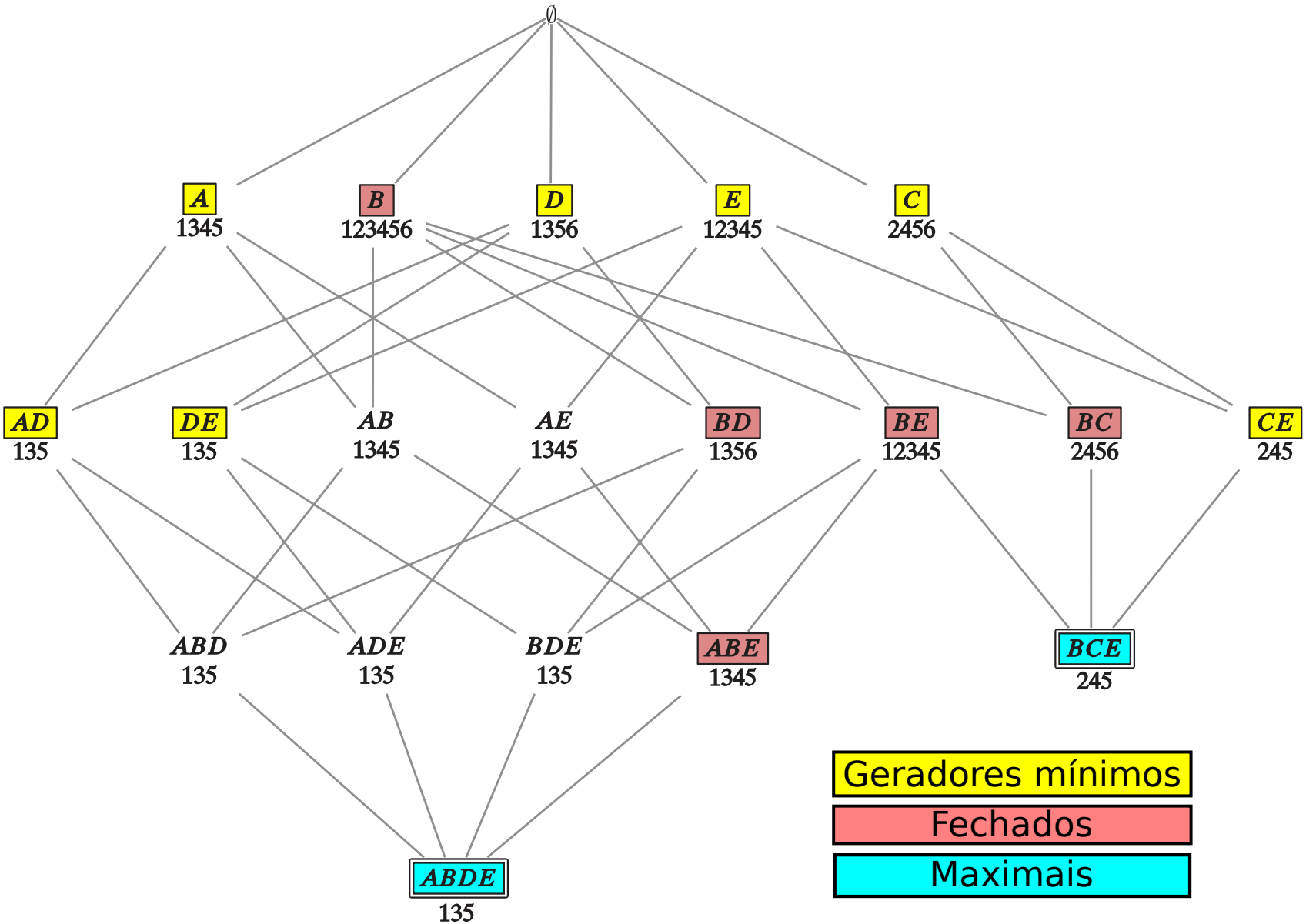
```
inspect(regras_closed[1]@lhs)

##     items
## [1] {relationship=Husband}

inspect(regras_closed[1]@rhs)

##     items
## [1] {marital-status=Married-civ-spouse}

print(regras_closed[1]@quality)

##     support confidence     lift itemset
## 1 0.4034233  0.9993914 2.181164      12
```

Itemsets Fechados

```
fechados <- apriori(transacoes5,
                    par=list(target="closed",sup=0.05),
                    control=list(verbose=FALSE))
plot(ruleInduction(fechados,transacoes5,conf=0.01))

## To reduce overplotting, jitter is added!  Use
jitter = 0 to prevent jitter.
```



Relação entre tipos de itemsets



Outras medidas de avaliação de itemsets e regras

- Às vezes, há muitos itemsets com suporte e confiança altos
- Opção: $lift(X \rightarrow Y) = \frac{rsup(X \cup Y)}{rsup(X)rsup(Y)}$
  - Mede dependência entre X e Y: quanto maior, maior dependência, maior interesse
  - Desvio do suporte da regra inteira em relação ao suporte de cada parte

Ordenação por lift

```
closed <- eclat(transacoes5,
                par=list(target="closed",
                         sup=0.01),
                control=list(verbose=F))
rClosed <- ruleInduction(closed, transacoes5, conf=0.01)
as(rClosed,"data.frame") %>% arrange(lift) %>% head()

##
## 1
## 2
## 3 {Star Wars: Episode IV - A New Hope (1977),Star Wars:
## 4                                   {Lord of the Rings: Th
## 5                                   {Silence of the Lambs
## 6
##      support confidence      lift itemset
## 1 0.01086957 0.18918919 0.7166932    5348
## 2 0.01086957 0.04117647 0.7166932    5348
```

Ordenação por tamanho da regra

```
as(rClosed,"data.frame") %>%
    arrange(desc(size(fechados)[itemset])) %>% head(2)

##
## 1 {Office Space (1999)} => {Shawshank Redemption, The (1
## 2 {Shawshank Redemption, The (1994)} => {Office Space (1
##      support confidence     lift itemset
## 1 0.01242236 0.40000000 1.515294      97
## 2 0.01242236 0.04705882 1.515294      97
```

Outra medida de qualidade: leverage

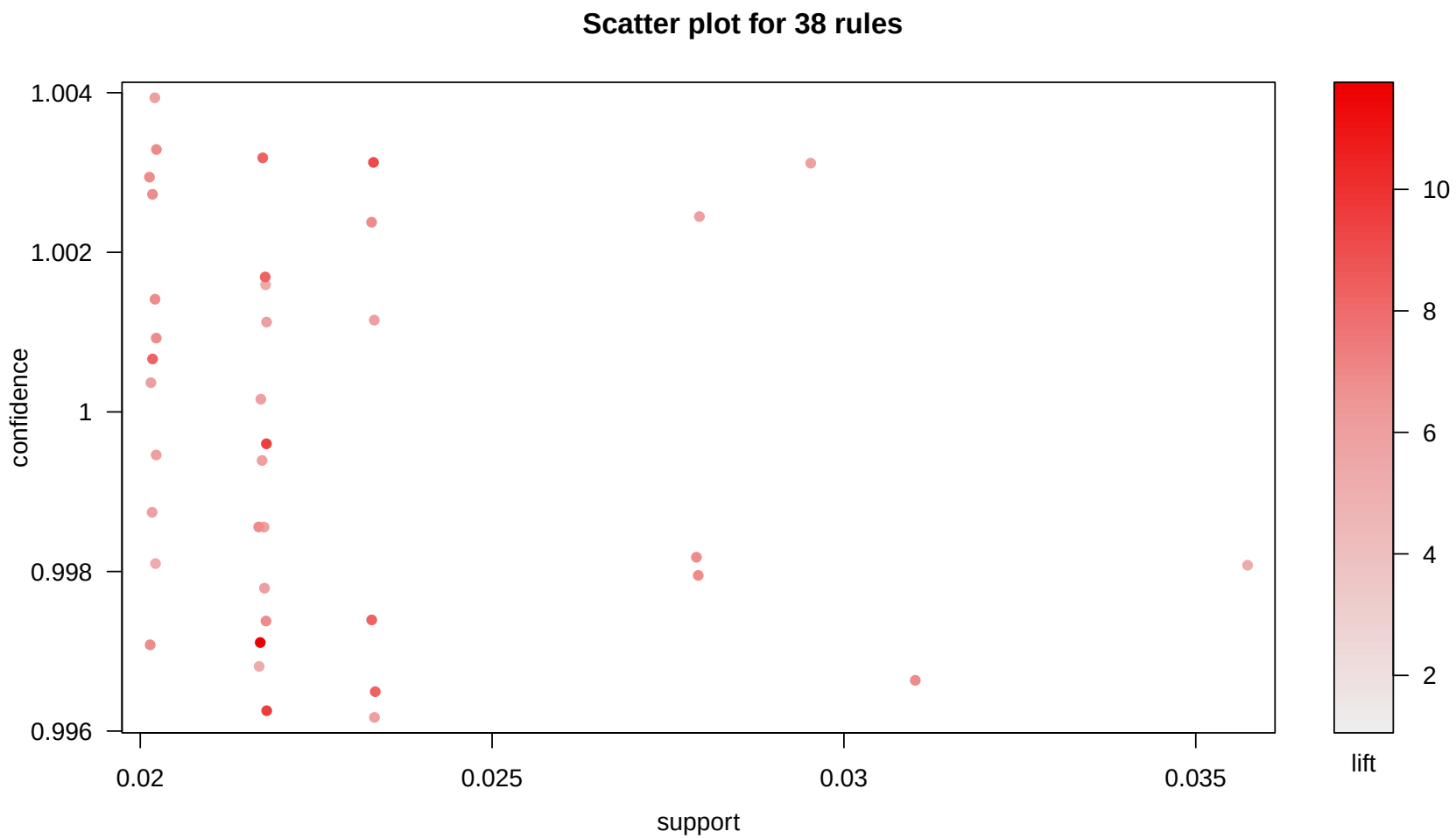
- Medida de surpresa de uma regra
- $leverage(X \rightarrow Y) = P(XY) - P(X) \cdot P(Y) =$
  $rsup(XY) - rsup(X) \cdot rsup(Y)$

```
pXY <- support(rClosed,transacoes5)
pX  <- support(lhs(rClosed), transacoes5)
pY  <- support(rhs(rClosed), transacoes5)
leverage <- pXY - pX * pY
as(rClosed,"data.frame") %>%
   arrange(desc(leverage[itemset])) %>% head(5)

##
## 1           {Saving Private Ryan (1998),Matrix, The (199
## 2           {Shawshank Redemption, The (1994),Matrix, The
## 3           {Shawshank Redemption, The (1994),Saving Priv
## 4 {Saving Private Ryan (1998),Matrix, The (1999)} => {St
## 5 {Star Wars: Episode IV - A New Hope (1977),Matrix, The
##      support confidence     lift itemset
```

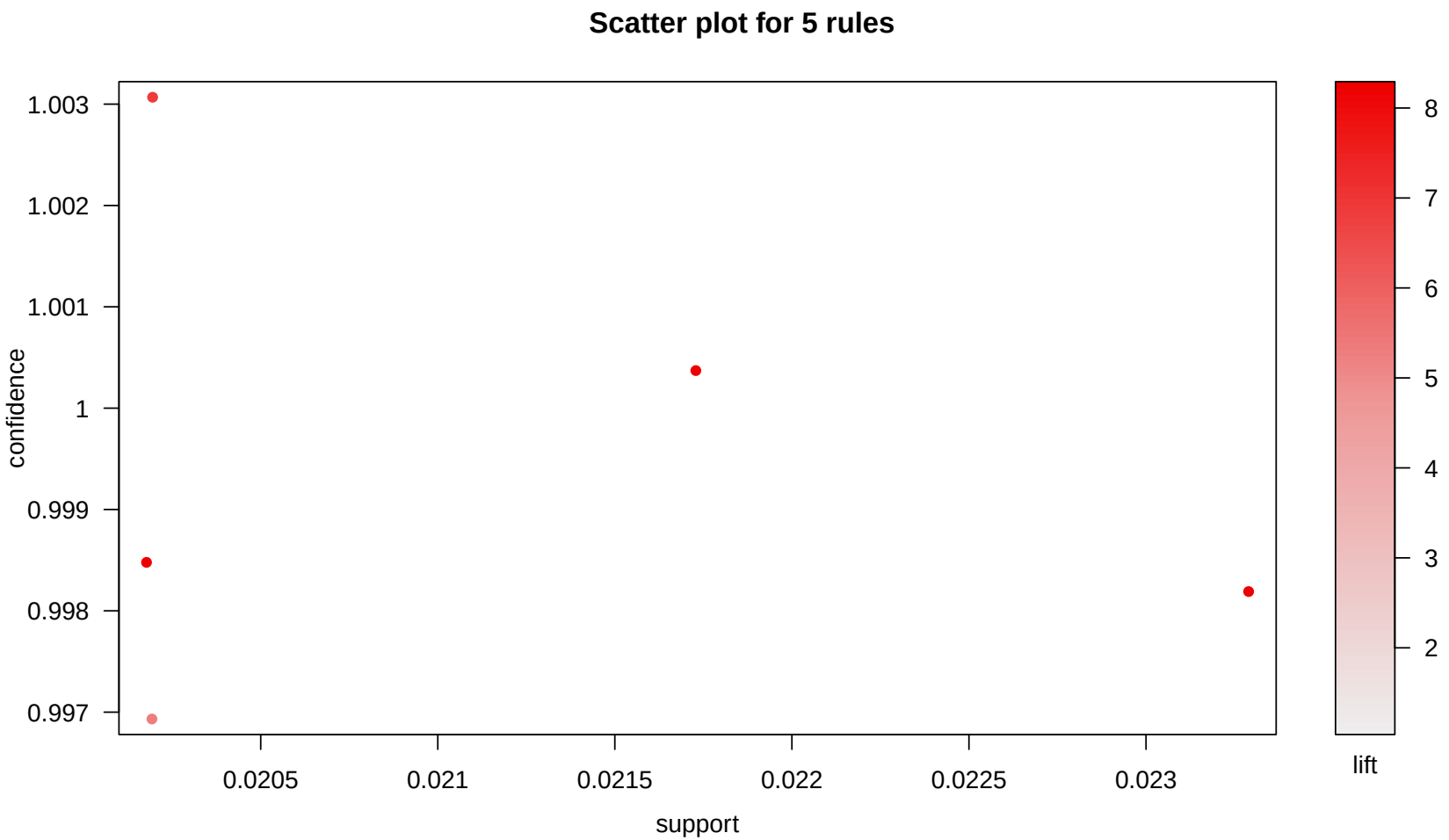
```
itemsetsFreq = eclat(transacoes5,
                     par=list(support=0.01),
                     control=list(verbose=FALSE))
summary(itemsetsFreq)

## set of 15090 itemsets
##
## most frequent items:
##                       Godfather, The (1972)
##                                       2892
##                         Pulp Fiction (1994)
##                                       2163
##              Godfather: Part II, The (1974)
##                                       1704
##                                 Fargo (1996)
##                                       1587
## Star Wars: Episode IV - A New Hope (1977)
##                                       1460
##                                    (Other)
```

```
itemsetsFreq = eclat(transacoes5, par=list(sup=0.02))

## Eclat
##
## parameter specification:
##  tidLists support minlen maxlen            target
##     FALSE    0.02      1     10 frequent itemsets
##    ext
##  FALSE
##
## algorithmic control:
##  sparse sort verbose
##       7   -2    TRUE
##
## Absolute minimum support count: 12
##
## create itemset ...
## set transactions ...[3127 item(s), 644 transaction(s)] c
## sorting and recoding items ... [259 item(s)] done [0.00s
```

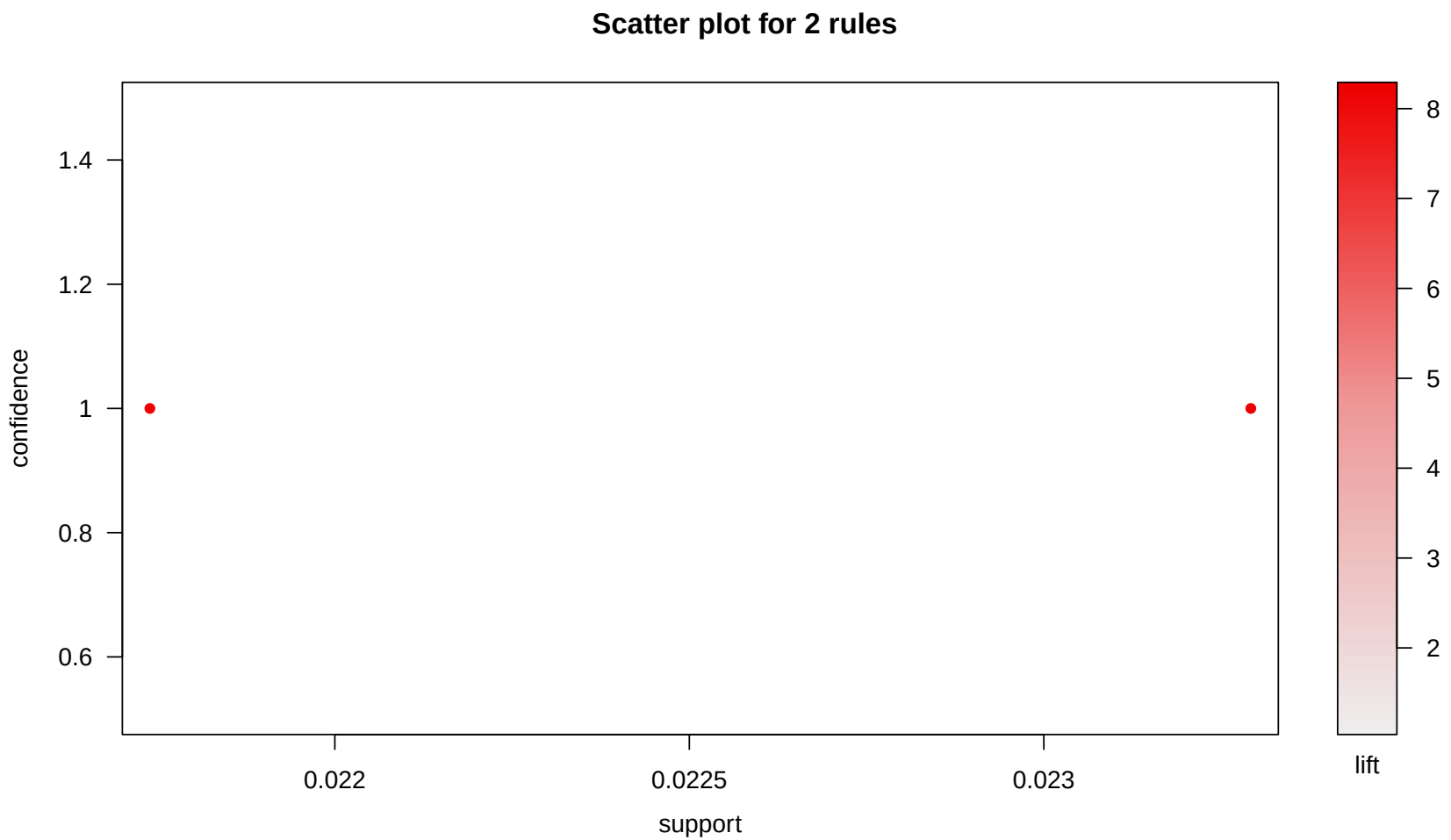
```
plot(maisRegras)

## To reduce overplotting, jitter is added!  Use
jitter = 0 to prevent jitter.
```



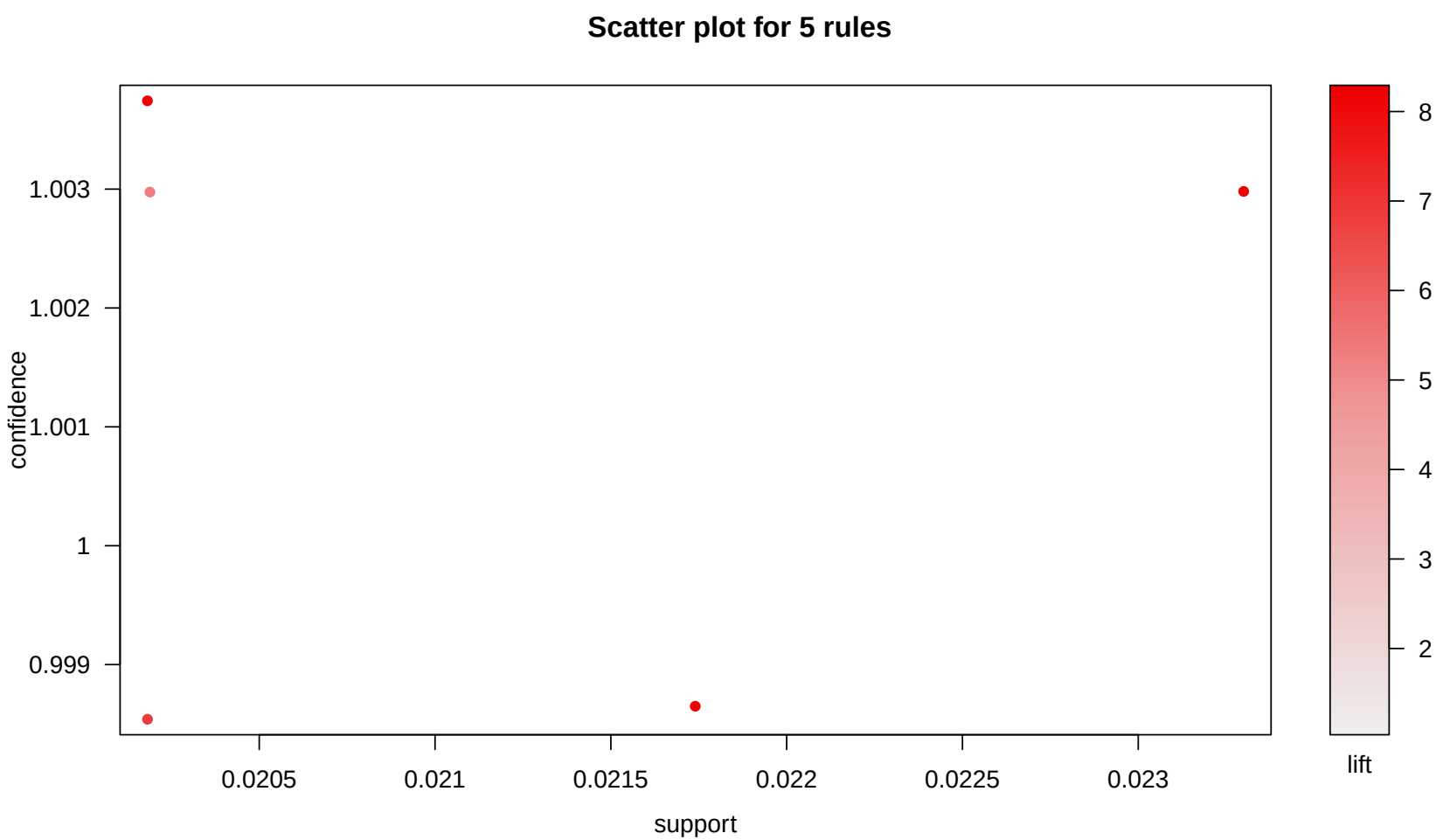
```
plot(subset(maisRegras,lhs %pin% "\\(200"))

## To reduce overplotting, jitter is added!  Use
jitter = 0 to prevent jitter.
```



```
plot(subset(maisRegras,rhs %pin% "\\(200"))
```



```r
plot(subset(maisRegras,
            (lhs %pin% "20") | (rhs %pin% "20")))

## To reduce overplotting, jitter is added!  Use
jitter = 0 to prevent jitter.
```



```r
lift4<- subset(maisRegras,
                    lift > 4)
inspect(lift4[1:2],itemSep="+",
        setStart="\n",setEnd="\n")
```

```
##     lhs
## [1]
## Shrek (2001)+
##      Lord of the Rings: The Two Towers, The (2002)
##                 =>
## Lord of the Rings: The Fellowship of the Ring, The (200
##                          0.02173913          1 8.25641
## [2]
## Indiana Jones and the Last Crusade (1989)+
##      Lord of the Rings: The Fellowship of the Ring, The
##  =>
## Raiders of the Lost Ark (Indiana Jones and the Raiders o
##  0.02018634          1 8.25641
```

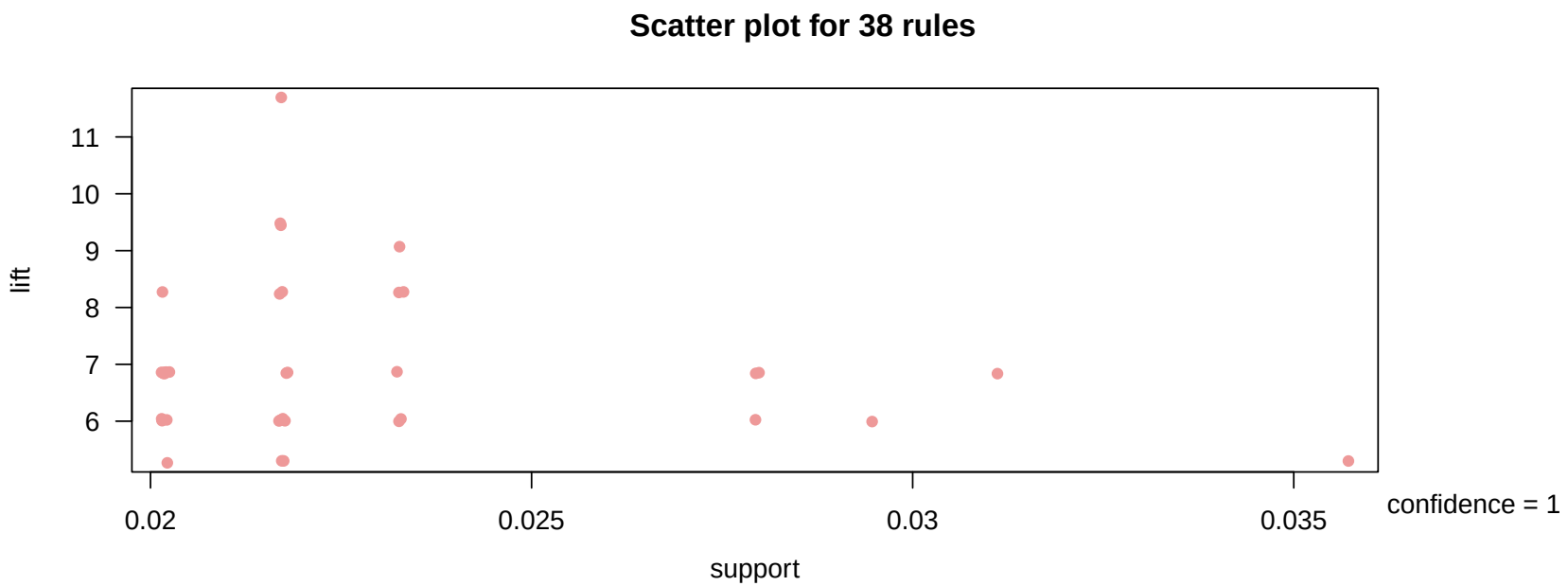
Ordenação de regras

```
inspect(sort(maisRegras,by=c("lift","conf")))
```

```
##      lhs
## [1]  {Schindler's List (1993),
##       Citizen Kane (1941)}
## [2]  {Star Wars: Episode IV - A New Hope (1977),
##       Star Wars: Episode V - The Empire Strikes Back (19
##       Men in Black (a.k.a. MIB) (1997)}
## [3]  {Star Wars: Episode V - The Empire Strikes Back (19
##       Men in Black (a.k.a. MIB) (1997)}
## [4]  {Godfather, The (1972),
##       Butch Cassidy and the Sundance Kid (1969)}
## [5]  {Shrek (2001),
##       Lord of the Rings: The Two Towers, The (2002)}
## [6]  {Indiana Jones and the Last Crusade (1989),
##       Lord of the Rings: The Fellowship of the Ring, The
## [7]  {Princess Bride, The (1987),
```

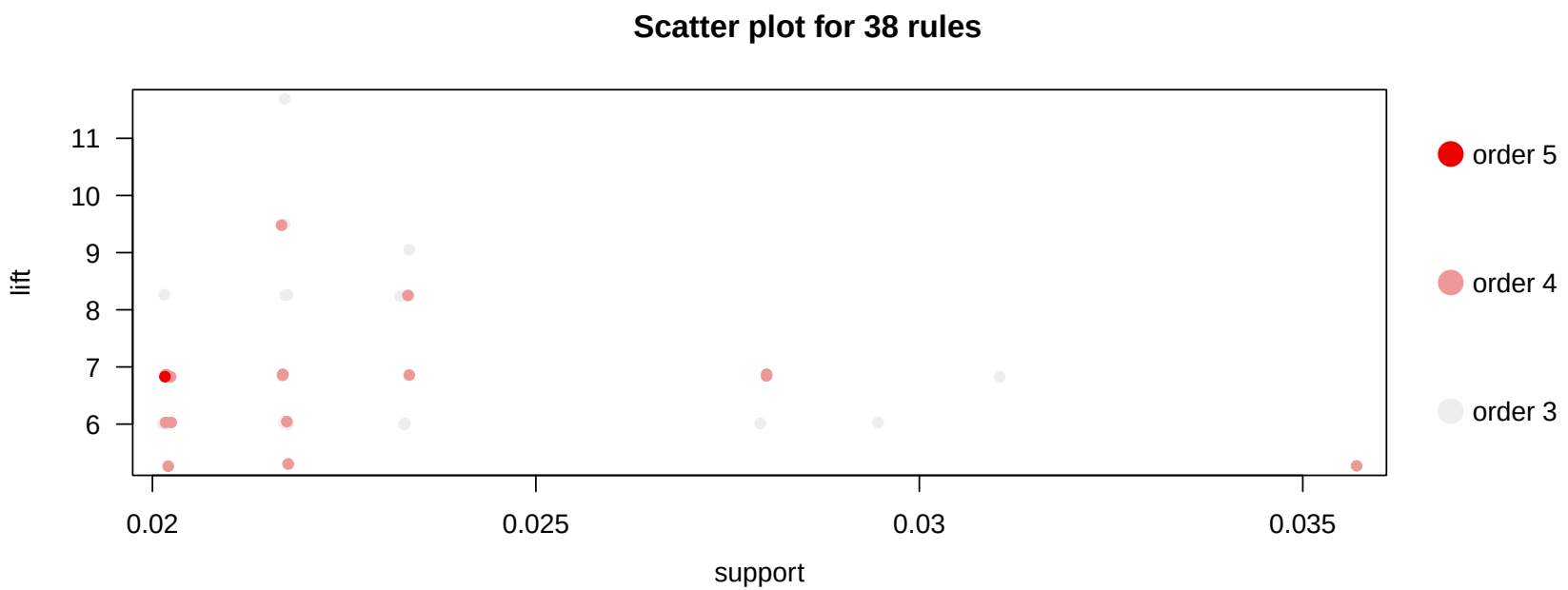
Visualização de regras

```
plot(lift4,measure=c("support", "lift"),
     shading="confidence")

## To reduce overplotting, jitter is added!  Use
jitter = 0 to prevent jitter.
```

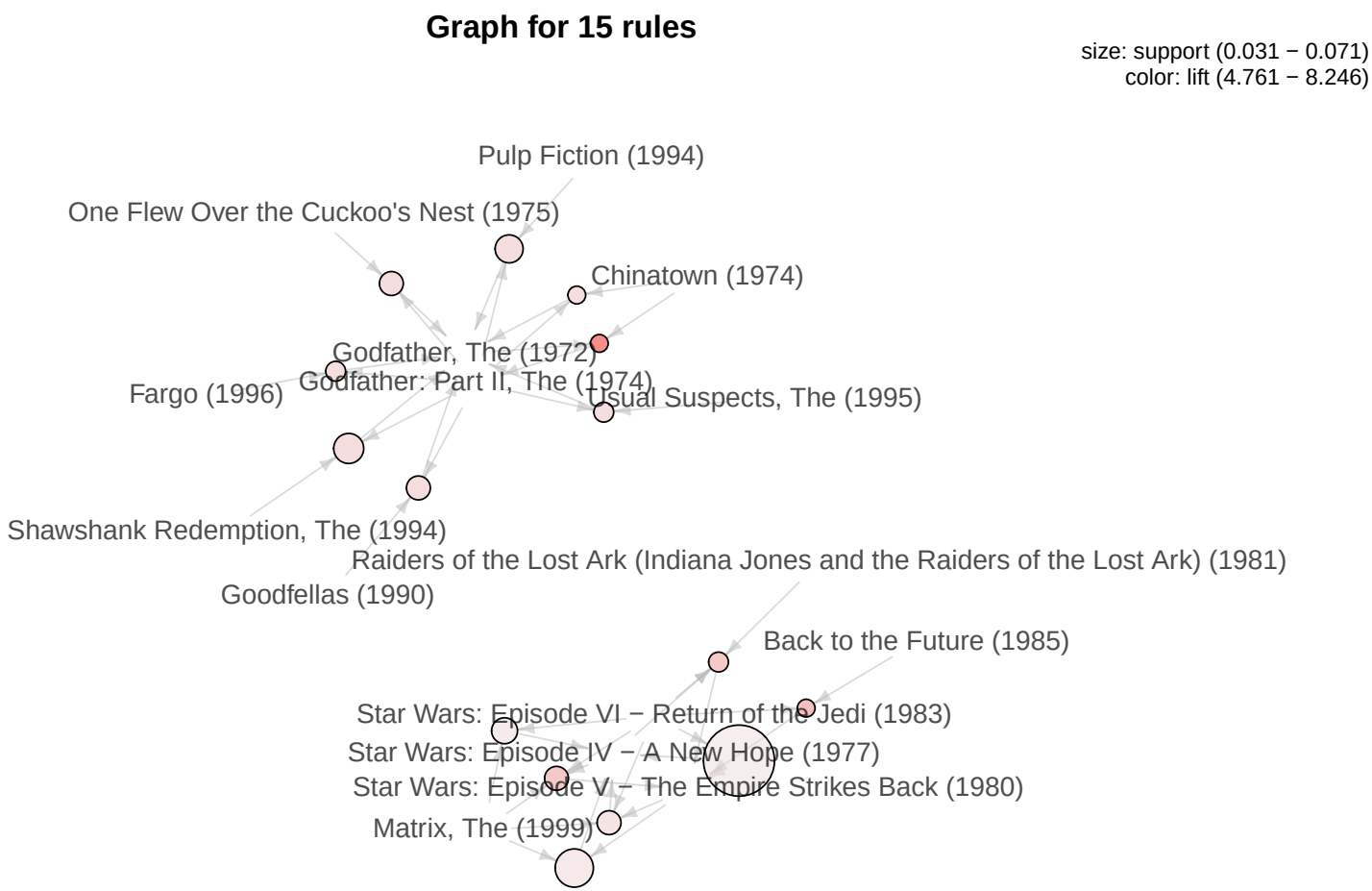


Visualização de regras

```
plot(maisRegras,measure=c("support", "lift"),
     shading="order")

## To reduce overplotting, jitter is added!  Use
jitter = 0 to prevent jitter.
```



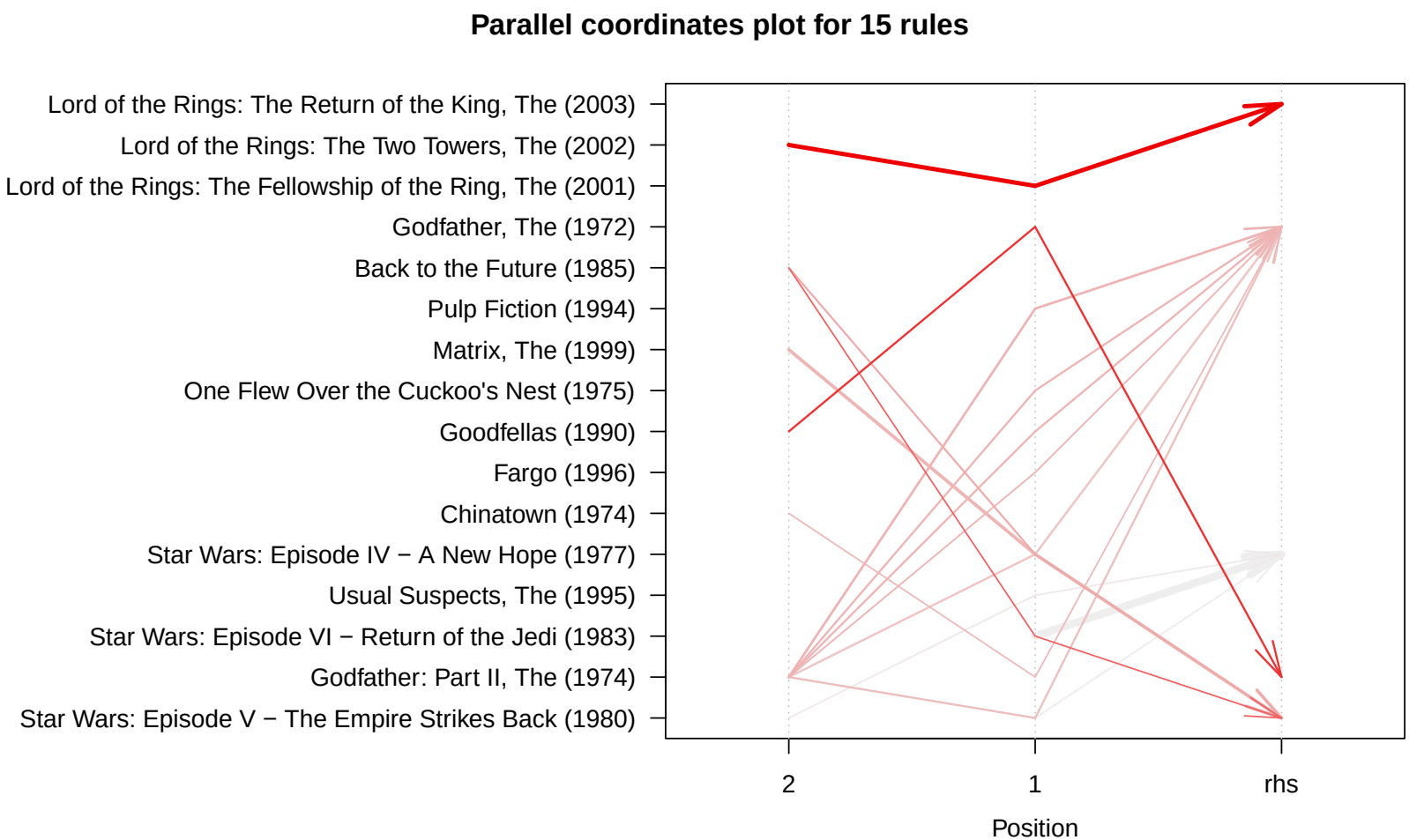
```
r <- ruleInduction(eclat(transacoes5,par=list(sup=0.03),
                control=list(verbose=FALSE)), conf=0.9)
plot(r,method="graph")
```



```r
is <- eclat(transacoes5,par=list(sup=0.03),
            control=list(verbose=FALSE))
r<-ruleInduction(is,transacoes5)

#Testar interatividade
plot(r,method="matrix",measure="lift",interactive=TRUE)
```

```
## Itemsets in Antecedent (LHS)
##  [1] "{Lord of the Rings: The Fellowship of the Ring, Th
##  [2] "{Godfather, The (1972),Chinatown (1974)}"
##  [3] "{Lord of the Rings: The Fellowship of the Ring, Th
##  [4] "{Godfather, The (1972),Goodfellas (1990)}"
##  [5] "{Indiana Jones and the Last Crusade (1989)}"
##  [6] "{Star Wars: Episode VI - Return of the Jedi (1983)
##  [7] "{Lord of the Rings: The Two Towers, The (2002),Lor
##  [8] "{Star Wars: Episode IV - A New Hope (1977),Star Wa
##  [9] "{Star Wars: Episode IV - A New Hope (1977),Raiders
## [10] "{Raiders of the Lost Ark (Indiana Jones and the Ra
## [11] "{Star Wars: Episode IV - A New Hope (1977),Termina
```

```
plot(sample(r,15),method="paracoord")
```



Exercícios

- Encontrar conjuntos de regras interessantes para as substâncias cuja exploração foram concedidadas para uma mesma empresa (usar Cessoes_de_Direitos.csv [link] do trabalho 1)

Outros assuntos relacionados a mineração de regras

- Mineração usando amostragem: $n = \frac{-2\ln(c)}{\tau\epsilon^2}$ para suporte mínimo τ, acurácia de suporte $1-\epsilon$ e nível de confiança $1-c$ [Zaki et al., 1997]
- Amostragem progressiva até acurácia desejada para estimativa do suporte [Parthasarathy, 2002]
- Para bases em memória secundária, usar uma amostra com um limiar de suporte mais baixo para selecionar candidatos para verificar suporte mínimo em uma passada na memória secundária [Toivonen et al., 1996]

Softwares de mineração de regras

- Mineração de regras de sequências: pacote `arulesSequences`
- Mineração de subgrafos: pacote subgraphMining [link]
- ELKI - [link]
- APriori, Eclat, FPGrowth de Christian Borgelt - [link]

Datasets disponíveis para testes

- `http://www.cs.uef.fi/~whamalai/datasets.html`
- `http://fimi.ua.ac.be/data/`


Parthasarathy, S. (2002).
Efficient progressive sampling for association rules.
In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 354–361. IEEE.


Toivonen, H. et al. (1996).
Sampling large databases for association rules.
In *VLDB*, volume 96, pages 134–145.


Zaki, M. J., Parthasarathy, S., Li, W., and Ogihara, M. (1997).
Evaluation of sampling for data mining of association rules.
In *Research Issues in Data Engineering, 1997. Proceedings. Seventh International Workshop on*, pages 42–50. IEEE.