Московский Государственный Университет им. М.В. Ломоносова
факультет Вычислительной математики и кибернетики
кафедра Математических методов прогнозирования

ПАКЕТ “ARULES” СИСТЕМЫ R

Отчёт *Огневой Дарьи,*
317 группа

Преподаватель: *Дьяконов*
Александр Геннадьевич

2012

Введение

Один из самых популярных и хорошо изученных подходов, используемых в анализе данных для выявления интересных связей в обширных базах данных, - ***частые наборы*** (frequent itemsets) и ***ассоциативные правила*** (association rules).
Пакет “arules” системы R представляет основу для создания и преобразования входных данных: обеспечивает фундамент для представления, преобразования и анализа транзакционных данных и моделей - *частых наборов* и *ассоциативных правил*, - так же обеспечивает интерфейс для реализации в C основанных на идее ассоциативных правил алгоритмов ***Apriori*** и ***Eclat***. Эти алгоритмы могут быть использованы для формирования *частых наборов*, *максимальных частых наборов* (maximal frequent itemsets), *объемлющих частых наборов* (closed frequent itemsets) и *ассоциативных правил*.
Так как обычно обрабатываются большие множества правил и наборов, пакет использует разряженные матрицы для уменьшения объёмов требуемой памяти. Более того, “arules” поддерживает расширение как для взаимодействия новых алгоритмов, так и для добавления новых типов, мер и ассоциаций.

Структура данных

Для работы со специфичными данными, свойственными рассматриваемой теории, в “arules” реализованы специальные типы данных, представленные на *Рисунке 1*.
Входные данные - ***transactions*** (транзакции) и ***tidLists*** (transaction ID lists - tid-списки - альтернативный способ представления транзакционных данных), на выходе из анализирующих алгоритмов - классы ***itemsets*** и ***rules***, представленные соответственно множеством наборов или правил. Оба класса являются расширением общего виртуального класса ***associations***, обеспечивающего единый интерфейс. Добавление нового типа ассоциаций в данную структуру происходит путём добавления нового класса - наследника *associations*.
Бинарные признаки (items) в классах *associations* и *transactions* реализованы классом ***itemMatrix***, являющимся базой для реализации класса ***ngCMatrix*** разряженных матриц из R-пакета Matrix.
Для контроля за поведением алгоритмов анализа используются 2 класса - ***ASparameter*** и ***AScontrol***. Т.к. каждый из алгоритмов *Apriori* и *Eclat* обладают своими спрециализированными параметрами, то для каждого из них реализованы свои соответствующие классы - с префиксом *AP* для *Apriori* и с префиксом *EC* для *Eclat*.

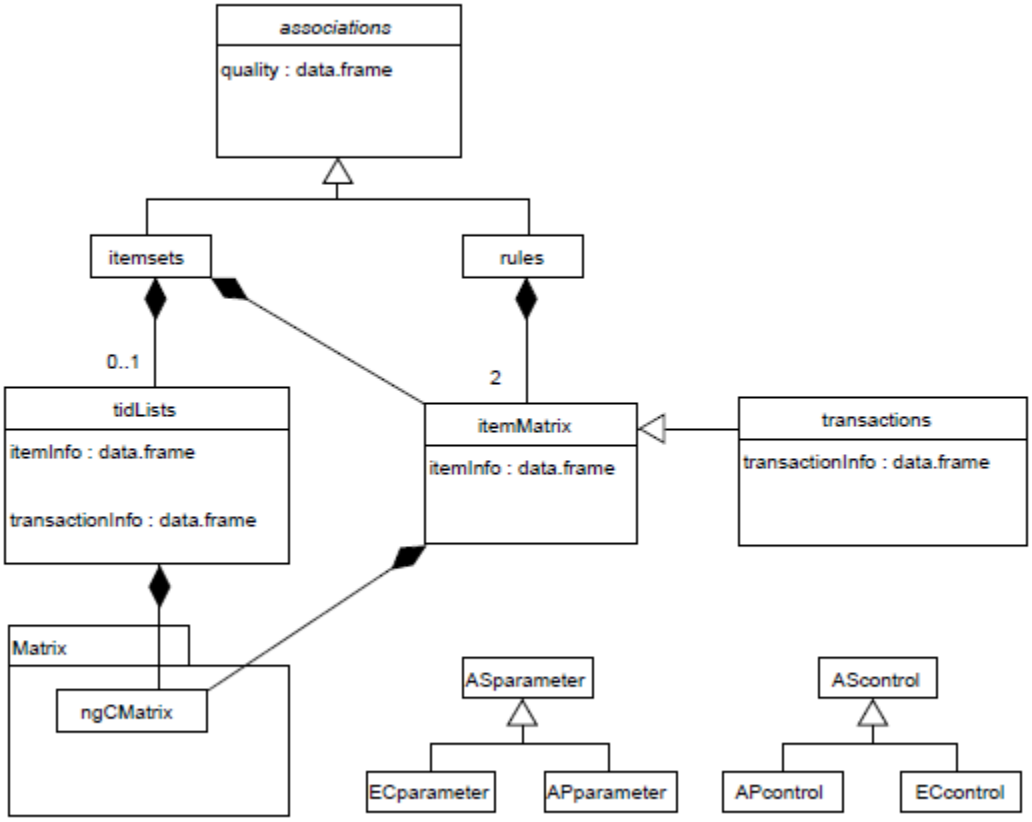

Рисунок 1: Диаграмма пакета “arules”.

Представление множеств частых наборов

Обычные частые наборы или типичные транзакции содержат относительно маленькое количество бинарных признаков по сравнению с общим количеством доступных бинарных признаков, потому естественным представлением такого типа данных является формат разряженных матриц - класс ***ngCMatrix*** пакета Matrix, в котором *ngCMatrix* - сжатая, разряженная, логическая, столбцово-ориентированная матрица, содержащая вектор элементов с значением TRUE в первой строке и указатели на первые элементы столбцов во второй. Нумерация в обоих векторах начинается с 0.

Т.к. более удобно работать с строчно-ориентированными матрицами, в пакете реализован класс ***itemMatrix*** - строчно-ориентированный вариант *ngCMatrix.* В этом случае *ngCMatrix* может быть доступна методом ***as()***, приводящим *itemMatrix* к *ngCMatrix*.

Более того, *itemMatrix* может хранить названия бинарных признаков (item labels) и

дополнительную информацию о бинарных признаках.
Основные операции с *itemMatrix* - ***dim()*** и множественный выбор (***[***). Первый элемент в описанных функциях отвечает за набор или транзакцию (строчку), второй - за бинарный признак (столбец). Например, результат команды *trans[1:5, 2:32]* - матрица первые 5 (с 1 по 5) транзакций на 31 бинарный признак(со 2 по 32).
Для определения числа наборов в *itemMatrix* используется команда ***length()***, возвращающая число строк (= первое значение *dim()*).
Одинаковые наборы находятся командой ***duplicated()***, удаляются - с помощью ***unique()***. Для поиска соотвествующих элементов в 2 множествах используется команда ***match()***.
Команда ***c()*** можно объединять по строчкам несколько объектов *itemMatrix*, т.е. создавать множество наборов всех данных объектов *itemMatrix*. Функция корректна только в случае одинакового числа и единого порядка бинарных признаков. Если же матрицы содержат одинаковое число признаков, но названия пронумерованы различным образом, необходимо использовать команду ***recode()*** для приведения к корректной форме.
Количество бинарных признаков определяется командой ***size()***, которая возвращает вектор с числом признаков, входящих в каждый набор.
Специализированные для пакета команды - ***itemFrequency()*** и ***itemFrequencyPlot()***.

itemFrequency

подсчитывает частоту встречаемости / поддержку (support) для всех отдельно взятых признаков объекта типа itemMatrix

- itemFrequency (x, ...)
- ## method for signature 'itemMatrix'
  itemFrequency (x, type)

Аргументы

x	объект
...	пропуск последующих аргументов
type	строка “relative” (по умолчанию) или “absolute” - соответственно относительная или абсолютная частота встречаемости

возвращает именованный числовой вектор, каждый элемент которого частота встречаемости/поддержка соответствующего бинарного признака объекта x; признаки нумеруются в порядке их расположения по столбцам бинарной матрицы x

Пример

```
data (“Adult”)
itemFrequency(Adult, type = “relative”)
```

itemFrequencyPlot

строит график распределения частот встречаемости признаков для объектов типа itemMatrix

- itemFrequencyPlot (x, ...)
- ## method for signature ’itemMatrix’
  itemFrequencyPlot (x, type = c("relative", "absolute"),
  support = NULL, topN = NULL,
  population = NULL, popCol = "black", popLwd = 1,
  lift = FALSE, horiz = FALSE,
  names = TRUE, cex.names = par("cex.axis"),
  xlab = NULL, ylab = NULL, mai = NULL, ...)

Аргументы

x	объект
...	пропуск последующих аргументов
type	строка “relative” (по умолчанию) или “absolute” - соответственно относительная или абсолютная частота встречаемости
support	числовое значение; отображаются только признаки с поддержкой не меньше support; если не задано population - поддержка высчитывается исходя из x, иначе - из population; является абсолютной // относительной в зависимости от type
topN	целое значение; отображается только topN признаков с бОльшей частотой или подъёмом (lift), если lift = TRUE; строятся признаки по убыванию значимости
population	объект типа класса x; если x - подматрица population, population характеризует частоту встречаемости каждого признака, отображённую линией на графике
popCol	цвет линии
popLwd	толщина линии
lift	логический тип; если lift = TRUE - график строится по значению подъёма, иначе - по частоте встречаемости; функция подъёма показывает насколько чаще встречается объект в x, чем в population

horiz	логический тип; если horiz = FALSE (по умолчанию), бар рисуется вертикально, иначе - горизонтально
names	логический индикатор, отвечающий за отображение меток
cex.names	числовое значение; коэффициент расширения для названий осей
xlab	строка; метка оси x (пустая строка в случае оси без метки)
ylab	строка; метка оси y (пустая строка в случае оси без метки)
mai	числовой вектор; размеры графика в дюймах

возвращает числовой вектор с промежуточными точками графика

Пример

```
 data (“Adult”)
 Adult.largeIncome <- Adult [Adult %in% “income=large”]

## simple plot
 itemFrequencyPlot (Adult.largeIncome)

## plot with the averages of the population plotted as a line (for first 72 variables / items)
itemFrequencyPlot (Adult.largeIncome[, 1:72], population = Adult[, 1:72])

## plot lift ratio (frequency in x / frequency in population) for items with a support of 20% ## in the population
itemFrequencyPlot (Adult.largeIncome, population = Adult, support = 0.2, lift = TRUE, horiz = TRUE)
```

Так же для быстрой визуализации *itemMatrix* можно построить график функцией ***image()***.

Для преобразования *itemMatrix* в список реализованы 2 возможности: уже упомянутая функция ***as()***, возвращающая список именованных метками признаков

векторов, и функция ***LIST()***, возвращающий список целых векторов с номерами столбцов вместо названий признаков (decode = TRUE, если декодировать нет необходимости: decode = FALSE). Так же для декодирования названий признаков в номера столбцов используется функция ***decode()***.

Транзакция данных

Для анализа транзакции данных преобразуются в бинарную матрицу со строками-наборами и столбцами-признаками со значением элемента 1, если соответствующий признак есть в транзакции, и 0, если отсутствует, реализованную в классе ***transactions***. Этот формат называется *горизонтальным расположением базы данных*. Кроме того, реализована возможность *вертикального расположения базы данных* в форме ***tid-списков*** (класс ***tidLists***). В этом случае для каждого признака формируется список транзакций, его содержащий.
Класс *transactions* является расширением класса *itemMatrix* и наследует все базовые атрибуты и методы. Более того, *transactions* содержит слот ***data.frame*** для хранения более подробной информации для каждой транзакции. Например, он может быть использован для хранения ID транзакций или покупателей, доходы или прибыль (для задачи анализа рыночных корзин). Благодаря этой информации могут быть выделены подмножества транзакций.
Объекты класса *transactions* могут быть созданы из матриц или списков или считаны из файла функцией ***read.transactions()***.

read.transactions
считывает транзакции из файла
• read.transactions(file, format = c("basket", "single"), sep = NULL, cols = NULL, rm.duplicates = FALSE, encoding = "unknown")

Аргументы

file	имя файла
format	строка; формат данных
sep	строка; определяет, как разделены столбцы в файле, или NULL (по умолчанию); в формате *basket* - это может быть регулярным выражением, или символом в случае формата *single*, по умолчанию разделителями являются пробелы
cols	для формата *single* параметр представлен числовым или символьным вектором длины 2, состоящий из номеров или имён столбца с соответственно транзакциями и идентификаторами признаков, если вектор символьный, то первая строка файла распознаётся как имена столбцов; для формата *basket* параметр представлен числом, предоставляя номер столбца с ID транзакций. для параметра NULL данные не содержат ID транзакций.
rm.duplicates	логический тип; определяет, стоит ли удалять повторяющиеся признаки в транзакциях
encoding	строка; определяет кодировку, которая передаётся в *readlines*

Детали

Для формата *basket* каждая строка в файле представляет собой транзакцию, в которой признаки (их метки) разделены символами *sep*. Для формата *single* каждая строка соответствует определённому признаку, содержащему идентификаторы транзакций и признаков.

возвращает объект класса *transactions*
Пример ## create a demo file using basket format for the example data <- paste(“item1,item2”,”item1”,”item2,item3”, sep=”\n”) cat(data) write(data, file = “demo_basket”) ## read demo data tr <- read.transactions(“demo_basket”, format = “basket”, sep=”,”) inspect(tr) ## create a demo file using single format for the example ## column 1 contains the transaction ID and column 2 contains one item data <- paste(“trans1 item1”, “trans2 item1”,”trans2 item2”, sep =”\n”) cat(data) write(data, file = “demo_single”) ## read demo data tr <- read.transactions(“demo_single”, format = “single”, cols = c(1,2)) inspect(tr)

Отображение на экране может быть реализовано функцией ***inspect()*** (см. пример выше).

Ассоциации: наборы и множества правил.

Как описывалось ранее, результат анализа транзакций данных пакетом “arules” - ***associations***, где под ассоциациями понимается множества объектов, описывающих отношения между некоторыми признаками (как наборы или правила), которые определяются различными мерами качества: мера важности (поддержки), мера заинтересованности (значимости, подъёма) или любые другие меры (вес покрытых ассоциацией признаков).
Вне зависимости от типа ассоциаций существует общая функциональная база.

summary()	возвращает короткий обзор множества
inspect()	отображает на экране всю информацию об определённой ассоциации
length()	возвращает количество элементов множества
items()	возвращает для каждой ассоциации множество признаков, участвующих в ассоциации (объединение *LHS* и *RHS* для каждого правила)

SORT()	сортирует множество по мере
[, или subset()	выделяет подмножество
union(), intersect()	различные операции над множествами: объединение, пересечение
match()	поиск соответствующих элементов первого множества во втором
WRITE()	запись ассоциаций в читабельной для человека форме
save(), load()	соответственно сохранение и загрузка компактно записанных данных

Класс *associations* применяется в пакете “arules” для реализации множества наборов и множества правил. Естественно, оба класса - *itemsets* и *rules* - наследуются текущим классом и могут использовать те же методы. Класс *itemsets* содержит один объект *itemMatrix* для хранения матрицы бинарных признаков (каждая строка - отдельный набор). Более того, он может содержать список идентификаторов транзакций - объект класса *tidLists*. Класс *rules* состоит из двух объектов *itemMatrix*, представляющих левую и правую сторону правила (***left-hand-side*** (***LHS***) и ***right-hand-side*** (***RHS***)): X=>Y, где X - *LHS*, Y - *RHS*.
Признаки в ассоциациях и мерах качества могут быть доступны и изменены, используя специальные приложения и методы для *items*, *lhs*, *rhs* и *quality*. Более того, класс *associations* имеет встроенную проверку корректности размерности данных.
Таким образом, для добавления нового типа ассоциаций необходимо просто создать новый класс - наследник *associations* - и наслаждаться определёнными, описанными выше функциями.

Алгоритмы анализа данных.

В пакете “arules” в открытом пользовании находятся функции, реализующие алгоритмы *Apriori* и *Eclat*.

apriori

анализирует частые наборы, ассоциативные правила или гиперрёбра ассоциаций, используя алгоритм Apriori; для нахождения часто встречающихся наборов используется поиск в ширину; включает некоторые улучшения

- apriori (data, parameter = NULL, appearance = NULL, control = NULL)

Аргументы

data	объект класса transactions или любая другая структура, которую можно преобразовать в данный класс (например, бинарная матрица)
parameter	объект класса ***APparameter*** или именованный список; по умолчанию - *support* = 0.1, *confidence* = 0.8, *maxlen* = 10
appearance	объект класса ***APappearance*** или именованный список; ограничивает появление признака с данным аргументом, по умолчанию - появление всех признаков не ограничено
control	объект класса ***APcontrol*** или именованный список; контролирует выполнение алгоритма

Детали
Создаёт только признаки в *RHS*.

возвращает объект класса *rules* или *itemsets*

Пример

```
data (“Adult”)
## mine association rules
rules <- apriori (Adult, parameter = list (supp = 0.5, conf = 0.9, target = “rules”))
summary (rules)
```

eclat

анализирует частые наборы алгоритмом Eclat

- eclat (data, parameter = NULL, control = NULL)

Аргументы

data	объект класса *transactions* или любая другая структура, которую можно преобразовать в данный класс
parameter	объект класса ***ECparameter*** или именованный список; по умолчанию - *support* = 0.1, *maxlen* = 5
control	объект класса ***ECcontrol*** или именованный список; контролирует выполнение алгоритма

Детали
Т.к. хранение tid-списков затратно по памяти, то они работают только с минимальными значениями поддержки, из-за чего множество наборов очень мало.
При возникновении проблем с распределением памяти выдаётся ошибка сегментации.

возвращает объект класса *itemsets*

Пример
```
data (“Adult”)
## mine itemsets with minimum support of 0.1.
itemsets <- eclat (Adult, parameter = list (supp = 0.1, maxlen = 15))
```

Вспомогательные функции.

support

вычисляет значение поддержки для выбранных транзакций базы данных

- support(x, transactions, ...)
- ## method for signature ’itemMatrix’ or ’associations’
  support(x, transactions, type= c("relative", "absolute"), control = NULL)

Аргументы

data	множество наборов, поддержку которых надо вычислить
...	пропуск последующих аргументов
type	строка “relative” (по умолчанию) или “absolute” соответствует вычислению относительной или абсолютной поддержки
control	именованный список с элементами *method* с указанием метода ("tidlists" или "ptree"), логические аргументы *reduce* и *verbose*, показывающие удалены ли неиспользуемые признаки и нужен ли подробный вывод

Детали
Поддержка наборов рассчитывается с учётом минимальной поддержки множества. Однако, если необходимо подсчитать поддержку небольшого числа элементов, она не требует анализа всей базы данных.
Если выбранный параметр метода *control method* = “ptree”, то счётчики для частых наборов будут представлять префиксное дерево. Транзакции последовательно обрабатываются и увеличивают соответствующий счётчик. Это метод используется по умолчанию.
Если выбранный параметр метода *control method* = “tidlists”, то поддержка подсчитывается с использованием пересечения tid-списков. Поддержка рассчитывается отдельно для каждого набора.
Если выбранный параметр метода *control reduce* = “tidlists”, то неиспользуемое наборы удаляются из данных ещё до генерации правил. Это может работать медленно для большого числа транзакций данных.

возвращает числовой вектор длины x, содержащий значения поддержек для множества x

Пример

```
data (“Income”)
## find frequent itemsets
itemsets <- eclat (Income)[1:5]
## inspect the support returned by eclat
inspect (itemsets)
## count support in the database
support(items(itemsets), Income)
```

ruleInduction

генерирует все правила из заданных наборов

- ruleInduction(x, ...)
- ## method for signature ’itemsets’
  ruleInduction (x, transactions, confidence = 0.8, control = NULL)

Аргументы

x	множество наборов, правила которых необходимо вычислить
...	пропуск последующих аргументов
transactions	множество транзакций данных для анализа наборов
confidence	число; минимальное значение значимости правил
control	именованный список; указывает метод построения правил: "apriori" или "ptree", и содержит логические параметры *reduce* и *verbose*, показывающие удалены ли неиспользуемые признаки и нужен ли подробный вывод

возвращает объект класса *rules*

Пример

```
data (“Adult”)
## find all closed frequent itemsets
closed <- apriori (Adult, parameter = list(target = “closed”, support = 0.4))
## rule induction
rules <- ruleInduction (closed, Adult, control = list (verbose = TRUE))
summary (rules)
## inspect the resulting rules
inspect (SORT (rules, by = “lift”)[1:5])
## use lattice of frequent itemsets
ec <- eclat(Adult, parameter = list(support = 0.4))
rec <- ruleInduction(ec)
inspect (rec[1:5])
```

sample

генерирует выборку фиксированного размера из элементов x с перестановками и без

- sample(x, size, replace = FALSE, prob = NULL, ...)

Аргументы

x	множество *associations* или *transactions*
...	пропуск последующих аргументов
size	размер генерируемой выборки
replace	логический тип; отвечает за наличие перестановок
prob	числовой вектор с вероятностными весами

возвращает объект типа x

Пример

```
data(“Adult”)
## sample with replacement
s <- sample(Adult, 500, replace = TRUE)
```

random.transactions

генерирует случайный объект *transactions*, используя различные методы

- random.transactions(nItems, nTrans, method = "independent", ..., verbose = FALSE)

Аргументы

nItems	целое число; количества признаков
nTrans	целое число; количества транзакций
size	размер генерируемой выборки
method	название метода: "independent" или "agrawal"
...	пропуск последующих аргументов
verbose	логический тип; нужен ли вывод отчёта по выполнению функции

Детали
Функция генерирует *nItems* раз *nTrans* транзакций данных.

возвращает объект класса *transactions*

Пример

```
## generate random 1000 transactions for 200 items with
## a success probability decreasing from 0.2 to 0.0001
trans <- random.transactions(nItems = 200, nTrans = 1000, iProb = seq(0.2,0.0001, length=200))

## use the method “agrawal”
patterns <- random.patterns(nItems = 200)
trans2 <- random.transactions(nItems = 200, nTrans = 1000, method = “agrawal”, patterns = patterns)
```

interestMeasure

генерирует выборку фиксированного размера из элементов x с перестановками и без

- interestMeasure(x, method, transactions = NULL, reuse = TRUE, ...)

Аргументы

x	множество наборов или правил
method	строка или вектор строк; одно или несколько названий интересующих мер
transactions	множество транзакций данных для анализа ассоциаций
reuse	логический тип; необходим ли информация в слоте для пересчёта мер
...	пропуск последующих аргументов

Детали
Реализованы меры:
“allConfidence”, “crossSupportRatio”, “support”, - для наборов;
“chiSquare”, “confidence”, “conviction”, “cosine”, “coverage”, “doc”, “gini”, “hyperLift”, “hyperConfidence”, “fishersExactTest”, “improvement”, “leverage”, “lift”, “oddsRatio”, “phi”, “RLD”, “support” - для правил.

возвращает числовой вектор со значениями меры для каждой ассоциации при выборе в параметрах функции одного типа меры и *data.frame* со всеми мерами для каждого элемента - в случае выбора нескольких мер

Пример

```
data (“Income”)
rules <- apriori (Income)
## calculate a single measure
hyperConfidence = interestMeasure(rules, method = “hyperConfidence”, Income)
inspect(head(SORT(rules, by = “hyperConfidence”)))
## calculate several measures
m <- interestMeasure(rules, c(“confidence”, “oddsRatio”, “leverage”), Income)
inspect(head(rules))
```

dissimilarity

вычисляет расстояния между векторами бинарной матрицы: *transactions* или *associations*

- dissimilarity(x, y = NULL, method = NULL, args = NULL, ...)
- ## method for signature ’itemMatrix’ or ’associations’
  dissimilarity(x, y = NULL, method = NULL, args = NULL, which = "transactions")
- ## method for signature ’matrix’
  dissimilarity(x, y = NULL, method = NULL, args = NULL)

Аргументы

x	множество элементов
y	*NULL* или второе множество для вычисления расстояний
method	строка - метод, определяющая понятие расстояния; одно из “affinity”, “cosine”, “dice”, “euclidean”, “jacard” (по умолчанию), “matching”, “pearson”; дополнительно для ассоциаций - “toivonen”, “gupta”
args	список аргументов соответствующего метода
...	пропуск последующих аргументов
which	строка, определяющая будут ли расстояния вычислены между транзакциями (по умолчанию) или бинарными признаками (“items”)

возвращает объект класса dist

Пример

```
## cluster items in Groceries with support > 5%
data (“Groceries”)
s <- Groceries [,itemFrequency(Groceries)>0.05]
d_jaccard <- dissimilarity (s, which =”items”)

## calculate Jaccard distances
data (“Adult”)
d_jaccard <- dissimilarity (sample (Adult, 200))

## calculate affinity-based distances and do hclust
s <- sample (Adult, 200)
d_affinity <- dissimilarity (s, method = “affinity”)

## cluster rules
rules <- apriori (Adult, parameter = list(support = 0.3))
rules <- subset (rules, subset = lift > 2)
## use affinity
## we need to supply the item affinities from the dataset (sample)
d_affinity <- dissimilarity (rules, method = “affinity”, args = list (affinity = affinity(s)))

## use gupta
d_gupta <- dissimilarity (rules, method = “gupta”, args = list (trans = Adult))
```

Так же в пакете реализованы функции проверок на подмножество, надмножество, максимальное и объемлющее множество: соответственно ***is.subset()***, ***is.superset()***, ***is.maximal()***, ***is.closed()***.

Пример

Рассмотрим задачу предобработки и анализа анкетных данных. Для этого возьмём базу Adult, содержащуюся в пакете “arules”.

```
> data("AdultUCI")
> dim(AdultUCI)
[1] 48842    15
> AdultUCI[1,]
  age workclass fnlwgt education education-num marital-status   occupation
1  39 State-gov  77516 Bachelors            13  Never-married Adm-clerical
   relationship  race  sex capital-gain capital-loss hours-per-week
1 Not-in-family White Male         2174            0             40
  native-country income
1  United-States  small
```

Это маркетинговые данные 48842 человек с признаками: возраст, работа, образование и т.д. - всего 14 характеристик. Изначально данные собирались с целью прогнозирования уровня дохода. 15ым признаком присоединён доход - *small*(не больше 50000 долларов) или *large*(не больше 50000 долларов).

```
> itemFrequencyPlot(Adult, support = 0.1, cex.names=0.8)
>
```

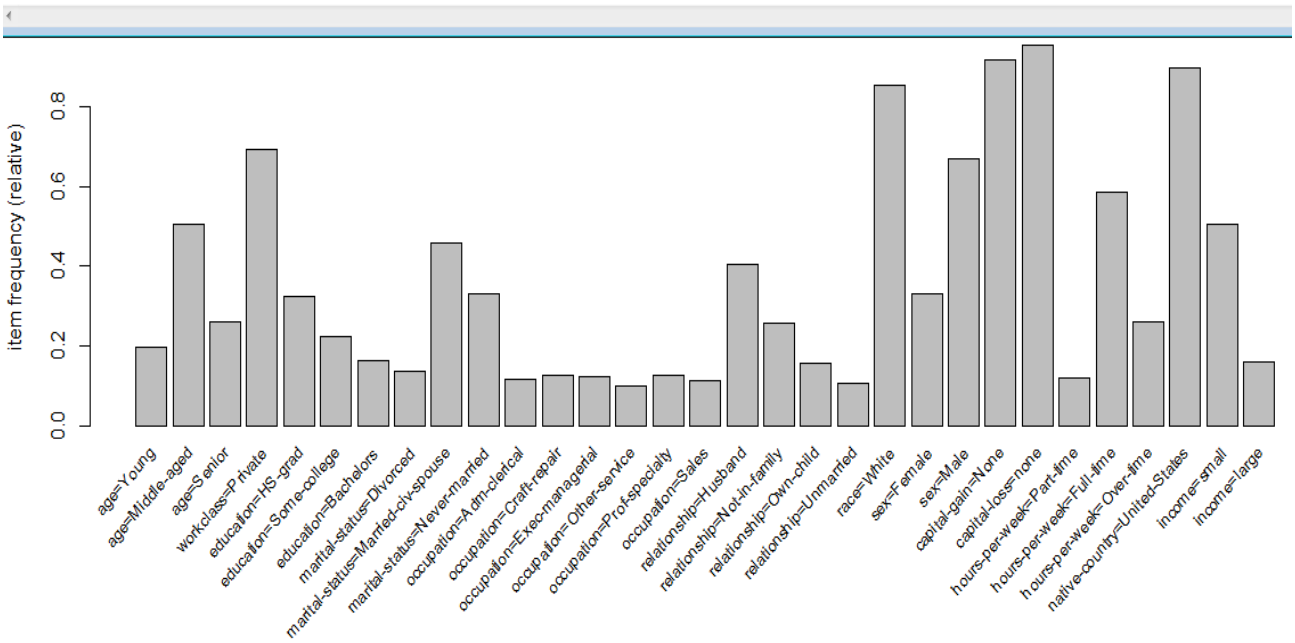

Рисунок 2: Визуализация бинаризованных данных.

Т.к. база состоит из смеси категориальных и целых признаков, то необходимо преобразовать их до корректного для данного пакета вида - до бинарной матрицы. Сначала - удалить неинтересующие и пронумеровать остальные (возможно, разбив на категории):

```
> AdultUCI[["fnlwgt"]] <- NULL
> AdultUCI[["education-num"]] <- NULL
> AdultUCI[[ "age"]] <- ordered(cut(AdultUCI[[ "age"]], c(15,25,45,65,100)),
+ labels = c("Young", "Middle-aged", "Senior", "Old"))
> AdultUCI[[ "hours-per-week"]] <- ordered(cut(AdultUCI[[ "hours-per-week"]],
+ c(0,25,40,60,168)),
+ labels = c("Part-time", "Full-time", "Over-time", "Workaholic"))
> AdultUCI[[ "capital-gain"]] <- ordered(cut(AdultUCI[[ "capital-gain"]],
+ c(-Inf,0,median(AdultUCI[[ "capital-gain"]][AdultUCI[[ "capital-gain"]]>0]),
+ Inf)),
+ labels = c("None", "Low", "High"))
> AdultUCI[[ "capital-loss"]] <- ordered(cut(AdultUCI[[ "capital-loss"]],
+ c(-Inf,0,
+   median(AdultUCI[[ "capital-loss"]][AdultUCI[[ "capital-loss"]]>0]),Inf)),
+ labels = c("none", "low", "high"))
```

потом бинаризовать всю базу:

```
> Adult <- as(AdultUCI, "transactions")
> Adult
transactions in sparse format with
 48842 transactions (rows) and
 115 items (columns)
```

В конечном итоге получим распределение, изображённое на *Рисунке 2*.
Для анализа данных используем алгоритм Apriori с минимальной поддержкой 0.1 и значимостью 0.6:

```
> itemFrequencyPlot(Adult, support = 0.1, cex.names=0.8)
> rules <- apriori(Adult,
+                  parameter = list(support = 0.01, confidence = 0.6))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen target   ext
        0.6    0.1    1 none FALSE            TRUE    0.01      1     10  rules FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[115 item(s), 48842 transaction(s)] done [0.10s].
sorting and recoding items ... [67 item(s)] done [0.02s].
creating transaction tree ... done [0.08s].
checking subsets of size 1 2 3 4 5 6 7 8 9 10 done [2.33s].
writing ... [276443 rule(s)] done [0.15s].
creating S4 object  ... done [0.58s].
```

В результате работы алгоритм вывел 276443 правила:

```
> summary(rules)
set of 276443 rules

rule length distribution (lhs + rhs):sizes
    1     2     3     4     5     6     7     8     9    10
    6   432  4981 22127 52669 75104 67198 38094 13244  2588

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   5.000   6.000   6.289   7.000  10.000

summary of quality measures:
    support          confidence          lift
 Min.   :0.01001   Min.   :0.6000   Min.   : 0.7171
 1st Qu.:0.01253   1st Qu.:0.7691   1st Qu.: 1.0100
 Median :0.01701   Median :0.9051   Median : 1.0554
 Mean   :0.02679   Mean   :0.8600   Mean   : 1.3109
 3rd Qu.:0.02741   3rd Qu.:0.9542   3rd Qu.: 1.2980
 Max.   :0.95328   Max.   :1.0000   Max.   :20.6826

mining info:
  data ntransactions support confidence
 Adult         48842    0.01        0.6
```

В дополнение к получившимся результатам алгоритма можно посмотреть на наиболее достоверные правила, определяющие уровень дохода:

```
> rulesIncomeSmall <- subset(rules, subset = rhs %in% "income=small" & lift > 1.2)
> rulesIncomeLarge <- subset(rules, subset = rhs %in% "income=large" & lift > 1.2)
> inspect(head(SORT(rulesIncomeSmall, by = "confidence"), n = 3))
  lhs                                rhs                support confidence     lift
1 {workclass=Private,
   marital-status=Never-married,
   relationship=Own-child,
   sex=Male,
   hours-per-week=Part-time,
   native-country=United-States} => {income=small} 0.01074895  0.7104195 1.403653
2 {workclass=Private,
   marital-status=Never-married,
   relationship=Own-child,
   sex=Male,
   hours-per-week=Part-time}     => {income=small} 0.01144507  0.7102922 1.403402
3 {workclass=Private,
   marital-status=Never-married,
   relationship=Own-child,
   sex=Male,
   capital-gain=None,
   hours-per-week=Part-time,
   native-country=United-States} => {income=small} 0.01046231  0.7097222 1.402276
```

```
> inspect(head(SORT(rulesIncomeLarge, by = "confidence"), n = 3))
  lhs                                      rhs               support confidence     lift
1 {marital-status=Married-civ-spouse,
   capital-gain=High,
   native-country=United-States}      => {income=large} 0.01562180  0.6849192 4.266398
2 {marital-status=Married-civ-spouse,
   capital-gain=High,
   capital-loss=none,
   native-country=United-States}      => {income=large} 0.01562180  0.6849192 4.266398
3 {relationship=Husband,
   race=White,
   capital-gain=High,
   native-country=United-States}      => {income=large} 0.01302158  0.6846071 4.264454
```

Последним шагов запишем полученные правила на диск:

```
> WRITE(rulesIncomeSmall, file = "data.csv", sep = ",", col.names = NA)
```