Homework 3

Pattern Mining and Social Network Analysis

BOUYSSOU Gatien , de POURTALES Caroline, LAMBA Ankit

07 novembre, 2020

Contents

Parameters in association rules

There are parameters controlling the number of rules to be generated.

Support

Support is an indication of how frequently the itemset appears in the dataset.

$$Support(A \rightarrow B) = \frac{\text{Number of transaction with both A and B}}{\text{Total Number of transaction}} = P(A \cap B)$$

Confidence

Confidence is an indication of how often the rule has been found to be true.

$$Confidence(A \rightarrow B) = \frac{\text{Number of transaction with both A and B}}{\text{Total Number of transaction with A}} = \frac{P(A \cap B)}{P(A)} = \frac{P_A(B)}{P(A)^2}$$

Lift

Lift is the factor by which, the co-occurence of A and B exceeds the expected probability of A and B co-occuring, had they been independent. So, higher the lift, higher the chance of A and B occurring together.

$$Lift(A \rightarrow B) = \frac{P(A \cap B)}{P(A) * P(B)}$$

Leverage

The leverage compares the frequency of A and B appearing together and the frequency that would be expected if A and B were independent.

$$levarage(A \rightarrow B) = P(A \cap B) - P(A) \times P(B)$$

Therefore, if A and B independent :

$levarage(A \rightarrow B) = 0$

Conviction

The conviction correspond to the frequency of items that are not B in the transaction over the frequency of B that don't contain A among all the transcations with B. Therefore, if A and B are independent the conviction should be equal to 1. When the confidence tends toward 1 the conviction tends toward infinity. It would mean that A and B are higly dependant.

$$conviction(A \rightarrow B) = \frac{P(A) * P(\bar{B})}{P(A \cap \bar{B})}$$

or :

$$conviction(A \rightarrow B) = \frac{1 - P(B)}{1 - \frac{P(A \cap B)}{P(A)}}$$

Apriori algorithm

Definition

Apriori searches for frequent itemset browsing the lattice of itemsets in breadth.
The database is scanned at each level of lattice. Additionally, Apriori uses a pruning technique based on the properties of the itemsets, which are: If an itemset is frequent, all its sub-sets are frequent and not need to be considered.

Example on Groceries data

```
##     items
## [1] {citrus fruit,
##      semi-finished bread,
##      margarine,
##      ready soups}
## [2] {tropical fruit,
##      yogurt,
##      coffee}
## [3] {whole milk}
## [4] {pip fruit,
##      yogurt,
##      cream cheese ,
##      meat spreads}
## [5] {other vegetables,
##      whole milk,
##      condensed milk,
##      long life bakery product}
## [6] {whole milk,
##      butter,
##      yogurt,
##      rice,
##      abrasive cleaner}
```

On R

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.2    0.1    1 none FALSE            TRUE       5    0.03      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 295
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
## sorting and recoding items ... [44 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 done [0.00s].
```

```
## writing ... [26 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
## set of 26 rules
```

```
##     lhs                     rhs                support    confidence coverage
## [1] {whipped/sour cream} => {whole milk}       0.03223183 0.4496454  0.07168277
## [2] {root vegetables}    => {whole milk}       0.04890696 0.4486940  0.10899847
## [3] {root vegetables}    => {other vegetables} 0.04738180 0.4347015  0.10899847
## [4] {tropical fruit}     => {whole milk}       0.04229792 0.4031008  0.10493137
## [5] {yogurt}             => {whole milk}       0.05602440 0.4016035  0.13950178
##     lift     count
## [1] 1.759754 317
## [2] 1.756031 481
## [3] 2.246605 466
## [4] 1.577595 416
## [5] 1.571735 551
```

Using Frequent itemset to find rules

Concept

TO DO

Example on Adult data

We call also use the ruleInduction method to find closed frequent itemset.

ruleInduction has as attribute a method function.

Closed Frequent itemsets :

An itemset X is a closed frequent itemset in set S if X is both closed and frequent in S.

Eclat algorithm :

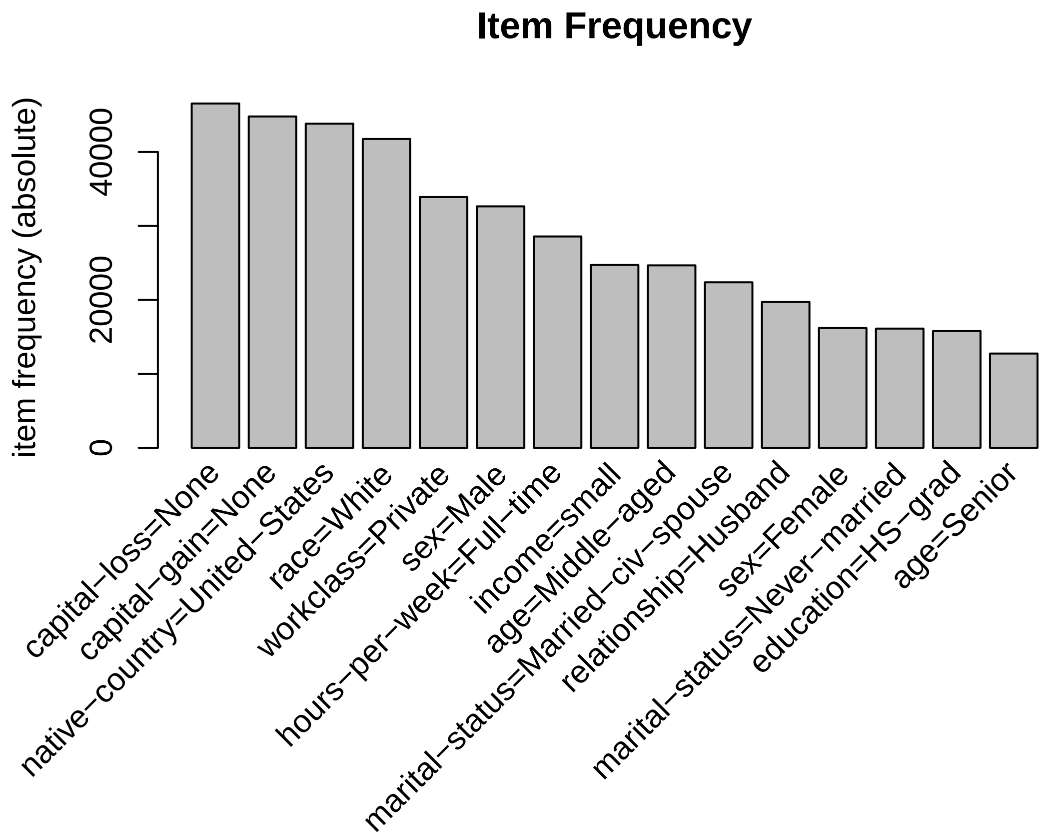
Mine frequent itemsets
This algorithm uses simple intersection operations for equivalence class clustering along with bottom-up lattice traversal.

On R

```
##     items                             transactionID
## [1] {age=Middle-aged,
##      workclass=State-gov,
##      education=Bachelors,
##      marital-status=Never-married,
##      occupation=Adm-clerical,
##      relationship=Not-in-family,
##      race=White,
##      sex=Male,
##      capital-gain=Low,
##      capital-loss=None,
##      hours-per-week=Full-time,
##      native-country=United-States,
##      income=small}                                 1
## [2] {age=Senior,
##      workclass=Self-emp-not-inc,
##      education=Bachelors,
##      marital-status=Married-civ-spouse,
##      occupation=Exec-managerial,
##      relationship=Husband,
##      race=White,
##      sex=Male,
##      capital-gain=None,
##      capital-loss=None,
##      hours-per-week=Part-time,
##      native-country=United-States,
##      income=small}                                 2
## [3] {age=Middle-aged,
##      workclass=Private,
##      education=HS-grad,
##      marital-status=Divorced,
##      occupation=Handlers-cleaners,
##      relationship=Not-in-family,
```

```
##      race=White,
##      sex=Male,
##      capital-gain=None,
##      capital-loss=None,
##      hours-per-week=Full-time,
##      native-country=United-States,
##      income=small}                                  3
## [4] {age=Senior,
##      workclass=Private,
##      education=11th,
##      marital-status=Married-civ-spouse,
##      occupation=Handlers-cleaners,
##      relationship=Husband,
##      race=Black,
##      sex=Male,
##      capital-gain=None,
##      capital-loss=None,
##      hours-per-week=Full-time,
##      native-country=United-States,
##      income=small}                                  4
## [5] {age=Middle-aged,
##      workclass=Private,
##      education=Bachelors,
##      marital-status=Married-civ-spouse,
##      occupation=Prof-specialty,
##      relationship=Wife,
##      race=Black,
##      sex=Female,
##      capital-gain=None,
##      capital-loss=None,
##      hours-per-week=Full-time,
##      native-country=Cuba,
##      income=small}                                  5
```

```
## Eclat
##
## parameter specification:
##  tidLists support minlen maxlen            target  ext
##     FALSE    0.01      1    100 frequent itemsets TRUE
##
## algorithmic control:
##  sparse sort verbose
##       7   -2    TRUE
##
## Absolute minimum support count: 488
##
## create itemset ...
## set transactions ...[115 item(s), 48842 transaction(s)] done [0.07s].
## sorting and recoding items ... [67 item(s)] done [0.01s].
## creating bit matrix ... [67 row(s), 48842 column(s)] done [0.01s].
## writing  ... [80228 set(s)] done [0.28s].
## Creating S4 object  ... done [0.02s].
```



If in control method = "apriori" is used, a very simple rule induction method is used. All rules are mined from the transactions data set using Apriori with the minimal support found in itemsets. And in a second step all rules which do not stem from one of the itemsets are removed. This procedure will be in many cases very slow (e.g., for itemsets with many elements or very low support).

```
##     lhs                                     rhs                    support confidence     lift
## [1] {marital-status=Married-civ-spouse,
##      sex=Female,
##      capital-gain=None,
##      native-country=United-States,
##      income=large}                       => {relationship=Wife} 0.01095369  0.9870849 20.68263
## [2] {marital-status=Married-civ-spouse,
##      race=White,
##      sex=Female,
##      capital-gain=None,
##      income=large}                       => {relationship=Wife} 0.01076942  0.9868668 20.67806
## [3] {marital-status=Married-civ-spouse,
##      race=White,
##      sex=Female,
##      native-country=United-States,
##      income=large}                       => {relationship=Wife} 0.01238688  0.9837398 20.61254
## [4] {marital-status=Married-civ-spouse,
##      race=White,
##      sex=Female,
##      capital-loss=None,
##      native-country=United-States,
##      income=large}                       => {relationship=Wife} 0.01113796  0.9837251 20.61223
## [5] {marital-status=Married-civ-spouse,
```

```
##      sex=Female,
##      capital-gain=None,
##      income=large}                          => {relationship=Wife} 0.01220261  0.9834983 20.60748
```

If in control method = "ptree" is used, the transactions are counted into a prefix tree and then the rules are selectively generated using the counts in the tree. This is usually faster than the above approach.

```
##     lhs                                    rhs                    support confidence     lift itemse
## [1] {marital-status=Married-civ-spouse,
##      sex=Female,
##      capital-gain=None,
##      native-country=United-States,
##      income=large}                          => {relationship=Wife} 0.01095369  0.9870849 20.68263    559
## [2] {marital-status=Married-civ-spouse,
##      race=White,
##      sex=Female,
##      capital-gain=None,
##      income=large}                          => {relationship=Wife} 0.01076942  0.9868668 20.67806    558
## [3] {marital-status=Married-civ-spouse,
##      race=White,
##      sex=Female,
##      native-country=United-States,
##      income=large}                          => {relationship=Wife} 0.01238688  0.9837398 20.61254    558
## [4] {marital-status=Married-civ-spouse,
##      race=White,
##      sex=Female,
##      capital-loss=None,
##      native-country=United-States,
##      income=large}                          => {relationship=Wife} 0.01113796  0.9837251 20.61223    558
## [5] {marital-status=Married-civ-spouse,
##      sex=Female,
##      capital-gain=None,
##      income=large}                          => {relationship=Wife} 0.01220261  0.9834983 20.60748    559
```

NOW THE BIG QUESTION ???

How to win money ?

```
## Eclat
##
## parameter specification:
##  tidLists support minlen maxlen            target  ext
##     FALSE    0.01      1    200 frequent itemsets TRUE
##
## algorithmic control:
##  sparse sort verbose
##       7   -2    TRUE
##
## Absolute minimum support count: 488
##
## create itemset ...
## set transactions ...[115 item(s), 48842 transaction(s)] done [0.05s].
## sorting and recoding items ... [67 item(s)] done [0.01s].
## creating bit matrix ... [67 row(s), 48842 column(s)] done [0.01s].
## writing  ... [80228 set(s)] done [0.26s].
## Creating S4 object  ... done [0.03s].
```

```
## set of 14 rules
##      lhs                              rhs                     support confidence     lift
## [1]  {capital-loss=None,
##       hours-per-week=Over-time,
##       income=large}                 => {capital-gain=High} 0.01148602  0.1817887 5.253802
## [2]  {race=White,
##       capital-loss=None,
##       hours-per-week=Over-time,
##       income=large}                 => {capital-gain=High} 0.01052373  0.1779778 5.143665
## [3]  {capital-loss=None,
##       hours-per-week=Over-time,
##       native-country=United-States,
##       income=large}                 => {capital-gain=High} 0.01046231  0.1779248 5.142132
## [4]  {hours-per-week=Over-time,
##       income=large}                 => {capital-gain=High} 0.01148602  0.1625145 4.696765
## [5]  {capital-loss=None,
##       income=large}                 => {capital-gain=High} 0.02319725  0.1602999 4.632763
## [6]  {capital-loss=None,
##       native-country=United-States,
##       income=large}                 => {capital-gain=High} 0.02119078  0.1600680 4.626061
## [7]  {hours-per-week=Over-time,
##       native-country=United-States,
##       income=large}                 => {capital-gain=High} 0.01046231  0.1594881 4.609302
## [8]  {race=White,
##       hours-per-week=Over-time,
##       income=large}                 => {capital-gain=High} 0.01052373  0.1591824 4.600466
## [9]  {race=White,
##       capital-loss=None,
##       native-country=United-States,
##       income=large}                 => {capital-gain=High} 0.01951190  0.1578860 4.562999
## [10] {race=White,
##       capital-loss=None,
##       income=large}                 => {capital-gain=High} 0.02069940  0.1576977 4.557557
## [11] {sex=Male,
##       capital-loss=None,
##       income=large}                 => {capital-gain=High} 0.01887720  0.1539232 4.448472
## [12] {sex=Male,
##       capital-loss=None,
##       native-country=United-States,
##       income=large}                 => {capital-gain=High} 0.01719831  0.1529776 4.421143
## [13] {race=White,
##       sex=Male,
##       capital-loss=None,
##       income=large}                 => {capital-gain=High} 0.01705499  0.1520906 4.395507
## [14] {race=White,
##       sex=Male,
##       capital-loss=None,
##       native-country=United-States,
##       income=large}                 => {capital-gain=High} 0.01605176  0.1518203 4.387696
```

Example on mushroom data

With python and scikit-learn

This database contains a lot of mushrooms with a set of characteristics. Each mushroom is classified either as edible or poisonous. The database has been found in kaggle and is available here : https://www.kaggle.com/uciml/mushroom-classification.

First, we want to have an overview of the data.

```
##   class cap-shape cap-surface  ... spore-print-color population habitat
## 0     p         x           s  ...                 k          s       u
## 1     e         x           s  ...                 n          n       g
## 2     e         b           s  ...                 n          n       m
## 3     p         x           y  ...                 k          s       u
## 4     e         x           s  ...                 n          a       g
##
## [5 rows x 23 columns]
```

As we can see, each column contains values that are single characters. Their meaning is given by the file values_name.txt.

```
## 8124
```

Now, we want to know the data repartition for each columns.

```
##
## class
## e    4208
## p    3916
## Name: class, dtype: int64
##
## cap-shape
## x    3656
## f    3152
## k     828
## b     452
## s      32
## c       4
## Name: cap-shape, dtype: int64
##
## cap-surface
## y    3244
## s    2556
## f    2320
## g       4
## Name: cap-surface, dtype: int64
##
## cap-color
## n    2284
## g    1840
## e    1500
## y    1072
## w    1040
## b     168
## p     144
## c      44
## r      16
```

```
## u      16
## Name: cap-color, dtype: int64
##
## bruises
## f    4748
## t    3376
## Name: bruises, dtype: int64
##
## odor
## n    3528
## f    2160
## s     576
## y     576
## l     400
## a     400
## p     256
## c     192
## m      36
## Name: odor, dtype: int64
##
## gill-attachment
## f    7914
## a     210
## Name: gill-attachment, dtype: int64
##
## gill-spacing
## c    6812
## w    1312
## Name: gill-spacing, dtype: int64
##
## gill-size
## b    5612
## n    2512
## Name: gill-size, dtype: int64
##
## gill-color
## b    1728
## p    1492
## w    1202
## n    1048
## g     752
## h     732
## u     492
## k     408
## e      96
## y      86
## o      64
## r      24
## Name: gill-color, dtype: int64
##
## stalk-shape
## t    4608
## e    3516
## Name: stalk-shape, dtype: int64
```

```
##
## stalk-root
## b    3776
## ?    2480
## e    1120
## c     556
## r     192
## Name: stalk-root, dtype: int64
##
## stalk-surface-above-ring
## s    5176
## k    2372
## f     552
## y      24
## Name: stalk-surface-above-ring, dtype: int64
##
## stalk-surface-below-ring
## s    4936
## k    2304
## f     600
## y     284
## Name: stalk-surface-below-ring, dtype: int64
##
## stalk-color-above-ring
## w    4464
## p    1872
## g     576
## n     448
## b     432
## o     192
## e      96
## c      36
## y       8
## Name: stalk-color-above-ring, dtype: int64
##
## stalk-color-below-ring
## w    4384
## p    1872
## g     576
## n     512
## b     432
## o     192
## e      96
## c      36
## y      24
## Name: stalk-color-below-ring, dtype: int64
##
## veil-type
## p    8124
## Name: veil-type, dtype: int64
##
## veil-color
## w    7924
## o      96
```

```
## n      96
## y       8
## Name: veil-color, dtype: int64
##
## ring-number
## o    7488
## t     600
## n      36
## Name: ring-number, dtype: int64
##
## ring-type
## p    3968
## e    2776
## l    1296
## f      48
## n      36
## Name: ring-type, dtype: int64
##
## spore-print-color
## w    2388
## n    1968
## k    1872
## h    1632
## r      72
## o      48
## b      48
## u      48
## y      48
## Name: spore-print-color, dtype: int64
##
## population
## v    4040
## y    1712
## s    1248
## n     400
## a     384
## c     340
## Name: population, dtype: int64
##
## habitat
## d    3148
## g    2148
## p    1144
## l     832
## u     368
## m     292
## w     192
## Name: habitat, dtype: int64
```

As you can see the there is almost as much poisonous as edible mushrooms. Moreover, the dataset contains some unknown values in the column stalk-root. We are going to discard those rows to keep lines that are complete.

```
## 5644
```

```
## e    3488
## p    2156
## Name: class, dtype: int64
```

Even without the discarded lines the dataset still have plenty of data and the class label is almost balanced.

```
## [['b', 'c', 'x', 'f', 'k', 's'], ['bell', 'conical', 'convex', 'flat', 'knobbed', 'sunken']]
```

```
##        class cap-shape cap-surface  ... spore-print-color population  habitat
## 0  poisonous    convex      smooth  ...             black  scattered    urban
## 1     edible    convex      smooth  ...             brown   numerous  grasses
## 2     edible      bell      smooth  ...             brown   numerous  meadows
## 3  poisonous    convex       scaly  ...             black  scattered    urban
## 4     edible    convex      smooth  ...             brown   abundant  grasses
##
## [5 rows x 23 columns]
```

```
## ['abundant', 'almond', 'anise', 'attached', 'bell', 'black', 'broad', 'brown', 'bruises', 'buff', 'bu
```

```
## 64
```

```
##       support                                     itemsets  length
## 0    0.875266                                      (broad)       1
## 1    0.642452                                      (brown)       1
## 2    0.669029                                    (bulbous)       1
## 3    0.818568                                      (close)       1
## 4    0.618001                                     (edible)       1
## ..        ...                                          ...     ...
## 186  0.616584        (pendant, white, partial, free, smooth)       5
## 187  0.608079  (white, broad, partial, close, bulbous, free)       6
## 188  0.711552      (white, partial, close, one, broad, free)       6
## 189  0.600992     (white, partial, one, broad, free, smooth)       6
## 190  0.603827    (white, partial, close, one, bulbous, free)       6
##
## [191 rows x 3 columns]
```

```
##       support                            itemsets  length
## 4    0.618001                            (edible)       1
## 31   0.618001                      (edible, free)       2
## 32   0.609497                       (edible, one)       2
## 33   0.618001                   (partial, edible)       2
## 34   0.618001                     (white, edible)       2
## 89   0.609497                 (edible, free, one)       3
## 90   0.618001             (edible, partial, free)       3
## 91   0.618001               (edible, white, free)       3
## 92   0.609497              (partial, edible, one)       3
## 93   0.609497                (white, edible, one)       3
## 94   0.618001            (partial, edible, white)       3
## 151  0.609497        (edible, partial, free, one)       4
## 152  0.609497          (edible, white, free, one)       4
## 153  0.618001      (edible, partial, free, white)       4
## 154  0.609497       (partial, edible, white, one)       4
## 184  0.609497  (white, partial, edible, one, free)       5
```

```
## Empty DataFrame
## Columns: [support, itemsets, length]
## Index: []
```

```
##           antecedents                    consequents  ...  leverage  conviction
## 19           (edible)                         (free)  ...  0.001971         inf
## 20           (edible)                          (one)  ...  0.008577    2.008505
## 21           (edible)                      (partial)  ...  0.000000         inf
## 22           (edible)                        (white)  ...  0.000000         inf
## 152     (edible, free)                         (one)  ...  0.008577    2.008505
## ..                ...                            ...  ...       ...         ...
## 818    (white, edible)        (partial, free, one)  ...  0.008577    2.008505
## 819  (partial, edible)          (white, free, one)  ...  0.008577    2.008505
## 820      (edible, one)      (white, partial, free)  ...  0.001944         inf
## 821     (edible, free)       (white, partial, one)  ...  0.008577    2.008505
## 822           (edible)  (white, partial, free, one)  ...  0.008577    2.008505
##
## [65 rows x 9 columns]
```

```
##       support                             itemsets
## 9    0.618001                             (edible)
## 129  0.618001                       (edible, free)
## 130  0.618001                    (partial, edible)
## 131  0.618001                      (white, edible)
## 132  0.609497                        (edible, one)
## 133  0.618001              (edible, partial, free)
## 134  0.618001                (edible, white, free)
## 135  0.609497                  (edible, free, one)
## 136  0.618001             (partial, edible, white)
## 137  0.609497               (partial, edible, one)
## 138  0.609497                 (white, edible, one)
## 139  0.618001        (edible, partial, free, white)
## 140  0.609497          (edible, partial, free, one)
## 141  0.609497            (edible, white, free, one)
## 142  0.609497         (partial, edible, white, one)
## 143  0.609497  (white, partial, edible, one, free)
```

```
## Empty DataFrame
## Columns: [antecedents, consequents, antecedent support, consequent support, support, confidence, lif
## Index: []
```

```
##          antecedents                      consequents  ...  leverage  conviction
## 1285          (free)                          (edible)  ...  0.001971    1.005203
## 1286       (partial)                          (edible)  ...  0.000000    1.000000
## 1288         (white)                          (edible)  ...  0.000000    1.000000
## 1291           (one)                          (edible)  ...  0.008577    1.023637
## 1294  (partial, free)                         (edible)  ...  0.001971    1.005203
## ...              ...                              ...  ...       ...         ...
## 1408     (free, one)          (white, partial, edible)  ...  0.008577    1.023637
## 1409         (white)      (partial, edible, one, free)  ...  0.000000    1.000000
## 1410       (partial)        (white, edible, one, free)  ...  0.000000    1.000000
## 1412           (one)    (white, partial, edible, free)  ...  0.008577    1.023637
## 1413          (free)     (white, partial, edible, one)  ...  0.001944    1.005019
##
## [65 rows x 9 columns]
```

Clustering with Apriori algorithm as dissimilarity measure

Concept

TO DO

Jaccard distance

A direct approach to cluster itemsets is to define a distance metric between two itemsets X_i and X_j. A good choice is the Jaccard distance defined as :

$$d(X_i, X_j) = \frac{|X_i \cap X_j|}{|X_i \cup X_j|}$$

The distance simply is the number of items that Xi and Xj have in common divided by the number of unique items in both sets.

Example on tennis data

On R

```
##     items        transactionID
## [1] {Result=0,
##      ACE.1=Low,
##      UFE.1=Low,
##      ACE.2=Low,
##      UFE.2=Low}               1
## [2] {Result=0,
##      ACE.1=None,
##      UFE.1=Low,
##      ACE.2=High,
##      UFE.2=Low}               2
## [3] {Result=1,
##      ACE.1=Low,
##      UFE.1=Low,
##      ACE.2=Low,
##      UFE.2=High}              3
## [4] {Result=1,
##      ACE.1=High,
##      UFE.1=High,
##      ACE.2=None,
##      UFE.2=High}              4
## [5] {Result=0,
##      ACE.1=Low,
##      UFE.1=High,
##      ACE.2=High,
##      UFE.2=High}              5
```

The associations rules for Player-1 winning :

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.3    0.1    1 none FALSE            TRUE       5    0.15      1
##  maxlen target  ext
```

```
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 17
##
## set item appearances ...[1 item(s)] done [0.00s].
## set transactions ...[12 item(s), 118 transaction(s)] done [0.00s].
## sorting and recoding items ... [11 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [13 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
## set of 13 rules
```

```
##     lhs                          rhs        support   confidence coverage
## [1] {ACE.1=High,UFE.1=Low} => {Result=1} 0.1525424 0.8181818  0.1864407
## [2] {ACE.1=High}           => {Result=1} 0.2881356 0.6938776  0.4152542
## [3] {ACE.1=High,UFE.2=Low} => {Result=1} 0.1694915 0.6451613  0.2627119
## [4] {ACE.2=Low}            => {Result=1} 0.2457627 0.6170213  0.3983051
## [5] {UFE.1=Low}            => {Result=1} 0.3220339 0.6129032  0.5254237
##     lift     count
## [1] 1.532468 18
## [2] 1.299644 34
## [3] 1.208397 20
## [4] 1.155691 29
## [5] 1.147977 38
```

The associations rules for Player-1 loosing :

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.3    0.1    1 none FALSE            TRUE       5    0.15      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 17
##
## set item appearances ...[1 item(s)] done [0.00s].
## set transactions ...[12 item(s), 118 transaction(s)] done [0.00s].
## sorting and recoding items ... [11 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [10 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
## set of 10 rules
```

```
##     lhs                         rhs        support   confidence coverage
## [1] {ACE.2=High}             => {Result=0} 0.2372881 0.6086957  0.3898305
## [2] {UFE.1=High}             => {Result=0} 0.2627119 0.5535714  0.4745763
## [3] {ACE.1=Low}              => {Result=0} 0.2372881 0.5283019  0.4491525
## [4] {UFE.2=Low}              => {Result=0} 0.2796610 0.5238095  0.5338983
## [5] {UFE.1=High,UFE.2=High} => {Result=0} 0.1525424 0.4864865  0.3135593
##     lift     count
## [1] 1.305929 28
## [2] 1.187662 31
## [3] 1.133448 28
## [4] 1.123810 33
## [5] 1.043735 18
```

All the rules with Result as association :

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.4    0.1    1 none FALSE            TRUE       5     0.1      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 11
##
## set item appearances ...[2 item(s)] done [0.00s].
## set transactions ...[12 item(s), 118 transaction(s)] done [0.00s].
## sorting and recoding items ... [12 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [46 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
## set of 46 rules
```

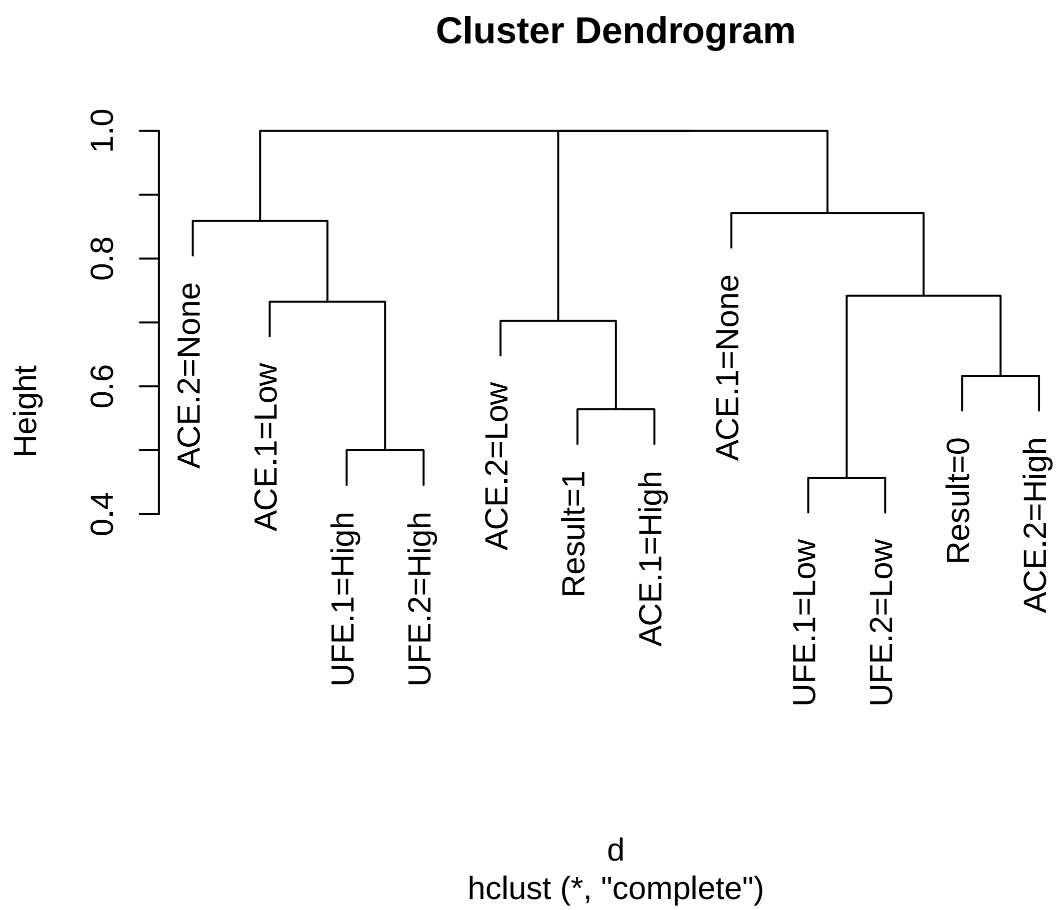
```
##     lhs                                 rhs        support   confidence
## [1] {ACE.1=None}                     => {Result=0} 0.1016949 0.7500000
## [2] {ACE.1=High,UFE.1=Low}           => {Result=1} 0.1525424 0.8181818
## [3] {UFE.1=High,ACE.2=High}          => {Result=0} 0.1186441 0.7000000
## [4] {ACE.2=High,UFE.2=Low}           => {Result=0} 0.1355932 0.6956522
## [5] {ACE.1=High,UFE.1=Low,UFE.2=Low} => {Result=1} 0.1271186 0.7894737
##     coverage  lift     count
## [1] 0.1355932 1.609091 12
## [2] 0.1864407 1.532468 18
## [3] 0.1694915 1.501818 14
## [4] 0.1949153 1.492490 16
## [5] 0.1610169 1.478697 15
```

Cluster the items

```
##      lhs                                 rhs        support   confidence
## [1]  {}                               => {Result=0} 0.4661017 0.4661017
```

```
## [2]  {}                                => {Result=1} 0.5338983 0.5338983
## [3]  {ACE.1=None}                       => {Result=0} 0.1016949 0.7500000
## [4]  {ACE.2=None}                       => {Result=1} 0.1355932 0.6400000
## [5]  {ACE.2=High}                       => {Result=0} 0.2372881 0.6086957
## [6]  {ACE.2=Low}                        => {Result=1} 0.2457627 0.6170213
## [7]  {ACE.1=High}                       => {Result=1} 0.2881356 0.6938776
## [8]  {ACE.1=Low}                        => {Result=0} 0.2372881 0.5283019
## [9]  {ACE.1=Low}                        => {Result=1} 0.2118644 0.4716981
## [10] {UFE.2=High}                       => {Result=0} 0.1864407 0.4000000
## [11] {UFE.1=High}                       => {Result=0} 0.2627119 0.5535714
## [12] {UFE.2=Low}                        => {Result=0} 0.2796610 0.5238095
## [13] {UFE.2=High}                       => {Result=1} 0.2796610 0.6000000
## [14] {UFE.1=High}                       => {Result=1} 0.2118644 0.4464286
## [15] {UFE.1=Low}                        => {Result=1} 0.3220339 0.6129032
## [16] {UFE.2=Low}                        => {Result=1} 0.2542373 0.4761905
## [17] {ACE.1=Low,ACE.2=High}             => {Result=0} 0.1101695 0.6500000
## [18] {ACE.2=High,UFE.2=High}            => {Result=0} 0.1016949 0.5217391
## [19] {UFE.1=High,ACE.2=High}            => {Result=0} 0.1186441 0.7000000
## [20] {UFE.1=Low,ACE.2=High}             => {Result=0} 0.1186441 0.5384615
## [21] {ACE.2=High,UFE.2=Low}             => {Result=0} 0.1355932 0.6956522
## [22] {UFE.1=Low,ACE.2=High}             => {Result=1} 0.1016949 0.4615385
## [23] {ACE.1=High,ACE.2=Low}             => {Result=1} 0.1440678 0.7727273
## [24] {UFE.1=High,ACE.2=Low}             => {Result=0} 0.1016949 0.4615385
## [25] {ACE.2=Low,UFE.2=Low}              => {Result=0} 0.1016949 0.4800000
## [26] {ACE.2=Low,UFE.2=High}             => {Result=1} 0.1355932 0.7272727
## [27] {UFE.1=High,ACE.2=Low}             => {Result=1} 0.1186441 0.5384615
## [28] {UFE.1=Low,ACE.2=Low}              => {Result=1} 0.1271186 0.7142857
## [29] {ACE.2=Low,UFE.2=Low}              => {Result=1} 0.1101695 0.5200000
## [30] {ACE.1=High,UFE.2=High}            => {Result=1} 0.1186441 0.7777778
## [31] {ACE.1=High,UFE.1=High}            => {Result=1} 0.1355932 0.5925926
## [32] {ACE.1=High,UFE.1=Low}             => {Result=1} 0.1525424 0.8181818
## [33] {ACE.1=High,UFE.2=Low}             => {Result=1} 0.1694915 0.6451613
## [34] {ACE.1=Low,UFE.2=High}             => {Result=0} 0.1101695 0.4333333
## [35] {ACE.1=Low,UFE.1=High}             => {Result=0} 0.1271186 0.6521739
## [36] {ACE.1=Low,UFE.1=Low}              => {Result=0} 0.1101695 0.4333333
## [37] {ACE.1=Low,UFE.2=Low}              => {Result=0} 0.1271186 0.6521739
## [38] {ACE.1=Low,UFE.2=High}             => {Result=1} 0.1440678 0.5666667
## [39] {ACE.1=Low,UFE.1=Low}              => {Result=1} 0.1440678 0.5666667
## [40] {UFE.1=High,UFE.2=High}            => {Result=0} 0.1525424 0.4864865
## [41] {UFE.1=High,UFE.2=Low}             => {Result=0} 0.1101695 0.6842105
## [42] {UFE.1=Low,UFE.2=Low}              => {Result=0} 0.1694915 0.4545455
## [43] {UFE.1=High,UFE.2=High}            => {Result=1} 0.1610169 0.5135135
## [44] {UFE.1=Low,UFE.2=High}             => {Result=1} 0.1186441 0.7777778
## [45] {UFE.1=Low,UFE.2=Low}              => {Result=1} 0.2033898 0.5454545
## [46] {ACE.1=High,UFE.1=Low,UFE.2=Low} => {Result=1} 0.1271186 0.7894737
##      coverage  lift      count
## [1]  1.0000000 1.0000000 55
## [2]  1.0000000 1.0000000 63
## [3]  0.1355932 1.6090909 12
## [4]  0.2118644 1.1987302 16
## [5]  0.3898305 1.3059289 28
## [6]  0.3983051 1.1556906 29
## [7]  0.4152542 1.2996437 34
## [8]  0.4491525 1.1334477 28
```

```
## [9]  0.4491525 0.8834981 25
## [10] 0.4661017 0.8581818 22
## [11] 0.4745763 1.1876623 31
## [12] 0.5338983 1.1238095 33
## [13] 0.4661017 1.1238095 33
## [14] 0.4745763 0.8361678 25
## [15] 0.5254237 1.1479775 38
## [16] 0.5338983 0.8919123 30
## [17] 0.1694915 1.3945455 13
## [18] 0.1949153 1.1193676 12
## [19] 0.1694915 1.5018182 14
## [20] 0.2203390 1.1552448 14
## [21] 0.1949153 1.4924901 16
## [22] 0.2203390 0.8644689 12
## [23] 0.1864407 1.4473304 17
## [24] 0.2203390 0.9902098 12
## [25] 0.2118644 1.0298182 12
## [26] 0.1864407 1.3621934 16
## [27] 0.2203390 1.0085470 14
## [28] 0.1779661 1.3378685 15
## [29] 0.2118644 0.9739683 13
## [30] 0.1525424 1.4567901 14
## [31] 0.2288136 1.1099353 16
## [32] 0.1864407 1.5324675 18
## [33] 0.2627119 1.2083973 20
## [34] 0.2542373 0.9296970 13
## [35] 0.1949153 1.3992095 15
## [36] 0.2542373 0.9296970 13
## [37] 0.1949153 1.3992095 15
## [38] 0.2542373 1.0613757 17
## [39] 0.2542373 1.0613757 17
## [40] 0.3135593 1.0437346 18
## [41] 0.1610169 1.4679426 13
## [42] 0.3728814 0.9752066 20
## [43] 0.3135593 0.9618190 19
## [44] 0.1525424 1.4567901 14
## [45] 0.3728814 1.0216450 24
## [46] 0.1610169 1.4786967 15
```




```
##      items         transactionID
## [1] {Result=0,
##      ACE.1=Low,
##      UFE.1=Low,
##      ACE.2=Low,
##      UFE.2=Low}                1
## [2] {Result=0,
##      ACE.1=None,
##      UFE.1=Low,
##      ACE.2=High,
##      UFE.2=Low}                2
## [3] {Result=1,
##      ACE.1=Low,
##      UFE.1=Low,
##      ACE.2=Low,
##      UFE.2=High}               3
## [4] {Result=0,
##      ACE.1=Low,
##      UFE.1=High,
##      ACE.2=High,
##      UFE.2=High}               5
## [5] {Result=0,
##      ACE.1=High,
##      UFE.1=High,
##      ACE.2=High,
```

```
##      UFE.2=High}            6
## [6] {Result=0,
##      ACE.1=Low,
##      UFE.1=Low,
##      ACE.2=High,
##      UFE.2=Low}             9
## [7] {Result=1,
##      ACE.1=High,
##      UFE.1=High,
##      ACE.2=Low,
##      UFE.2=High}            10
## [8] {Result=0,
##      ACE.1=Low,
##      UFE.1=Low,
##      ACE.2=High,
##      UFE.2=High}            11
```

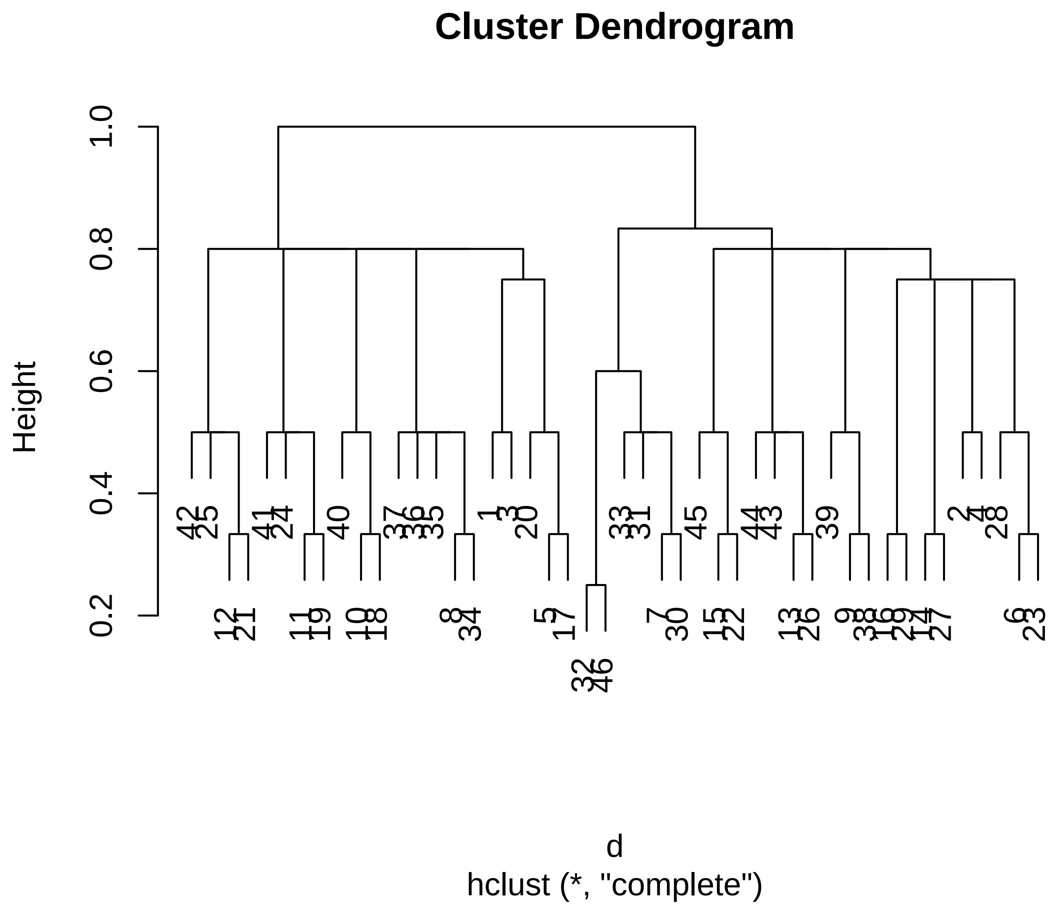
Cluster the rules

- With Jaccard measure :

```
##      lhs                            rhs        support   confidence
## [1]  {}                          => {Result=0} 0.4661017 0.4661017
## [2]  {}                          => {Result=1} 0.5338983 0.5338983
## [3]  {ACE.1=None}                => {Result=0} 0.1016949 0.7500000
## [4]  {ACE.2=None}                => {Result=1} 0.1355932 0.6400000
## [5]  {ACE.2=High}                => {Result=0} 0.2372881 0.6086957
## [6]  {ACE.2=Low}                 => {Result=1} 0.2457627 0.6170213
## [7]  {ACE.1=High}                => {Result=1} 0.2881356 0.6938776
## [8]  {ACE.1=Low}                 => {Result=0} 0.2372881 0.5283019
## [9]  {ACE.1=Low}                 => {Result=1} 0.2118644 0.4716981
## [10] {UFE.2=High}                => {Result=0} 0.1864407 0.4000000
## [11] {UFE.1=High}                => {Result=0} 0.2627119 0.5535714
## [12] {UFE.2=Low}                 => {Result=0} 0.2796610 0.5238095
## [13] {UFE.2=High}                => {Result=1} 0.2796610 0.6000000
## [14] {UFE.1=High}                => {Result=1} 0.2118644 0.4464286
## [15] {UFE.1=Low}                 => {Result=1} 0.3220339 0.6129032
## [16] {UFE.2=Low}                 => {Result=1} 0.2542373 0.4761905
## [17] {ACE.1=Low,ACE.2=High}      => {Result=0} 0.1101695 0.6500000
## [18] {ACE.2=High,UFE.2=High}     => {Result=0} 0.1016949 0.5217391
## [19] {UFE.1=High,ACE.2=High}     => {Result=0} 0.1186441 0.7000000
## [20] {UFE.1=Low,ACE.2=High}      => {Result=0} 0.1186441 0.5384615
## [21] {ACE.2=High,UFE.2=Low}      => {Result=0} 0.1355932 0.6956522
## [22] {UFE.1=Low,ACE.2=High}      => {Result=1} 0.1016949 0.4615385
## [23] {ACE.1=High,ACE.2=Low}      => {Result=1} 0.1440678 0.7727273
## [24] {UFE.1=High,ACE.2=Low}      => {Result=0} 0.1016949 0.4615385
## [25] {ACE.2=Low,UFE.2=Low}       => {Result=0} 0.1016949 0.4800000
## [26] {ACE.2=Low,UFE.2=High}      => {Result=1} 0.1355932 0.7272727
## [27] {UFE.1=High,ACE.2=Low}      => {Result=1} 0.1186441 0.5384615
## [28] {UFE.1=Low,ACE.2=Low}       => {Result=1} 0.1271186 0.7142857
## [29] {ACE.2=Low,UFE.2=Low}       => {Result=1} 0.1101695 0.5200000
## [30] {ACE.1=High,UFE.2=High}     => {Result=1} 0.1186441 0.7777778
## [31] {ACE.1=High,UFE.1=High}     => {Result=1} 0.1355932 0.5925926
## [32] {ACE.1=High,UFE.1=Low}      => {Result=1} 0.1525424 0.8181818
## [33] {ACE.1=High,UFE.2=Low}      => {Result=1} 0.1694915 0.6451613
```

```
## [34] {ACE.1=Low,UFE.2=High}          => {Result=0} 0.1101695 0.4333333
## [35] {ACE.1=Low,UFE.1=High}          => {Result=0} 0.1271186 0.6521739
## [36] {ACE.1=Low,UFE.1=Low}           => {Result=0} 0.1101695 0.4333333
## [37] {ACE.1=Low,UFE.2=Low}           => {Result=0} 0.1271186 0.6521739
## [38] {ACE.1=Low,UFE.2=High}          => {Result=1} 0.1440678 0.5666667
## [39] {ACE.1=Low,UFE.1=Low}           => {Result=1} 0.1440678 0.5666667
## [40] {UFE.1=High,UFE.2=High}         => {Result=0} 0.1525424 0.4864865
## [41] {UFE.1=High,UFE.2=Low}          => {Result=0} 0.1101695 0.6842105
## [42] {UFE.1=Low,UFE.2=Low}           => {Result=0} 0.1694915 0.4545455
## [43] {UFE.1=High,UFE.2=High}         => {Result=1} 0.1610169 0.5135135
## [44] {UFE.1=Low,UFE.2=High}          => {Result=1} 0.1186441 0.7777778
## [45] {UFE.1=Low,UFE.2=Low}           => {Result=1} 0.2033898 0.5454545
## [46] {ACE.1=High,UFE.1=Low,UFE.2=Low} => {Result=1} 0.1271186 0.7894737
##      coverage  lift      count
## [1]  1.0000000 1.0000000 55
## [2]  1.0000000 1.0000000 63
## [3]  0.1355932 1.6090909 12
## [4]  0.2118644 1.1987302 16
## [5]  0.3898305 1.3059289 28
## [6]  0.3983051 1.1556906 29
## [7]  0.4152542 1.2996437 34
## [8]  0.4491525 1.1334477 28
## [9]  0.4491525 0.8834981 25
## [10] 0.4661017 0.8581818 22
## [11] 0.4745763 1.1876623 31
## [12] 0.5338983 1.1238095 33
## [13] 0.4661017 1.1238095 33
## [14] 0.4745763 0.8361678 25
## [15] 0.5254237 1.1479775 38
## [16] 0.5338983 0.8919123 30
## [17] 0.1694915 1.3945455 13
## [18] 0.1949153 1.1193676 12
## [19] 0.1694915 1.5018182 14
## [20] 0.2203390 1.1552448 14
## [21] 0.1949153 1.4924901 16
## [22] 0.2203390 0.8644689 12
## [23] 0.1864407 1.4473304 17
## [24] 0.2203390 0.9902098 12
## [25] 0.2118644 1.0298182 12
## [26] 0.1864407 1.3621934 16
## [27] 0.2203390 1.0085470 14
## [28] 0.1779661 1.3378685 15
## [29] 0.2118644 0.9739683 13
## [30] 0.1525424 1.4567901 14
## [31] 0.2288136 1.1099353 16
## [32] 0.1864407 1.5324675 18
## [33] 0.2627119 1.2083973 20
## [34] 0.2542373 0.9296970 13
## [35] 0.1949153 1.3992095 15
## [36] 0.2542373 0.9296970 13
## [37] 0.1949153 1.3992095 15
## [38] 0.2542373 1.0613757 17
## [39] 0.2542373 1.0613757 17
## [40] 0.3135593 1.0437346 18
```

```
## [41] 0.1610169 1.4679426 13
## [42] 0.3728814 0.9752066 20
## [43] 0.3135593 0.9618190 19
## [44] 0.1525424 1.4567901 14
## [45] 0.3728814 1.0216450 24
## [46] 0.1610169 1.4786967 15
```

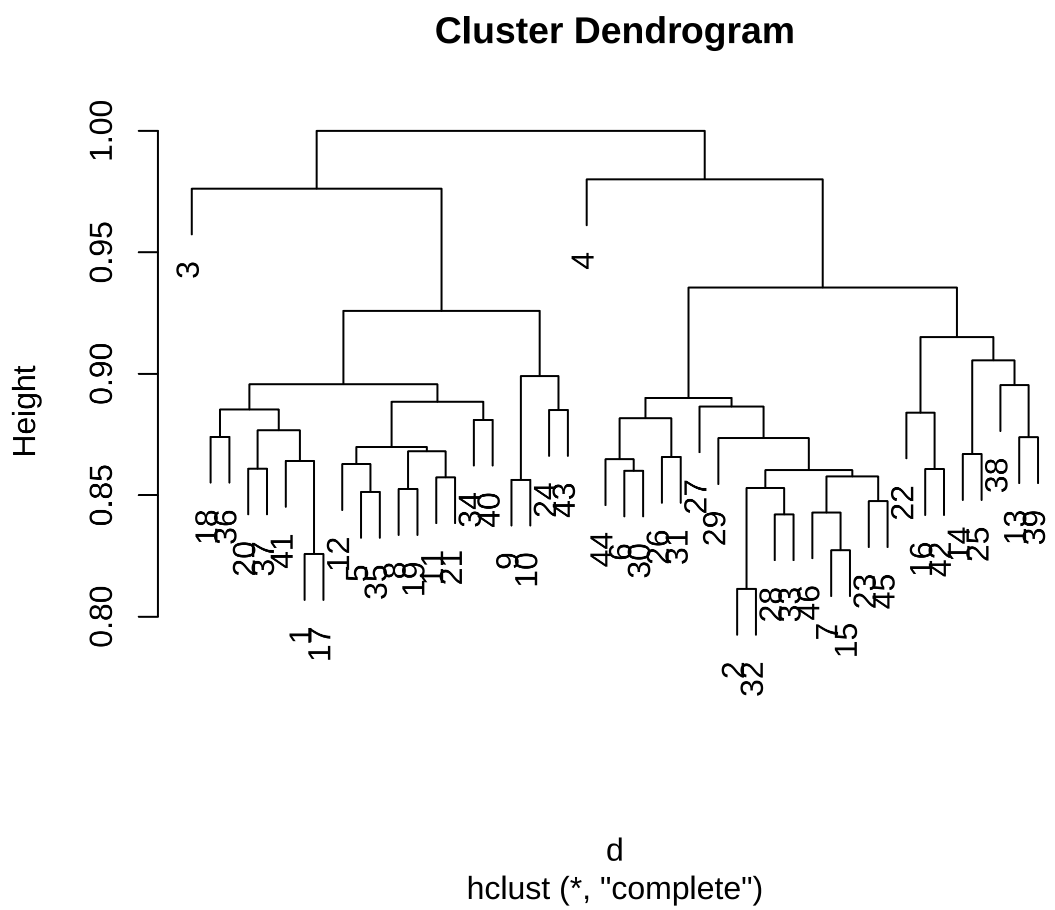



```
##      lhs                          rhs          support   confidence coverage
## [1]  {}                        => {Result=0} 0.4661017 0.4661017  1.0000000
## [2]  {ACE.1=None}              => {Result=0} 0.1016949 0.7500000  0.1355932
## [3]  {ACE.2=High}              => {Result=0} 0.2372881 0.6086957  0.3898305
## [4]  {ACE.1=Low}               => {Result=0} 0.2372881 0.5283019  0.4491525
## [5]  {UFE.2=High}              => {Result=0} 0.1864407 0.4000000  0.4661017
## [6]  {UFE.1=High}              => {Result=0} 0.2627119 0.5535714  0.4745763
## [7]  {UFE.2=Low}               => {Result=0} 0.2796610 0.5238095  0.5338983
## [8]  {ACE.1=Low,ACE.2=High}    => {Result=0} 0.1101695 0.6500000  0.1694915
## [9]  {ACE.2=High,UFE.2=High}   => {Result=0} 0.1016949 0.5217391  0.1949153
## [10] {UFE.1=High,ACE.2=High}   => {Result=0} 0.1186441 0.7000000  0.1694915
## [11] {UFE.1=Low,ACE.2=High}    => {Result=0} 0.1186441 0.5384615  0.2203390
## [12] {ACE.2=High,UFE.2=Low}    => {Result=0} 0.1355932 0.6956522  0.1949153
## [13] {UFE.1=High,ACE.2=Low}    => {Result=0} 0.1016949 0.4615385  0.2203390
## [14] {ACE.2=Low,UFE.2=Low}     => {Result=0} 0.1016949 0.4800000  0.2118644
## [15] {ACE.1=Low,UFE.2=High}    => {Result=0} 0.1101695 0.4333333  0.2542373
## [16] {ACE.1=Low,UFE.1=High}    => {Result=0} 0.1271186 0.6521739  0.1949153
## [17] {ACE.1=Low,UFE.1=Low}     => {Result=0} 0.1101695 0.4333333  0.2542373
## [18] {ACE.1=Low,UFE.2=Low}     => {Result=0} 0.1271186 0.6521739  0.1949153
```

```
## [19] {UFE.1=High,UFE.2=High} => {Result=0} 0.1525424 0.4864865  0.3135593
## [20] {UFE.1=High,UFE.2=Low}  => {Result=0} 0.1101695 0.6842105  0.1610169
## [21] {UFE.1=Low,UFE.2=Low}   => {Result=0} 0.1694915 0.4545455  0.3728814
##      lift      count
## [1]  1.0000000 55
## [2]  1.6090909 12
## [3]  1.3059289 28
## [4]  1.1334477 28
## [5]  0.8581818 22
## [6]  1.1876623 31
## [7]  1.1238095 33
## [8]  1.3945455 13
## [9]  1.1193676 12
## [10] 1.5018182 14
## [11] 1.1552448 14
## [12] 1.4924901 16
## [13] 0.9902098 12
## [14] 1.0298182 12
## [15] 0.9296970 13
## [16] 1.3992095 15
## [17] 0.9296970 13
## [18] 1.3992095 15
## [19] 1.0437346 18
## [20] 1.4679426 13
## [21] 0.9752066 20
```

This clustering regroups Player-1 winner together very well.

- With affinity measure :




```
##      lhs                          rhs        support   confidence coverage
## [1]  {}                        => {Result=0} 0.4661017 0.4661017  1.0000000
## [2]  {ACE.1=None}              => {Result=0} 0.1016949 0.7500000  0.1355932
## [3]  {ACE.2=High}              => {Result=0} 0.2372881 0.6086957  0.3898305
## [4]  {ACE.1=Low}               => {Result=0} 0.2372881 0.5283019  0.4491525
## [5]  {ACE.1=Low}               => {Result=1} 0.2118644 0.4716981  0.4491525
## [6]  {UFE.2=High}              => {Result=0} 0.1864407 0.4000000  0.4661017
## [7]  {UFE.1=High}              => {Result=0} 0.2627119 0.5535714  0.4745763
## [8]  {UFE.2=Low}               => {Result=0} 0.2796610 0.5238095  0.5338983
## [9]  {ACE.1=Low,ACE.2=High}    => {Result=0} 0.1101695 0.6500000  0.1694915
## [10] {ACE.2=High,UFE.2=High}   => {Result=0} 0.1016949 0.5217391  0.1949153
## [11] {UFE.1=High,ACE.2=High}   => {Result=0} 0.1186441 0.7000000  0.1694915
## [12] {UFE.1=Low,ACE.2=High}    => {Result=0} 0.1186441 0.5384615  0.2203390
## [13] {ACE.2=High,UFE.2=Low}    => {Result=0} 0.1355932 0.6956522  0.1949153
## [14] {UFE.1=High,ACE.2=Low}    => {Result=0} 0.1016949 0.4615385  0.2203390
## [15] {ACE.1=Low,UFE.2=High}    => {Result=0} 0.1101695 0.4333333  0.2542373
## [16] {ACE.1=Low,UFE.1=High}    => {Result=0} 0.1271186 0.6521739  0.1949153
## [17] {ACE.1=Low,UFE.1=Low}     => {Result=0} 0.1101695 0.4333333  0.2542373
## [18] {ACE.1=Low,UFE.2=Low}     => {Result=0} 0.1271186 0.6521739  0.1949153
## [19] {UFE.1=High,UFE.2=High}   => {Result=0} 0.1525424 0.4864865  0.3135593
## [20] {UFE.1=High,UFE.2=Low}    => {Result=0} 0.1101695 0.6842105  0.1610169
## [21] {UFE.1=High,UFE.2=High}   => {Result=1} 0.1610169 0.5135135  0.3135593
##      lift      count
## [1]  1.0000000 55
## [2]  1.6090909 12
```

```
## [3]  1.3059289 28
## [4]  1.1334477 28
## [5]  0.8834981 25
## [6]  0.8581818 22
## [7]  1.1876623 31
## [8]  1.1238095 33
## [9]  1.3945455 13
## [10] 1.1193676 12
## [11] 1.5018182 14
## [12] 1.1552448 14
## [13] 1.4924901 16
## [14] 0.9902098 12
## [15] 0.9296970 13
## [16] 1.3992095 15
## [17] 0.9296970 13
## [18] 1.3992095 15
## [19] 1.0437346 18
## [20] 1.4679426 13
## [21] 0.9618190 19
```

The CLIQUE algorithm

The ENCLUS algorithm

ENtropy-based CLUStering

Frequent pattern-based classification

Classification based on Association

CBA Algorithm

Implementation the CBA algorithm with the M1 or M2 pruning strategy introduced by Liu, et al. (1998).

Candidate classification association rules (CARs) are mined with the standard APRIORI algorithm. Rules are ranked by confidence, support and size. Then either the M1 or M2 algorithm are used to perform database coverage pruning and to determin the number of rules to use and the default class.

TO DO DEFINITION

Example on tennis data

Recall from Homework 1 With Random Forest, the accuracy rate was 0.6931818.
With Logistic regression it was 0.7667.

From classification to associations rules

```
##     lhs                     rhs          support confidence coverage lift count size coveredTransactions
## [1] {ACE.1=[3.5, Inf],
##      ACE.2=[-Inf,1.5),
##      UFE.2=[8.5, Inf]} => {Result=1}   0.125      0.917    0.136 1.61    11    4                  12
## [2] {ACE.1=[3.5, Inf],
##      ACE.2=[-Inf,1.5)} => {Result=1}   0.159      0.875    0.182 1.54    14    3                   4
## [3] {ACE.1=[3.5, Inf],
##      UFE.2=[8.5, Inf]} => {Result=1}   0.216      0.760    0.284 1.34    19    3                  13
## [4] {UFE.1=[7.5, Inf],
##      ACE.2=[-Inf,1.5),
##      UFE.2=[8.5, Inf]} => {Result=1}   0.227      0.690    0.330 1.21    20    4                  18
## [5] {}                 => {Result=0}   0.432      0.432       NA 1.00    88    1                  41
```

```
##                      true
## classifier.prediction  0  1
##                     0 14  5
##                     1  3  8
```

The accuracy rate is :

```
## [1] 0.733
```

The sensitivity is the percentage of true output giving Player1-winner among the population of true Player1-winner :

```
## [1] 0.615
```

The specificity is the percentage of true output giving Player2-winner (= Player1-looser) among the population of true Player2-winner:

```
## [1] 0.824
```

The precision is the percentage of true output giving Player1-winner among all the outputs giving Player1-winner (even if not winner) :

```
## [1] 0.727
```

So the F_Mesure is :

```
## [1] 0.667
```

From associations rules to classification

```
##     items        transactionID
## [1] {Result=0,
##      ACE.1=Low,
##      UFE.1=Low,
##      ACE.2=Low,
##      UFE.2=Low}             1
## [2] {Result=0,
##      ACE.1=None,
##      UFE.1=Low,
##      ACE.2=High,
##      UFE.2=Low}             2
## [3] {Result=1,
##      ACE.1=Low,
##      UFE.1=Low,
##      ACE.2=Low,
##      UFE.2=High}            3
## [4] {Result=1,
##      ACE.1=High,
##      UFE.1=High,
##      ACE.2=None,
##      UFE.2=High}            4

##
## Mining CARs...
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.4    0.1    1 none FALSE            TRUE       5    0.05      1
##  maxlen target  ext
##       5  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 4
##
## set item appearances ...[12 item(s)] done [0.00s].
## set transactions ...[12 item(s), 88 transaction(s)] done [0.00s].
## sorting and recoding items ... [12 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 done [0.00s].
## writing ... [84 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
##
## Pruning CARs...
## CARs left: 10

##
## classifier.prediction  0  1
##                     0 11  3
##                     1  6 10
```

The accuracy rate is :

```
## [1] 0.7
```

The sensitivity is the percentage of true output giving Player1-winner among the population of true Player1-winner :

```
## [1] 0.769
```

The specificity is the percentage of true output giving Player2-winner (= Player1-looser) among the population of true Player2-winner:

```
## [1] 0.647
```

The precision is the percentage of true output giving Player1-winner among all the outputs giving Player1-winner (even if not winner) :

```
## [1] 0.625
```

So the F_Mesure is :

```
## [1] 0.69
```

Classification based on Multiple Association Rules

Classification based on Predictive Association Rules

Evaluation

Compare the algorithms