WIRTSCHAFTSUNIVERSITÄT WIEN

DIPLOMARBEIT

Titel der Diplomarbeit:
Visualisierung von Assoziationsregeln mit R

Verfasserin/Verfasser: Martin Vodenicharov

Matrikel-Nr.: 0253795

Studienrichtung: Betriebswirtschaft

Beurteilerin/Beurteiler: Prov.Doz. Dr. Michael Hahsler

Ich versichere:

dass ich die Diplomarbeit selbstständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfe bedient habe.

dass ich dieses Diplomarbeitsthema bisher weder im In- noch im Ausland (einer Beurteilerin/ einem Beurteiler zur Begutachtung) in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

dass diese Arbeit mit der vom Begutachter beurteilten Arbeit übereinstimmt.

_____ _____

Datum Unterschrift

Visualisierung von Assoziationsregeln mit R

Visualisation of Assotiation Rules with R

Eingereicht von

Martin Vodenicharov

Studienkennzahl J151

Matrikelnummer: 0253795

Diplomarbeit

am Institut für Informationswirtschaft

Wirtschaftsuniversität Wien

Studienrichtung: Betriebswirtschaft

Begutachter: Prov.Doz. Dr. Michael Hahsler

Wien, 29. Juni 2007

Zusammenfassung

Assoziationsregeln sind heutzutage ein weit verbreitetes Instrument, um bestehende Daten zu analysieren. Dadurch lassen sich bis jetzt unbekannte Muster entdecken. Auf der anderen Seite sollen die Muster so präsentiert werden, dass man daraus nützliche Maßnahmen ableiten kann. Das Ziel dieser Arbeit ist es, Assoziationsregeln und deren Visualisierungsmöglichkeiten vorzustellen. Mit Hilfe der Programmiersprache `R` werden Assoziationsregeln gebildet und präsentiert. Dabei werden unter anderem die Pakete `arules`, `vcd` und `Rgraphviz` dazu benutzt, um Assoziationsregeln zu generieren und zu präsentieren. Es werden auch andere Pakete verwendet.

Schlagwörter: Assoziationsregeln, Visualisierung, R, Itemsets, Mosaik Plots, Doubledecker Plots, gerichtete Graphen

Abstract

Mining association rules is very common technique to analyse given data and to find patterns, that were unknown before. These patterns should be presented in proper manner, so that one can derive helpfully actions. The purpose of this paper is to present Association rules in visual form. With the help of the programming language `R` rules are generated und presented using the packages `arules`, `vcd` and `Rgraphviz` and other packages.

Keywords: Association rules, visualizing, R, itemsets, mosaiplots, doubledecker plots, directed graphs

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einführung

1.1 Einleitung und Motivation

In unserer Gesellschaft nimmt die Bedeutung der Information immer stärker zu. Dazu trägt auch die rasante Entwicklung der neuen Technologien bei. Aus diesem Grund ist es heutzutage möglich mittels rechnergestützter Systeme umfassende Datenbestände zu erfassen. Hinter dieser Datenbestände verbirgt sich eine große Menge an Wissen, das allerdings nicht so leicht zu gewinnen ist. Deswegen benötigt man Methoden, mit denen man diese Information extrahieren kann. Die Assoziationsanalyse ist eine Methode zur Wissensgewinnung. Dabei wird versucht in großen Datenbestände Muster zu erkennen. Das meist bekannte Beispiel dafür ist die Warenkorbanalyse. Eine Aussage aus der Warenkorbanalyse könnte etwa so lauten: *90 % der Leute, die Bier und Chips gekauft haben, haben auch Nüsse gekauft.*

Das Ziel dieser Arbeit ist es einen Überblick über das Thema Assoziationsregeln und Visualisierung zu geben und dann in weiterer Folge einige Visualisierungstechniken in Programmiersprache R zu implementieren. Zu diesem Zweck werden zuerst Assoziationsregeln vorgestellt. Da es nicht immer so einfach ist das neu gewonnene Wissen zu überschauen, wird mit Hilfe von Visualisierungstechniken versucht, die gewonnenen Assoziationsregeln am besten zu analysieren. Als praktischer Teil dieser Arbeit sollte ein Kapitel dienen, wo eine oder mehrere Visualisierungstechniken mit Hilfe der Programmiersprache R implementiert werden.

1.2 Problemstellung

Assoziationsregeln und Visualisierungstechniken scheinen zwei verschiedene Begriffe zu sein. Dennoch haben Assoziationsregeln fast keinen Nutzen, wenn man sie nicht analysieren und präsentieren kann. Nun stellen sich in diesem Zusammenhang einige Fragen:

- Gibt es geeignete Visualisierungstechniken um eine und/oder mehrere Assoziationsregeln darzustellen?
- Wie gut lassen sich Assoziationsregeln mit Hilfe von Visualisierungstechniken darstellen?

Aus diesen Fragen lässt sich eine bilden, die für diese Arbeit von essenzieller Bedeutung ist:

- Gibt es eine oder mehrere Visualisierungstechniken, die sich mit Hilfe von Programmiersprache R implementieren lassen, um Assoziationsregeln wirkungsvoll darstellen zu können?

1.3 Aufbau der Arbeit

Die Arbeit besteht aus fünf großen Kapiteln. Diese sind neben der Einführung noch Assoziationsregeln, Visualisierungstechniken und Assoziationsregeln, Visualisierungstechniken mit R und Zusammenfassung. Im ersten Kapitel wird noch die Forschungsfrage formuliert.

Im Kapitel Assoziationsregeln werden diese vorgestellt. Dabei wird versucht diese strukturiert darzustellen. Wie sie entstanden sind, welchen Nutzen man hat und was besonders wichtig dabei ist, sind die Merkmale einer Assoziationsregel. Neben dieser Begriffe werden auch Algorithmen zur Generierung von Assoziationsregeln vorgestellt. Das Ziel dabei ist zu verstehen, wie eine Regel zu Stande kommt.

Im Kapitel Visualisierungstechniken und Assoziationsregeln werden verschiedene Möglichkeiten vorgestellt, wie die gewonnene Information am besten präsentiert werden kann.

Im Kapitel Visualisierungstechniken mit R wird zuerst die Programmiersprache R ganz kurz vorgestellt. Dann werden zwei sehr wichtige Pakete, `arules` und `vcd`, präsentiert. Es werden auch andere für den Zweck dieser Arbeit nützlichen Paketen vorgestellt. Danach werden eine oder mehrere Visualisierungstechniken implementiert, wobei die Pakete `arules` und `vcd` die Grundlage dafür bilden. Zum Schluss gibt es ein Kapitel, wo die Erkenntnisse dieser Arbeit zusammengefasst werden.

Kapitel 2

Assoziationsregeln

In diesem Kapitel wird zuerst die Entstehung der Assoziationsregeln diskutiert. Dabei werden die Assoziationsregeln als ein Werkzeug in Data Mining-Prozess dargestellt. Data Mining ist aber auch nur ein Teil von Knowledge Discovery in Databases. In diesem Zusammenhang wird der Nutzen von Data Mining und Assoziationsregeln im engen Sinne näher erläutert. Danach wird eine formale Definition der Assoziationsregeln in Anlehnung an [1] geliefert. Es folgt ein Abschnitt, bei dem es um die verschiedenen Merkmale einer Assoziationsregel geht. Wenn diese Merkmale uns schon bekannt sind werden einige Arten von Assoziationsregeln präsentiert. Zum Schluss werden verschiedene Algorithmen zur Generierung von Assoziationsregeln diskutiert, wobei der Schwerpunkt auf die Arbeit von Agrawal et al. [1] liegt.

2.1 Entstehung und Einordnung

Bevor wir uns mit den Assoziationsregeln im engen Sinne befassen, ist es sinnvoll den Begriff *Data Mining* zu definieren. Das würde uns helfen sie besser zu verstehen. An dieser Stelle lässt sich das Zitat von Petersohn einführen [28]:

> „Auf dem Weg zur Erschließung von Wettbewerbsvorteilen verwenden Entscheidungsträger immer stärker modernste Informationstechnologien. Die intensive Nutzung von Daten nimmt dabei eine Schlüsselposition ein. Eine systematische und schnelle Beschaffung, Verwaltung, Bereitstellung, Analyse und Interpretation von Daten liefert Information, die derzeit als die unternehmerische Ressource angesehen wird."

Diesem Zitat kann man einige Begriffe entnehmen, die für die Definition von *Data Mining* von Bedeutung sind. Das sind die systematische und schnelle Beschaffung, Verwaltung und Bereitstellung. Ganz grob beschrieben steht *Data Mining* für Datenabbau, Datengewinnung. Ähnlich dem Bergbau, wo Kohle gewonnen werden, ist unter *Data Mining* die Gewinnung von interessanten Daten zu verstehen.

Eine sehr kurze und intuitive Definition findet man in [34]: *Data Mining* ist ein Prozess für automatisierte Entdeckung von nützlicher Information in großen Datenbeständen. Das alles wurde möglich, weil

1. heute eine Firma oder ein Unternehmen viel mehr Daten sammeln können, als dies vor 20 oder 30 Jahren der Fall war und
2. diese Daten mit Hilfe von komplexen Algorithmen analysiert werden können, um bis jetzt unbekannte Muster zu erkennen.

Den Algorithmen ist ein Kapitel gewidmet, wo ausführlicher darüber diskutiert wird.

Manchmal wird irrtümlicherweise der Begriff *Data Mining* mit dem Begriff *Knowledge Discovery in Databases* (KDD) gleich gesetzt. Allerdings ist Data Mining ein Teil von Knowledge Discovery in Databases [34]. Wie in Abbildung 2.1 zu sehen ist, steht *Data Mining* mitten im Prozess der Knowledge Discovery in Databases. Die ursprünglichen Daten werden zuerst für den Prozess der Data Mining vorbereitet. Dies bedeutet, dass verschiedene Datensätze zusammengefügt werden, doppelte Beobachtungen werden bereinigt und bestimmte für den Zweck der Analyse Aufzeichnungen werden ausgewählt [34].

Jiawei Han und Micheline Kamber sehen das Problem mit KDD und Data Mining fast genau so. Für sie ist Data Mining ein Prozess des *Herausziehens oder Abbau* des Wissens aus sehr großen Datenbeständen. Wir sind an den letzten Schritten dieses Prozesses interessiert (siehe Abbildung 2.1).

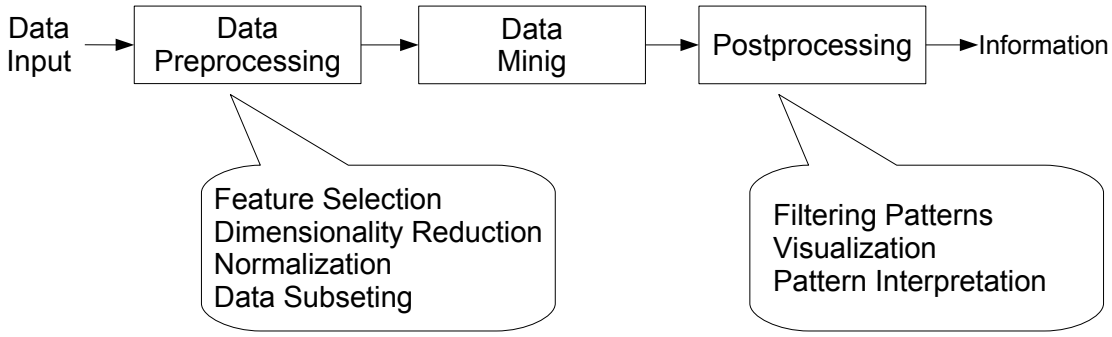


Abbildung 2.1: Data Mining und KDD
nach Tan et al. [34]

Nämlich:

1. Data Mining – essenzieller Prozess, wo intelligente Methoden angewendet werden, um Muster aus einer Datenmenge zu extrahieren.
2. Filtering Patterns - wirklich interessante Muster, die Wissen darstellen, erkennen und zwar basierend auf bestimmte Merkmale (siehe Abschnitt 2.4)
3. Visualization – hier werden Visualisierungstechniken angewendet, um das gewonnene Wissen zu präsentieren. (siehe Abschnitt 3.1)

Alle dieser Definitionen haben etwas Gemeinsames (Abbau, Gewinnung und *Herausziehen*). Folgende Definition soll aber der Prozess Data Mining zum Ausdruck bringen:

> „Data Mining beinhaltet die Extraktion und Entdeckung von implizitem, bisher nicht bekanntem und potentiell nützlichem Wissen aus Daten.“

Dieses Zitat stammt aus [29] und wurde auch in [28] verwendet.

Assoziationsregeln lassen sich als eine der Methoden von *Data Mining*. Neben der Assoziationsanalyse stehen andere Werkzeuge zur Verfügung, mit deren Hilfe man Wissen gewinnen kann.

In Anlehnung an [28] können Data Mining Verfahren in folgende Klassen eingeteilt werden.

1. Klassenbildung

2. Assoziationsanalyse
3. Klassifizierung
4. Zeitreihenanalyse

Da wir uns in dieser Arbeit mit Assoziationsregeln und deren Visualisierung beschäftigen werden, werden die einzelne Verfahrensklassen nicht näher erläutert. Weitere Informationen zu den einzelnen Klassen findet man in [28]. In diesem Buch werden die Klassen samt ihrer Verfahren ausführlich dargestellt.

Fassen wir zusammen. Assoziationsregeln sind eines der Werkzeuge im Data Mining Prozess. Hauptsächlich werden sie in der Warenkorbanalyse angewendet, um bestimmte, bis jetzt unbekannte Muster zu entdecken. Allerdings finden sie auch in anderen Bereichen Verwendung. Im Abschnitt 2.2 wird diskutiert, wozu sie gut eingesetzt werden können. Woraus eine Assoziationsregel besteht kann im Abschnitt 2.4 nachgelesen werden.

2.2 Nutzen

In diesem Abschnitt geht es um den Nutzen, den man durch die Anwendung von *Data Mining* und Assoziationsregeln erzielen kann. Nahezu alle Bücher, die sich mit *Data Mining* befassen, fangen mit einem Kapitel, wo beschrieben wird, wie sehr wir auf KDD und *Data Mining* heutzutage angewiesen sind. Das hat sich so ergeben, da wir heute viel mehr Daten sammeln können, als wir bearbeiten.

Jim Gray sagt es ganz einfach im Vorwort von [11]:

> „We are deluged by data - scientific data, medical data, demographic data, financial data, and marketing data. People have no time to look at this Data."

Er kommt auf eine sehr logische und intuitive Lösung. Es muss irgendwie automatisch gemacht werden. Dazu sind schon Verfahren entwickelt worden, die das ermöglichen. *Data Mining* als Teil des KDD macht genau das. Rohe Daten aufbereiten, diese dann verarbeiten und zum Schluss präsentieren.

Um ein Beispiel zu nennen, wird Bezug auf ein Buch genommen, das vor mehr als fünfzehn Jahren erschienen ist: *Knowledge Discovery in Databases* von Gregory Piatetsky-Shapiro und William J. Frawley [29]. Es fasst die Ergebnisse eines Workshop zum Thema *Knowledge Discovery in Databases* zusammen und besteht aus den besten Artikeln dieses Workshop. Gregory Piatetsky-Shapiro, William J. Frawley und Christopher J. Matheus führen ein Beispiel ein, das oben geführtes Zitat verdeutlichen soll. Die National Aeronautics and Space Administration (NASA) ist die zivile US-Bundesbehörde für Luft- und Raumfahrt. Sie sollte während der 1990er täglich ein Terabyte Daten erzeugen. Das ist so viel, dass eine einzige Person einige Jahre damit beschäftigt sein sollte, sich die Bilder anzuschauen, die an einem Tag gemacht wurden.

Das ist aber bei weitem nicht alles. Heutzutage sind nahezu alle Supermärkte und Geschäfte mit einem Scanner an der Kasse ausgestattet, so dass jeder einzelne Einkauf registriert und in eine Datenbank abgespeichert werden kann. So entstandene Datenbanken können später mittels verschiedener Verfahren analysiert werden, um bestimmte Muster entdecken zu können. Ein kleines Beispiel soll das veranschaulichen. Nehmen wir an, dass in einem Geschäft jeder Einkauf registriert wird. Durch Anwendung verschiedener Verfahren zur Musterentdeckung (zum Beispiel Assoziationsanalyse) kommt man zu dem Schluss, dass Leute, die Bier kaufen auch Chips kaufen. Dieser *Assoziationsregel* kann wertvolles Wissen entnommen werden. Man kann Bier und Chips neben einander ins Geschäft stellen, sodass die Umsätze dieser Produkte gesteigert werden. Andererseits kann man sie so weit wie möglich auseinander nehmen, damit man zu spontanen Einkäufen angeregt wird.

Ein anderes Beispiel befindet sich in [23]. Es handelt sich dabei um ein System zur Lehrveranstaltungsanmeldung mit über vier tausend Studenten. Es wurde nach Mustern gesucht, um so die Raumeinteilung besser zu gestalten.

Neben diesen Anwendungsgebieten kann die Assoziationsanalyse auch für *Text Mining* genutzt werden. Wie bereits erwähnt wurde, werden heutzutage riesige Bestände an Information gesammelt. Zum Großteil liegt diese Information in Form von Text. „So generiert Reuters beispielsweise 27.000 Seiten an Informationen pro Sekunde und das Wachstum des Internets erreicht Werte, die noch vor einigen Jahren undenkbar gewesen wären. Die Folge eines solchen Überangebots an Information ist ein Effekt, den man als *Information Overload* bezeichnet" [13]. Das Ziel des Text Mining ist es, aus unstrukturierten Daten Informationen zu gewinnen [13]. Text Mining lässt sich folgendermaßen definieren:

> „Text Mining steht als Oberbegriff für sämtliche Methoden, mit denen sich unbekannte, aber potenziell nützliche Informationen, die implizit in großen Textsammlungen enthalten sind, auffinden lassen" [2]

„Für professionelle und kommerzielle Institutionen ist die Erfolgskomponente einer eigenen Internetpräsenz zu einer unabdingbaren Notwendigkeit geworden. Kenntnisse über das Besucherspektrum der eigenen Homepage sind dabei unerlässlich, um entscheiden zu können, ob die Seiten auch das gewünschte Zielpublikum erreichen" [13]. *Logfile Analyse* ist eine Methode, mit deren Hilfe man die Surfgewonheiten der Internetbenutzer analysieren kann. Dadurch lassen sich Antworten auf folgende Fragen finden:

- Auf welche Seite greifen Benutzer vermehrt zu?
- Auf welchen Pfaden bewegen sich die Besucher typischerweise?
- Welches sind die erfolgreichen Pfade, auf denen tatsächlich bestellt wird?
- Welche Suchwege geht ein Benutzer, der *nicht* bestellt? [17]

„Es werden allgemeingültige Regeln in Form von Assoziationsregeln gewonnen, die den Einfluss verschiedener Pfade und Teilsequenzen aufeinander beschreiben. Sind sie einmal gewonnen, ermöglichen sie es, gewisse Voraussagen treffen zu können. Das geschieht unter anderem durch eine wöchentliche Analyse der Logfiles" [17]

2.3 Definition

Sei $\Gamma = I_1, I_2, ...I_m$ eine Menge von binären Attributen, die Artikel oder Items genannt werden. Sei T eine Datenbank mit Transaktionen. Jede Transaktion t ist durch einen binären Vektor repräsentiert, wobei $t[\mathrm{k}] = 1$ wenn Artikel I_k in t vorkommt und $t[\mathrm{k}] = 0$ wenn nicht. Für jede Transaktion in der Datenbank existiert ein Tupel. Sei X eine Menge von Artikel in Γ. Wir sagen, dass eine Transaktion t *erfüllt* X, wenn für alle Artikel I_k in X, $t[\mathrm{k}] = 1$. [1]

Agrawal et al. geben in [1] eine formale Definition. Unter einer Assoziationsregel verstehen sie eine Implikation von der Form $X \Longrightarrow I_j$, wo X eine Menge von Attributen (Artikel)

in Γ ist und I_j ein Attribut von Γ ist, das in X nicht vorhanden ist. Also bedeutet dies, dass I_j und X disjunkt sind. Die Assoziationsregel $X \Longrightarrow I_j$ ist erfüllt in der Menge von Transaktionen T mit einem Konfidenzfaktor (dazu kommen wir später noch) $0 \leq \mathrm{c} \leq 1$, wenn mindestens c % von den Transaktionen in T, die X erfüllen auch I_j erfüllen. [1]

Ein Beispiel soll an dieser Stelle dies veranschaulichen: *90 % der Leute, die Bier und Chips gekauft haben, haben auch Nüsse gekauft.*

Nehmen wir mal an, dass unsere drei Produkte (Bier, Chips und Nüsse), die einzige drei sind, die man in einem Geschäft erwerben kann. Das entspricht Γ. T wäre unsere Datenbank, wo alle Transaktionen t gespeichert sind. I_1 ist Bier, I_2 Chips und I_3 Nüsse. Eine Transaktion t könnte so aussehen: Bier und Chips. Dann würde der Vektor dieser Transaktion so aussehen: t (1,1,0). X ist in unserem Beispiel *Bier und Chips* und I_j sind die Nüsse.

Bei der Gewinnung von Assoziationsregeln treten zwei Problemen auf [12]. Das erste ist die Komplexität des Algorithmus. Die Anzahl der gefundenen Regeln steigt exponentiell mit steigender Anzahl an Attributen. Das zweite Problem ist direkt mit dem Thema dieser Arbeit verbunden. Die interessante Regeln müssen aus einer Menge an generierten Assoziationen herausgenommen werden. Wie in [12] hingewiesen wird, kann das ganz aufwendig sein, denn mehrere tausende Regeln sind nicht ungewöhnlich.

2.4 Bestandteile und Merkmale einer Assoziationsregel

In diesem Abschnitt werden die verschiedenen Merkmale einer Assoziationsregel dargestellt. Je nachdem, was man beschreiben möchte gibt es hauptsächlich zwei Kategorien. Diese lassen sich durch zwei einfache Fragen definieren:

1. Woraus besteht eine Assoziationsregel?
2. Wie „gut" ist eine Assoziationsregel?

Die erste Frage bezieht sich auf die Bestandteile einer Assoziationsregel. Zur Erinnerung: Eine Assoziationsregel ist eine Implikation von der Form $A \Rightarrow B$, d.h. wenn A dann B, wo A und B zwei Itemmengen sind [13]. Die zweite Frage soll die Güte einer Regel beschreiben. Da alle Bestandteile und Merkmale eng mit einander verbunden sind ist es essentiell sie gleichzeitig darzustellen.

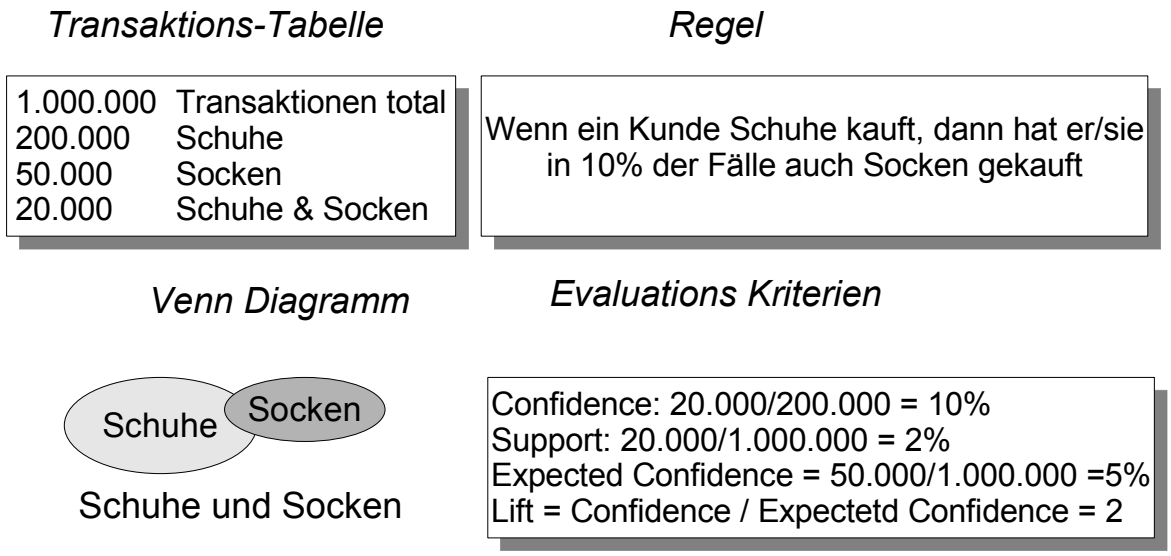


Abbildung 2.2: Assoziationsregel
in Anlehnung an [13]

Abbildung 2.2 zeigt eine Assoziationsregel.

Es sei die Regel *„Wenn ein Kunde Schuhe kauft, dann hat er/sie in 10% der Fällen auch Socken gekauft"* gegeben. In diesem Beispiel sind die *Schuhe* der so genannte *Regelrumpf* (wird auch Prämisse genannt). Auf der anderen Seite der Regel steht der *Regelkopf* (Konklusion). Hier sind *Socken* der Regelkopf [13].

In diesem Beispiel bestehen *Regelrumpf* und *Regelkopf* jeweils aus nur einem Item (Schuhe bzw. Socken). Es kann durchaus möglich sein, dass das nicht der Fall ist. Es ist sogar so, dass in den meisten Fällen der Regelrumpf aus mehreren Items besteht. Was für Assoziationsregeln es geben kann, erfahren wir im Abschnitt 2.5. Im Zusammenhang mit *Regelrumpf* und *Regelkopf* muss noch etwas erwähnt werden. Beide Itemmengen müssen disjunkt sein.

In Abbildung 2.2 sind andere Merkmale, die bis jetzt nicht definiert worden sind. Das sind *support, confidence, expected confidence und lift.* Diese sollen an dieser Stelle näher erläutert werden.

Bevor wir uns aber mit diesen Begriffen befassen wollen wir zeigen, warum überhaupt solche Merkmale entstanden sind. Zu diesem Zweck lässt sich ein Zitat aus [19] einführen:

> „Paradoxically, data mining itself can produce such great amounts of data that there is a new knowledge management problem"

Also benötigen wir Interessantheitsmaße, mit denen man im Stande ist, die Menge der extrahierten Assoziationsregeln zu reduzieren. Diese Interessantheitsmaße müssen objektiv gebildet werden. Zu diesem Zweck liegt ihnen eine Statistik zugrunde [11]. Des Weiteren kann eine Referenzschwelle durch den Anwender angegeben werden. Assoziationsregeln, die diese Schwelle nicht erreichen, werden als uninteressant betrachtet und sind daher nicht relevant [11].

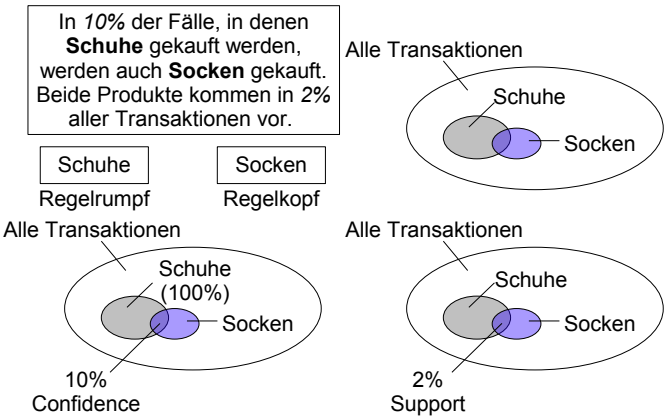


Abbildung 2.3: Merkmale einer Assoziationsregel
in Anlehnung an [13]

An dieser Stelle lässt sich eine Abbildung einführen (siehe Abbildung 2.3). Diese zeigt einen Überblick über alle Teile einer Assoziationsregel und bezieht sich zudem auf unser Beispiel (siehe Abbildung 2.2)

2.4.1 Support

Support beschreibt die Wahrscheinlichkeit, dass sowohl Regelrumpf als auch Regelkopf in einer Transaktion gemessen an allen Transaktionen auftreten. In unserem Beispiel beträgt der Support der Regel *Schuhe* → *Socken* 2%. Wie der Transaktionstabelle in Abbildung 2.2 zu entnehmen ist kommen gemeinsam sowohl *Schuhe* als auch *Socken* in einer Transaktion 20.000 mal vor. Gemessen an allen Transaktionen, hier 1.000.000, sind das genau 2%. Mathematisch sieht das für das oben beschriebene Beispiel so aus:

$$sup(Schuhe \rightarrow Socken) = p(Schuhe \cup Socken)$$

Der Support ist ein Maß, das die Häufigkeit beschreibt, mit der eine Regel in der Datenbank vorkommt [13]. In anderen Büchern, wie zum Beispiel in [27] findet man an Stelle von *support* doch eine andere Bezeichnung. In diesem Buch wird der Begriff *Abdeckungsgrad (coverage)* verwendet. Beide Begriffe, *support und coverage* beschreiben genau dasselbe. Wir wollen hier aber auch eine allgemein gültige Definition von *support* liefern.

$$sup(A \rightarrow B) = \frac{\mid \{t \in D \mid (A \cup B) \subseteq t\} \mid}{\mid D \mid}$$

wobei A der Regelrumpf, B der Regelkopf, t eine Transaktion und D die Transaktionsdatenbank ist [13]. Da der Support eine Wahrscheinlichkeit ist, kann er Werte zwischen 0 und 1 einnehmen. Wann eine Assoziationsregel brauchbar bzw. interessant ist, kann nicht so leicht beurteilt werden. Denn, wenn der Supportwert relativ groß ist, (ungefähr über 80%), dann könnte das bedeuten, dass die Regel einfach bekannte Tatsachen abbildet. Auf der anderen Seite können Assoziationsregeln mit relativ niedrigem Supportwert oft bis jetzt unbekannte Muster verbergen [13].

Wir fassen zusammen. Support einer Assoziationsregel misst die Nützlichkeit dieser Regel. Er wird berechnet, indem man die Anzahl der Transaktionen, in denen beide Teile der Regel vorkommen ins Verhältnis zu der Anzahl aller Transaktionen setzt. Regeln mit hohem Support spiegeln bekannte Tatsachen wieder und solche mit niedrigem können interessante Muster enthalten.

2.4.2 Confidence

Um *Confidence* beschreiben zu können, lässt sich ein Zitat an dieser Stelle einführen:

> „Die *Confidence* einer Regel ist definiert als der Anteil an Transaktionen, die Regelrumpf und Regelkopf beinhalten, an der Menge der Transaktionen, die den Regelrumpf erfüllen." [13]

Für unser Beispiel beträgt die *Confidence* 10%. Das sind die 20.000 Transaktionen, in denen sowohl *Schuhe* als auch *Socken* vorkommen, dividiert durch die 200.000 Transaktionen, in denen nur *Schuhe* vorkommen. Sie ist die bedingte Wahrscheinlichkeit von A gegeben B der Regel $A \rightarrow B$. Die formale Definition sieht so aus:

$$conf(A \rightarrow B) = \frac{\mid \{t \in D \mid (A \cup B) \subseteq t\} \mid}{\mid \{t \in D \mid A \subseteq t\} \mid} = \frac{sup(A \rightarrow B)}{sup(A)}$$

oder

$$conf(A \rightarrow B) = p(B|A)$$

Confidence soll die Gültigkeit einer Assoziationsregel messen [11]. Da man relativ viel Regeln bekommen kann, werden *Support* und *Confidence* dazu benutzt um die nicht wirklich relevanten Assoziationsregel zu kürzen.

Eine Eigenschaft dieses Interessantheitsmaßes soll an dieser Stelle noch erwähnt werden. Die *Confidence* ist resistent gegen einige Veränderungen. In Abbildung 2.4 ist die Confidence für alle vier Fälle gleich.

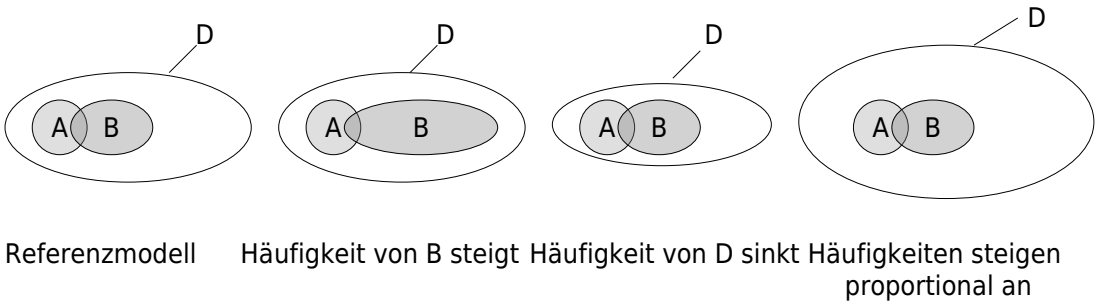


Abbildung 2.4: Konstante Confidence
in Anlehnung an [13]

Ganz links in Abbildung 2.4 steht unser Referenzmodell. Das größte Oval soll der Anzahl der Transaktionen entsprechen und ist mit D gekennzeichnet. Drinnen stellen die Figuren, die mit A und B gekennzeichnet sind, den Regelrumpf beziehungsweise den Regelkopf der

Merkmal	Anzahl	Formale Definition
Alle Transaktionen	1.000.000	D
Socken	50.000	B
Schuhe	200.000	A
Schuhe und Socken	20.000	$A \cap B$

Tabelle 2.1: Anzahl der Merkmale

Regel A $\rightarrow$ B dar. Die dunkle Fläche, die sich durch die Überschneidung von A und B ergibt, repräsentiert die *Confidence*. Jetzt wollen wir mal untersuchen, was passiert, wenn wir einige der Parameter ändern. Zu diesem Zweck beobachten wir folgende drei Fälle:

- Häufigkeit von B steigt (Anzahl der Socken steigt an)
- Häufigkeit von D sinkt (Anzahl der Transaktionen steigt an)
- Häufigkeiten steigen proportional an

Zur Erinnerung seien die einzelnen Werte aus dem obigen Beispiel (siehe Tabelle 2.1) sowie die Definition von *Confidence* noch einmal angegeben:

$$conf(Schuhe \rightarrow Socken) = \frac{sup(Schuhe \cap Socken)}{sup(Schuhe)}$$

Da weder die Anzahl der Transaktionen noch die Anzahl der Socken für die Berechnung der *Confidence* berücksichtigt werden müssen, ändert sich an *Confidence* nichts. Genau das passiert auch, wenn alle Merkmale mit einem Faktor multipliziert werden, denn diesen Faktor kann man abkürzen. Abbildung 2.4 soll das veranschaulichen.

2.4.3 Expected Confidence und Lift

Diese Interessantheitsmaße sind eng mit einander verbunden. *Lift* wird berechnet, indem man die *Confidence* einer Regel $A \rightarrow B$ durch die *Expected Confidence* dividiert [13]. Die *Expected Confidence* entspricht dem *sup(B)*. Für uns bedeutet dies, dass die *Expected Confidence* gleich 5% ist. Also 50.000 Transaktionen, in denen *Socken* vorkommen dividiert durch die Anzahl aller Transaktionen(1.000.000). Der *Lift* ergibt sich dann 10% / 5% = 2 (vgl. Abbildung 2.2).

Eine formale Definition von *Lift* findet man in [13]:

$$lift(A \rightarrow B) = \frac{conf(A \rightarrow B)}{sup(B)} = \frac{sup(A \cup B)}{sup(A)sup(B)}$$

Der *Lift* misst somit die Korrelation zwischen Regelrumpf und Regelkopf. Beträgt er mehr als 1 deutet das auf eine positive Korrelation hin. Wenn er kleiner 1 ist zeigt das eine negative Korrelation.

2.4.4 Andere Merkmale und Interessantheitsmaße

Da wir bereits gesehen haben, dass *Support* und *Confidence* allein nicht ausreichend sind, um die Güte einer Assoziationsregel zu beurteilen, müssen auch andere Interessantheitsmaße entwickelt werden.

- Das erste Maß, mit dem wir uns befassen ist die *p-s Funktion*. Ihre Namen hat sie bekommen von ihrem Erfinder, Piatetsky-Shapiro [29]. An dieser Stelle soll die formale Definition angegeben werden.

  $$p - s\ (A \rightarrow B) = sup(A \rightarrow B)\ \text{-}\ sup(A)sup(B)$$

  Der Grundgedanke dabei ist, dass der Support einer Regel höher sein sollte, als der erwartete Support bei statistischer Unabhängigkeit. Ist der Wert dieser Funktion positiv so bedeutet dies eine positive Korrelation. Wenn der Wert negativ ist deutet das auf einen negativen Zusammenhang hin [13].

- Eine andere Funktion ist die Laplace Funktion [18]. Sie wurde von Bayardo und Agrawal vorgeschlagen und sieht so aus:

  $$laplace(A \rightarrow B) = \frac{sup(A \rightarrow B) + 1}{sup(A) + k}$$

  Dabei ist k eine Konstante, deren Wert größer als 1 ist. Dieser Funktion ist der Confidence sehr ähnlich und weist daher dieselben Nachteilen auf.

- Die *Gain Funktion* ([13][S. 446]) ist der *p-s Funktion* sehr ähnlich. Sie ist folgendermaßen definiert:

  $gain(A \rightarrow B) = sup(A \rightarrow B) - \theta\ sup(A)$

  „Der Parameter θ kann dabei Werte zwischen Null und Eins annehmen. Der Wertebereich der *Gain Funktion* liegt somit zwischen -θ und 1-θ. Über Variation des Parameters können Effekte auf die optimalen Regeln beobachtet werden. Ein $gain(A \rightarrow B) = 0$ besagt, dass in jeder θ-ten Transaktion, in der A enthalten ist, auch B enthalten ist. Werte über (unter) Null symbolisieren einen stärkeren (schwächeren) Zusammenhang zwischen A und B als durch θ vorgegeben" [13]. Somit entspricht θ dem *sup(B)* von der *p-s Funktion*.

- Differences of Confidence (DOC) ist ein anderes Maß, mit dem man Assoziationsregeln vergleichen kann. Sie wurde von Hofmann et al. in [14] vorgeschlagen. Man kann messen, wie die eine Variable A in einer gegebenen Regel $A \rightarrow B$ zur Vorhersage von B beiträgt, anstatt sie mit einer zufälligen Situation zu vergleichen [14]. Eine Möglichkeit dafür ist den Unterschied der Konfidenzen für die Regeln $A \rightarrow B$ und $\neg A \rightarrow B$ zu berechnen. Wir definieren:

  $doc(A \rightarrow B) = conf(A \rightarrow B) - conf(\neg A \rightarrow B)$

  Das kann auch durch Wahrscheinlichkeiten ausgedrückt werden:

  $$doc(A \rightarrow B) = \frac{P(A \cap B) - P(A)P(B)}{P(A)P(\neg A)}$$

  Wozu dieses Maß gut ist werden wir später (siehe Abschnitt 3.4) zeigen. Um einiges vorweg zu nehmen, muss erwähnt werden, dass es dabei um Visualisierungstechniken geht und zwar um die so genannten *Doubledecker Plots*. Damit kann man Assoziationsregeln mit mehreren Items im Regelrumpf besser analysieren. Was uns an dieser Stelle interessiert, ist, wann eine Regel gemessen an *Differences of Confidence* gut ist. Diese Kennzahl kann Werte zwischen -1 und 1 annehmen. Umso größer der Wert ist desto stärker eine Assoziationsregel.

2.5 Arten

Da wir bereits wissen, woraus eine Assoziationsregel besteht, können wir nun einigen Arten von Regeln näher erläutern. Zu diesem Zweck werden hier die wichtigsten Attribute einer Assoziationsregel noch einmal erwähnt:

- Items sind uninterpretierbare, diskrete Dinge oder Entitäten. Werden auch Attribute genannt [13].
- Itemmenge ist die Menge alle Items.
- Regelrumpf ist die linke Seite einer Regel. Wird auch noch Prämisse genannt und besteht aus einem oder mehreren Items.
- Regelkopf ist die rechte Seite einer Regel. Wird auch Konklusion genannt und besteht aus einem oder mehreren Items.

In Anlehnung an Han et al. [11] werden jetzt verschiedene Arten von Assoziationsregeln vorgestellt.

- Ausgehend von der Art der Werte:
  - Wenn eine Regel Assoziationen zwischen Präsenz oder Abwesenheit von Items betrifft, ist das eine *Boolesche Assoziationsregel*. Die Regel *Bier* $\rightarrow$ *Chips* ist eine Boolesche Assoziationsregel.
  - Wenn eine Regel Zusammenhänge zwischen quantitativen Items oder Attributen beschreibt, dann ist sie eine *quantitative Assoziationsregel*. In solchen Regeln werden die quantitativen Werte der Items oder Attributen partitioniert. Die Regel *Alter (30 ... 39)* $\cap$ *Einkommen (42 ... 49)* $\rightarrow$ *kauft (Fernseher)* ist ein Beispiel dafür.
- Ausgehend von der Dimensionen der Daten unterscheiden Han et al. zwischen:
  - eindimensionale Assoziationsregeln. Wir können die Regel *Bier* $\rightarrow$ *Chips* so umschreiben *kauft (Bier)* $\rightarrow$ *kauft (Chips)*. Die Dimension hier ist nur eine *kauft*.
  - mehrdimensionale Assoziationsregeln. Ein Beispiel ist die Regel von vorhin. *Alter (30 ... 39)* $\cap$ *Einkommen (42 ... 49)* $\rightarrow$ *kauft(Fernseher)*. Dabei sind *Alter, Einkommen* und *Kaufen* die einzelnen Dimensionen.
- Ausgehend von dem Niveau der Abstraktion. Es gibt Methoden, die Assoziationsregeln auf unterschiedlichem Abstraktionsniveau finden können. Um dies zu veranschaulichen seien zwei Regeln gegeben:

$$\{Alter\ (30...39)\} \rightarrow \{kauft\ (LaptopComputer)\ \}$$
$$\{Alter\ (30...39)\} \rightarrow \{kauft\ (Computer)\ \}$$

In diesem Fall ist *Computer* eine höhere Abstraktionsebene von *Laptop Computer*. Solche Regeln werden *multi-level Assoziationsregeln* genannt. Wenn aber die Regeln, die innerhalb eines Sets generiert wurden, keinen Bezug auf unterschiedliche Abstraktionsebenen nehmen, werden sie *single-level Assoziationsregeln* genannt.

2.6 Algorithmen zur Generierung von Assoziationsregeln

Die Algorithmen zur Generierung von Assoziationsregeln sind aus dem Bedürfnis heraus entstanden, dass heutzutage die große Unternehmen riesige Mengen an Daten sammeln können. Diese Daten müssen mittels Datenverarbeitungsprogramme analysiert werden, um bis jetzt unbekannte Muster und Erkenntnisse zu gewinnen. Eine Methode, die sich ganz hervorragend dafür bietet, ist die Assoziationsanalyse.

Allerdings können durch eine Assoziationsanalyse so viel Regeln generiert werden, sodass es einem Anwender nicht leichter fällt, brauchbare Regeln zu finden. Tabelle 2.2 zeigt, wie sich die Anzahl der möglichen Kombinationen mit x Items in einer Menge von 100 Items ändert.

Länge der Itemsets	Anzahl an Kombinationen
1	100
2	4.950
3	161.700
4	3.921.225
5	75.287.520
6	1.192.052.400
7	16.007.560.800
8	186.087.894.300

Tabelle 2.2: Anzahl möglicher Kombinationen
in Anlehnung an Berry et al. [3]

Wie man in Tabelle 2.2 sieht, steigt die Anzahl der möglichen Kombinationen exponentiell. Und das bei nur 100 Items. Man stelle sich ein großes Geschäft vor. Dort werden deutlich mehr als 100 Produkte in seinen Regalen zu finden sein. Dann hätte das für die Komplexität eines Algorithmus zur Generierung von Assoziationsregel einen erheblichen Einfluss. Um dieses Problem lösen zu können, haben Agrawal et al. in [1] einen Algorithmus vorgeschlagen, der basierend auf Merkmale wie *Support* (siehe Abschnitt 2.4.1) und *Confidence* (siehe Abschnitt 2.4.2) schon bei der Generierung die Menge der Regeln reduziert.

TID	Items
1	Brot, Milch
2	Brot, Windeln, Bier, Eier
3	Milch, Windeln, Bier, Cola
4	Brot, Milch, Windeln, Bier
5	Brot, Milch, Windeln, Cola

Tabelle 2.3: Warentransaktionen
in Anlehnung an [34]

TID	Brot	Milch	Windeln	Bier	Eier	Cola
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	1	1	0	1	0	1
4	1	1	1	1	0	0
5	1	1	1	1	0	1

Tabelle 2.4: Binäre Darstellung
in Anlehnung an [34]

In Anlehnung an [1] und [34] werden wir jetzt zeigen, wie das Generieren von häufigen Itemset funktioniert. In Tabelle 2.3 werden einige Transaktionen dargestellt. Diese Transaktionen werden uns während dieses Abschnittes begleiten, um so den Algorithmus besser verstehen zu können. Eine andere Möglichkeit, die Transaktionen darzustellen ist die binäre Darstellung. Für unsere Absicht, nämlich das Finden von Assoziationsregeln, ist die binäre Darstellung besser. Tabelle 2.4 veranschaulicht diese Darstellung. Jetzt wollen wir auch das Problem definieren:

> „Gegeben sei eine Transaktionsdatenbank T. Finde alle Regeln, die Support $\geq$ *minsupp* und Confidence $\geq$ *minconf* haben, wobei *minsupp* und *minconf* die entsprechende Support und Confidence Referenzschwellen sind." [34]

Hierfür bietet sich eine Methode, die die Werte des *Support* und der *Confidence* aller möglichen Regeln berechnet. Das ist auf der einen Seite sehr aufwendig, da man exponentiell viele Regeln extrahieren kann. Auf der anderen Seite werden bei der Generierung der Regeln sowieso mehr als die Hälfte abgeworfen. Somit gehen die Berechnungen zum Großteil verloren [34].

Zu diesem Zweck gehen Agrawal et al. [1] wie folgt vor. Sie definieren zuerst das Problem formal (siehe Abschnitt 2.3). Dann spalten sie die Lösung in zwei Schritte:

- Schritt 1: Generiere alle Kombinationen von Items, die eine Referenzschwelle, *minsupport*, übersteigen. Solche Kombinationen von Items werden als *häufig* bezeichnet.

Alle andere Kombinationen, deren Supportwerte dem *minsupport* nicht entsprechen werden *klein* (engl. „small") genannt. Dabei können andere Nebenbedingungen die Menge der zulässigen Regeln beeinflussen. Zum Beispiel möchte man, dass nur bestimmte Items in Regelrumpf kommen. Dann würde es reichen, wenn nur diese Regeln generiert werden, die diese Items auch enthalten.

- Schritt 2: Aus den häufigen Itemsets, die im Schritt 1 generiert wurden, werden dann die Assoziationsregeln gebildet. Dabei werden nur die Regeln erzeugt, die die *minconfidence* überschreiten.

Wenn alle häufige Itemmengen generiert wurden (Schritt 1), dann ist es eine leichte Aufgabe die Assoziationsregeln (Schritt 2) zu bilden.

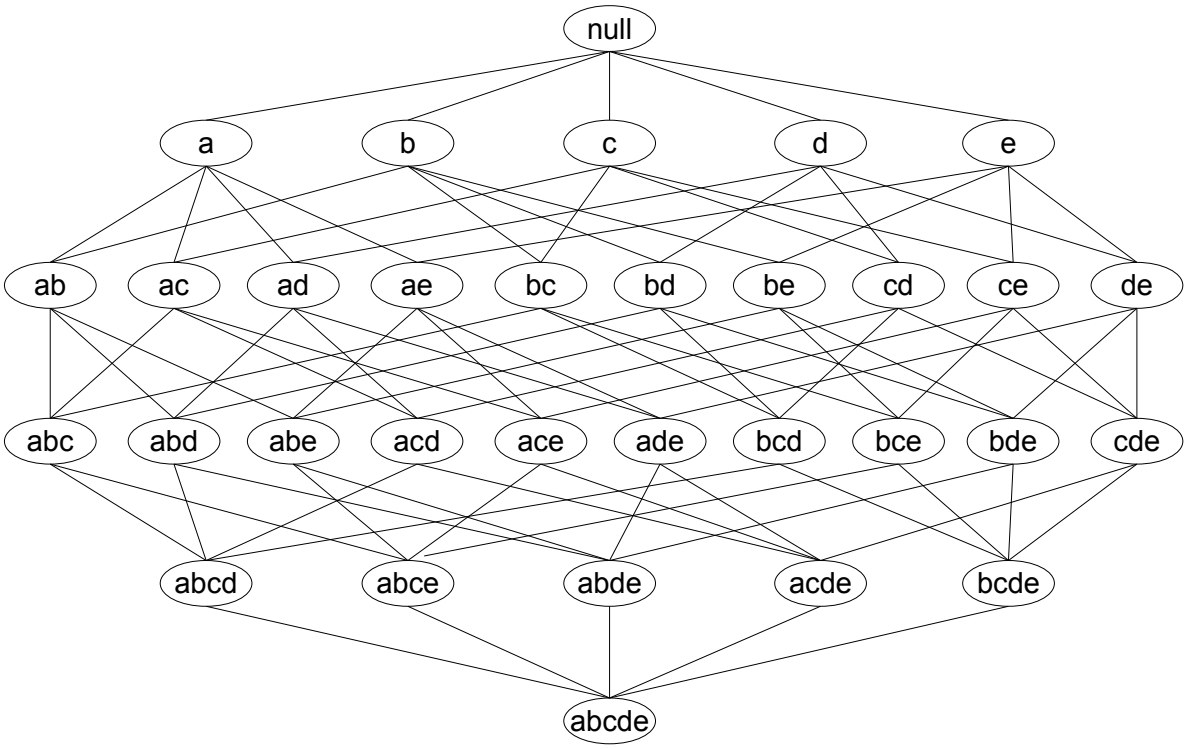


Abbildung 2.5: Itemset Gitter
in Anlehnung an [34]

Jetzt widmen wir uns dem ersten Schritt, Generierung von häufigen Itemsets. Mit Hilfe eines Gitters können alle möglichen Itemsets aufgezeigt werden. Abbildung 2.5 zeigt ein Itemset-Gitter für $I = a,b,c,d,e$. Im Allgemeinen kann eine Datenmenge mit k Elementen möglicherweise 2^k -1 häufige Itemsets generieren. Da aber k sehr groß sein könnte, ist der zu durchsuchende Bereich exponentiell groß [34]. Eine Möglichkeit, dies zu tun, ist die direkte Methode. Das bedeutet, dass für jedes potenzielle Itemset im Gitter (siehe Abbildung

2.5 der Supportwert ermittelt wird. Dies wiederum heißt, dass jedes Itemset mit jeder Transaktion in der Datenbank verglichen werden muss. Sollte das potenzielle Itemset in einer Transaktion vorkommen so wird sein Supportwert um eins erhöht. Zum Beispiel wird der Supportwert für das Itemset *(Brot, Milch)* drei Mal erhöht, da es in drei Transaktionen vorkommt (siehe Tabelle 2.3 [34]). Dieser Ansatz besitzt allerdings eine Komplexität von der Form *O(NMw)*, wo N die Anzahl der Transaktionen ist, M die Anzahl der potenzielle Itemsets darstellt (M = 2^k - 1) und w die maximale Transaktionslänge [34].

Um die Komplexität bei der Generierung von häufigen Itemsets zu reduzieren, kann man sich überlegen, welche Parameter man beeinflussen kann. In Anlehnung an [34] können folgende Möglichkeiten in Betracht gezogen werden:

1. Man kann die Anzahl der potenziellen Itemsets M reduzieren. Hierfür bietet sich die *Apriori* Eigenschaft. Dadurch kann man potenzielle Itemsets ausschließen ohne ihre Support berechnen zu müssen. Dazu kommen wir gleich.

2. Man kann die Anzahl der Vergleich reduzieren. Anstatt jedes potenzielle Itemsets mit jeder Transaktion zu vergleichen kann man die Anzahl der Vergleiche reduzieren, indem man eine andere Datenstruktur verwendet.

Ganz kurz beschrieben besagt die *Apriori* Eigenschaft, dass wenn ein Itemset häufig ist, dann sind auch alle seine Teilmengen häufig. Formal lässt sich das so ausdrücken [4]:

> Falls X häufig ist, dann ist auch $X' \subseteq X$ häufig

Auf der anderen Seite kann man diese Eigenschaft auch anders auslegen. Sollte ein Itemset nicht häufig sein, dann sind auch alle seine Teilmengen nicht häufig. Wir wissen, dass ein Itemset I nicht häufig ist, wenn es den Schwellenwert für *minsupport* nicht übersteigt, also $P(I) < minsupport$. Wird zu diesem Itemset ein neues Item A hinzugefügt, dann kann das so entstandene Itemset $(I \cup A)$ nicht häufiger als I sein. Daher ist $(I \cup A)$ auch nicht häufig. Daraus folgt: $P(I \cup A) < minsupport$ [11].

Diese Eigenschaft wird später dazu benutzt um potenzielle Itemsets auszuschließen, ohne ihren Support berechnen zu müssen. Um dies zu veranschaulichen betrachten wir Abbildung 2.6. Nehmen wir an, dass Itemset {c,d,e} häufig ist. Natürlich müssen alle Transaktionen, die {c,d,e} enthalten, auch seine Teilmengen {c,d}, {d,e}, {c,e}, {c}, {d} und {e} enthalten. Demzufolge, wenn {c,d,e} häufig ist, müssen auch seine Teilmengen häufig sein (in Abbildung 2.6 sind es die grauen Felder) [34].

Jetzt wollen wir zeigen, dass das auch umgekehrt gilt. Dazu betrachten wir Abbildung 2.7. Es wird angenommen, dass das Itemset {a,b,c} nicht häufig ist. Dann sind auch alle seine Obermengen, {a,b,c,d}, {a,b,c,e} und {a,b,c,d,e}, auch nicht häufig. In Abbildung 2.7 sind alle grauen Felder nicht häufig [34].

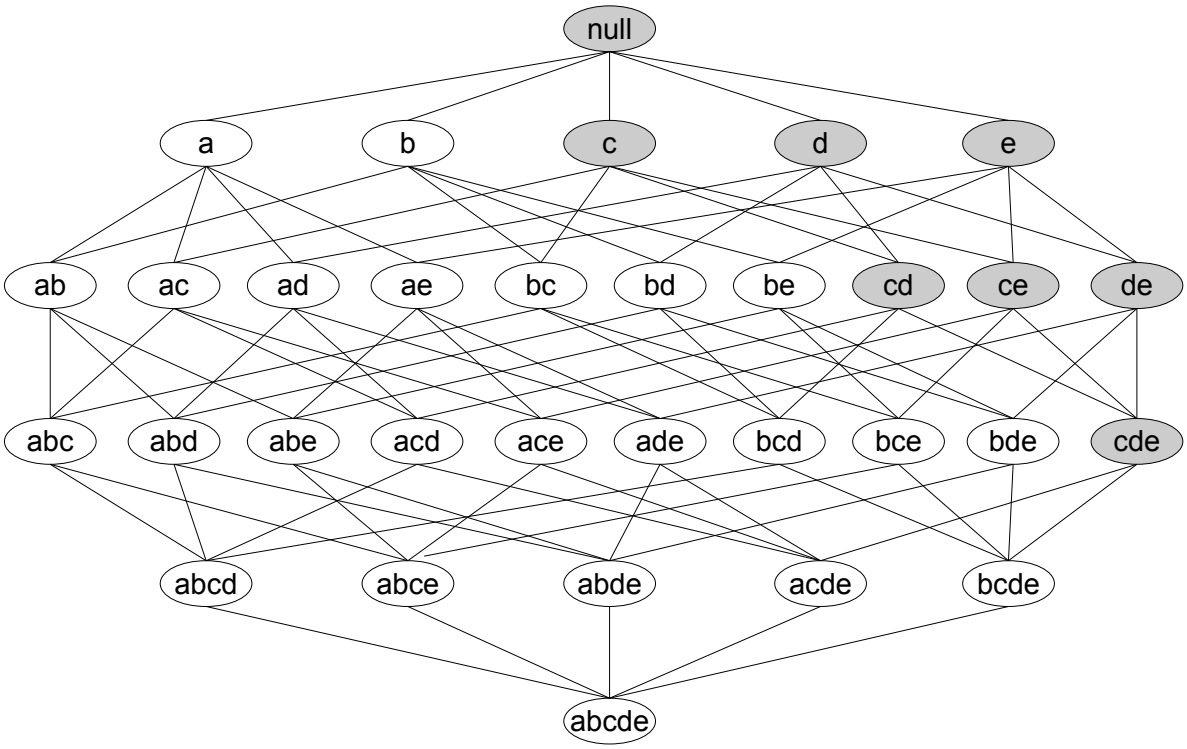


Abbildung 2.6: Apriori Eigenschaft - häufige Itemsets
in Anlehnung an [34]

Abbildung 2.8 zeigt, wie *Apriori* Algorithmus grundsätzlich funktioniert. Diese Überlegungen werden in Anlehnung an [34] durchgeführt. Das Beispiel bezieht sich auf die Transaktionen in Tabelle 2.3. Des Weiteren wird angenommen, dass die Referenzschwelle für *Support* 60% ist. Das entspricht einer Anzahl von 3 Transaktionen. Am Anfang werden alle Items als potenzielles 1-Itemset betrachtet. Nachdem die Supportwerte für alle Items ermittelt wurden, werden die potenziellen Itemsets {Cola} und {Eier} ausgeschlossen, da sie die minimale Anzahl an Transaktionen 3 nicht erreicht haben. In der nächsten Wiederholung werden die potenziellen 2-Itemsets generiert und zwar nur aus der Menge der häufigen 1-Itemsets. Man sieht schon, dass unter der Menge der potenziellen 2-Itemsets in der mittleren Tabelle in Abbildung 2.8 keine Itemsets enthalten sind, in denen *Cola* oder *Eier* vorkommen. Das ist eine Folgerung der *Apriori* Eigenschaft, die besagt: Wenn ein 1-Itemset nicht häufig ist, dann sind folglich auch alle seine Obermengen nicht häufig. Danach werden die Supportwerte für die potenziellen 2-Itemsets berechnet. Die potenziellen 2-Itemsets {Bier, Brot} und {Bier, Milch} sind nicht häufig und daher werden für die Generierung von potenziellen 3-Itemsets nicht verwendet. Aus den übrig gebliebenen 4 häufigen Itemsets werden potenzielle 3-Itemsets gebildet. Das einzige Itemset, das gebildet werden kann ist daher {Brot, Milch, Windeln} [34].

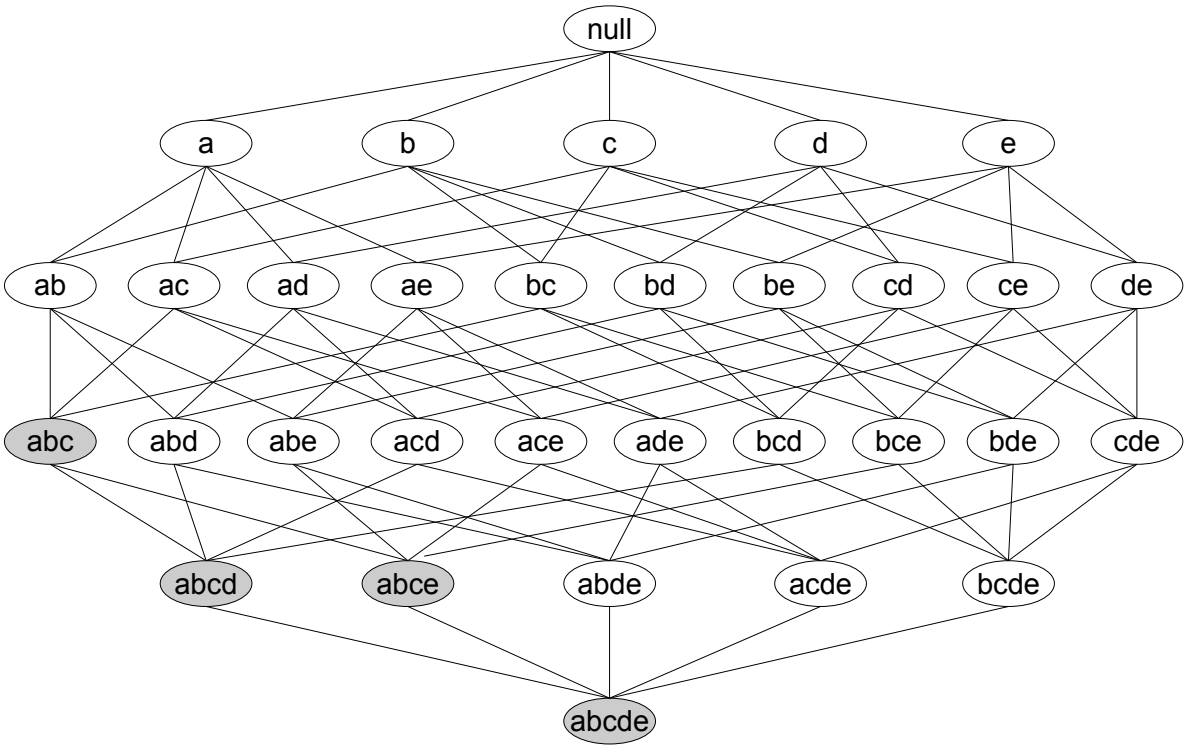


Abbildung 2.7: Apriori Eigenschaft - nicht häufige Itemsets
in Anlehnung an [34]

Jetzt können die Ergebnisse des *Apriori* Algorithmus mit denen verglichen werden, die entstanden wären, wenn man die *Apriori* Eigenschaft nicht genutzt hätte. Solch eine direkte Methode hätte zuerst alle 6 Items als Itemsets eingestuft. Kein Unterschied zur *Apriori*-Methode. In der weiterer Folge hätte die direkte Methode $\binom{6}{2} = 15$ 2-Itemsets gebildet und später $\binom{6}{3} = 20$. Also insgesamt 41(6+ 15 + 20). Macht man sich die *Apriori* Eigenschaft aber zunutze, entstehen dadurch nur 13 potenzielle häufigen Itemsets [34]. Daraus folgt:

$$\binom{6}{1} + \binom{4}{2} + 1 = 6 + 6 + 1 = 13$$

Abbildung 2.9 veranschaulicht den Pseudocode zur Generierung von häufigen Itemsets, was ein Teil vom *Apriori* Algorithmus ist. Sei C_k die Menge der potenziellen k-Itemsets und F_k sei die Menge der häufigen k-Itemsets. Der Algorithmus durchläuft die Transaktionsdatenbank, um den Supportwert für jedes Item zu bestimmen (Schritte 1 und 2). Darauf werden alle häufigen 1-Itemsets, F_1 bekannt sein. Dann werden potenzielle k-Itemsets aus den häufigen *(k-1)*-Itemsets gebildet (Schritt 5). Hierfür wird die Funktion *apriori-gen* verwendet. Diese Funktion wird später näher erläutert. Die Supportwert der potenziellen Itemsets werden in Schritte 6 bis 10 bestimmt. Danach werden alle potenziellen Itemsets

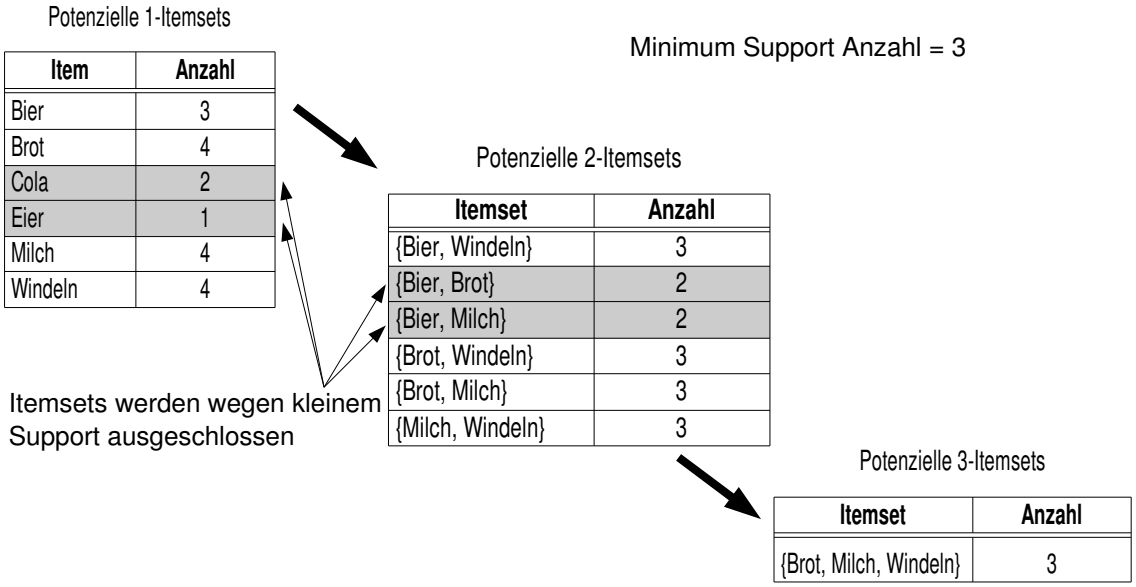


Abbildung 2.8: Generierung von häufigen Itemsets mit Hilfe von *Apriori* Eigenschaft in Anlehnung an [34]

eliminiert, deren Supportwerte kleiner als *minsupport* sind (Schritt 12). Der Algorithmus hört auf, wenn keine neuen häufigen Itemsets generiert werden können (Schritt 13) [34].

Grundsätzlich gibt es verschiedene Möglichkeiten, potenzielle Itemsets zu bilden. In Abbildung 2.9 ist die Funktion *apriori-gen* im Schritt 5. An dieser Stelle sollen drei Möglichkeiten erwähnt werden, wobei eine davon ausführlich diskutiert wird. Allen gemeinsam ist, dass sie zuerst potenzielle k-Itemsets generieren und diese dann mit Hilfe von Referenzschwelle kürzen. In Anlehnung an [34] können potenzielle Itemsets folgendermaßen gebildet werden:

- Direkte Methode: Es werden zuerst alle potenziellen k-Itemsets generiert und erst dann wird die Menge mit Hilfe von *minsupport* reduziert. Diese Methode ist nicht effizient, da sie eine Komplexität von $O(\sum_{k=1}^{d} k \times \binom{d}{k})$ aufweist [34].
- $F_{k-1} \times F_1$ Methode: Diese Methode erweitert jedes (k-1)-Itemset mit anderen häufigen Items (1-Itemset). Ein Nachteil dieses Verfahrens ist, dass es nicht sichergestellt werden kann, dass ein und dasselbe potenzielle Itemset mehrmals generiert wird [34].
- $F_{k-1} \times F_{k-1}$ Methode: Das ist die Methode, die im *Apriori* Algorithmus verwendet wird. Sie wird jetzt näher erläutert.

Diese Methode erzeugt potenzielle k-Itemsets, indem zwei häufige (k-1)-Itemsets verbunden werden. Und das darf nur dann gemacht werden, wenn die ersten (k-2)-Items der

1: $k = 1$
2: $F_k = \{\ i \mid i \in I \wedge \sigma\ (\{i\}) \geq N \times minsup\}$. {Finde alle häufigen 1-Itemsets}
3: **repeat**
4: k=k+1.
5: C_k = apriori-gen(F_{k-1}). {Generiere potenzielle Itemsets}
6: **for** each transaction $t \in T$ **do**
7: C_t = subset(C_k,t) {Identifiziere alle Kandidaten, die zu t gehören}
8: **for** each candidate itemset $c \in C_t$ **do**
9: $\sigma(c)$=$\sigma(c)$ + 1. {Erhöhe Support um 1}
10: **end for**
11: **end for**
12: $F_k = \{\ c | c \in C_k \wedge \sigma\ (c) \geq N \times minsup\}$. {Finde häufigen k-Itemsets}
13: Result = $\bigcup F_k$.

Abbildung 2.9: Pseudocode zur Generierung von häufigen Itemsets
in Anlehnung an [34]

beiden Itemsets identisch sind [34]. „Dabei wird unterstellt, dass die Items in den Itemsets in einer *lexikographischen Reihenfolge* sortiert sind" [13]. Seien A=$\{a_1, a_2, ...a_{k-1}\}$ und B=$\{b_1, b_2, ...b_{k-1}\}$ zwei häufige *(k-1)*-Itemset. Beide werden zusamengeführt, wenn folgende Bedingungen erfüllt sind:

$$a_i = b_i \text{ (wobei } i = 1, 2, \ldots, (k\text{-}2)) \text{ und } a_{k-1} \neq b_{k-1}$$

In Abbildung 2.10 werden die häufigen Itemsets {Brot, Windeln} und {Brot, Milch} zusammengeführt. Somit entsteht ein potenzielles 3-Itemset {Brot, Milch, Windeln}. Natürlich muss geprüft werden, ob dieses 3-Itemsets häufig ist.

Nachdem die häufigen Itemsets generiert wurden, werden daraus Assoziationsregeln gebildet. Hierfür wird die andere wichtige Kennzahl, die zur Bildung der Regeln verwendet wird, die *Confidence* (siehe Abschnitt 2.4.2). Ein häufiges k-Itemset Y kann bis zu 2^k-2 Assoziationsregeln erzeugen, wenn man von den Regeln absieht, die eine leere Menge im Regelrumpf beziehungsweise im Regelkopf haben ($Y \rightarrow \emptyset$ und $\emptyset \rightarrow Y$). Eine Assoziationsregel wird gebildet, indem das Itemset Y in zwei Untermengen X und Y–X aufgeteilt wird und zwar so, dass $X \rightarrow Y$–X die Referenzschwelle für *Confidence* erfüllt. Es wird darauf hingewiesen, dass alle so generierten Regeln bereits die Referenzschwelle für *Support* erfüllen [34].

Die Berechnung der *Confidence* einer Assoziationsregel benötigt keine neue Durchsuchung der Transaktionsdatenbank. Es sei die Regel {1,2} $\rightarrow$ {3} gegeben, die von dem häufigen Itemset X = {1,2,3} gebildet wurde. Die *Confidence* dieser Regel ist $\sigma(\{1,2,3\})/\sigma(\{1,2\})$. Da {1,2,3} häufig ist muss {1,2} auch häufig sein (gemäß *Apriori* Eigenschaft). Weil die Supportwerte der beiden Itemsets während der Generierung der häufigen Itemsets ermittelt wurden, besteht keinen Bedarf die Transaktionsdatenbank wieder zu lesen [34].

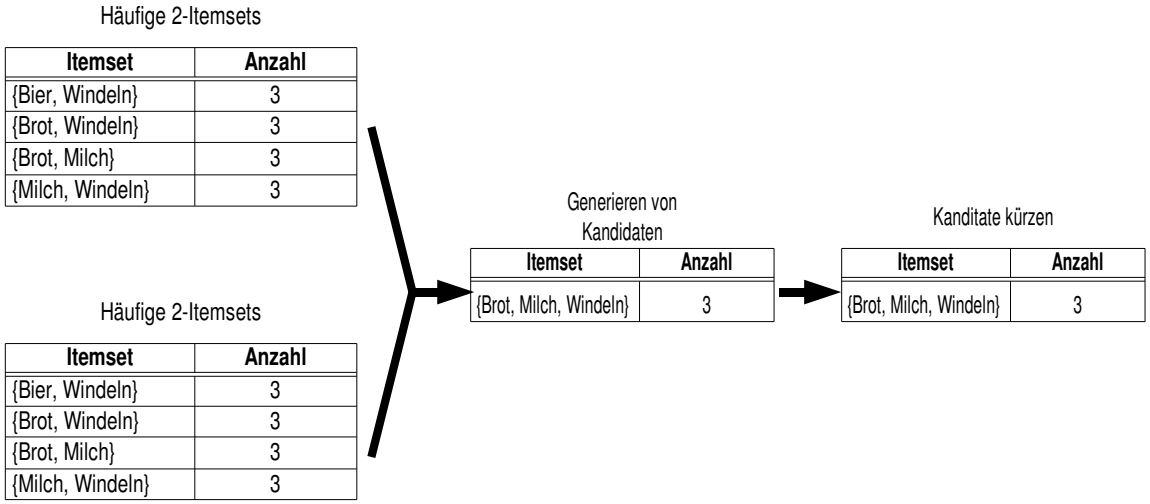


Abbildung 2.10: Beispiel zur Generierung von häufigen Itemsets
in Anlehnung an [34]

Im Unterschied zum *Support* besitzt die *Confidence* diese Eigenschaft nicht. Zum Beispiel kann die *Confidence* der Regel $X \rightarrow Y$ größer, kleiner oder gleich als die *Confidence* von $\hat{X} \rightarrow \hat{Y}$ sein, wobei $\hat{X} \subseteq X$ und $\hat{Y} \subseteq Y$. Dennoch kann beim Vergleich der Regeln, die aus einem häufigen Itemset Y gebildet wurden, folgender Satz hergeleitet werden.

> Wenn eine Regel $X \rightarrow Y - X$ die *Confidence* Referenzschwelle nicht erfüllt, dann muss jede Regel $X' \rightarrow Y - X'$, wo X' eine Teilmenge von X ist, die *Confidence* Referenzschwelle auch nicht erfüllen [34].

Es werden folgende Regeln beobachtet: $X' \rightarrow Y - X'$ und $X \rightarrow Y - X$, wo $X' \subset X$. Die *Confidence* der Regel sind $\sigma(Y)/\sigma(X')$ beziehungsweise $\sigma(Y)/\sigma(X)$. Da aber X' eine Untermenge von X ist, gilt $\sigma(X') \geq \sigma(X)$. Daher kann die erste Regel nicht größere *Confidence* als die zweite haben [34].

Abbildung 2.11 zeigt ein Gitter mit allen möglichen Assoziationsregeln, die aus dem häufigen Itemset {a,b,c,d} generiert werden können. Sollte eine Regel die *Confindence* Referenzschwelle nicht erreichen, so können alle darunter liegenden Regeln sofort gekürzt werden,

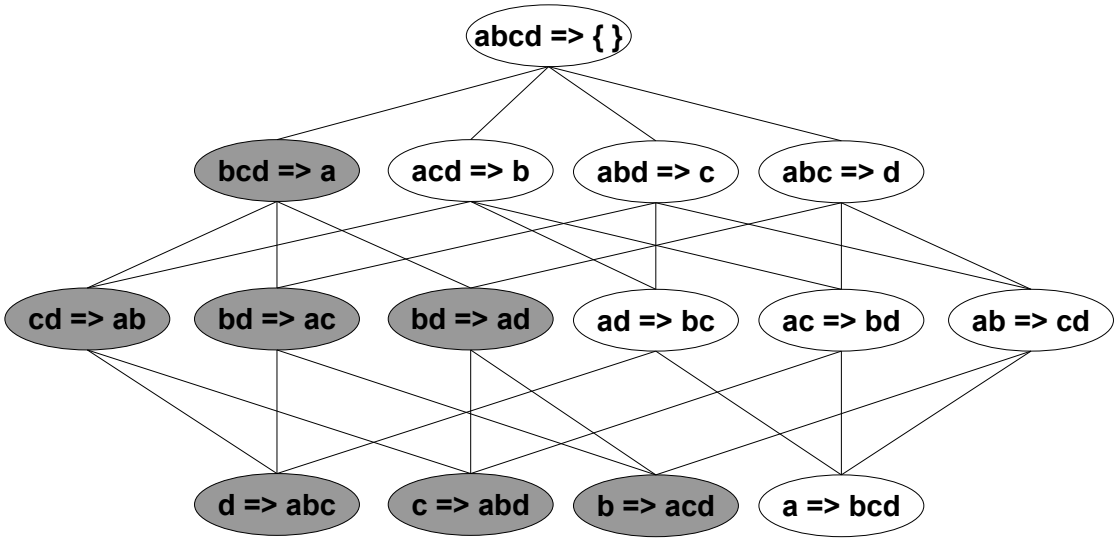


Abbildung 2.11: Beispiel zur Generierung von Assoziationsregeln
in Anlehnung an [34]

da sie ebenfalls die *Confidence* Referenzschwelle nicht erreichen. Es wird angenommen, dass die Regel $\{b,c,d\} \rightarrow a$ die *Confidence* Referenzschwelle nicht erreicht. Daher können alle Regeln, die a im Regelkopf enthalten, verworfen werden (in Abbildung 2.11 sind es die grauen Felder) [34].

Kapitel 3

Visualisierungstechniken und Assoziationsregeln

In diesem Kapitel werden verschiedene Methoden vorgestellt, die die Visualisierung von Assoziationsregeln ermöglichen. Am Anfang wird es einen kurzen Überblick über die diversen Techniken geben. Diese können in folgende Bereiche eingeteilt werden:

- Tabellen
- Graphen
- Matrizen

Alle diese Techniken werden im Zusammenhang mit Assoziationsregeln dargestellt. Dabei gilt es zu erforschen, welche Visualisierungstechniken geeignet sind, um Assoziationsregeln wirkungsvoll darstellen zu können. Zu diesem Zweck werden an dieser Stelle einige Fragen definiert. Diese sollen dazu dienen, um die Problematik bei der Visualisierung von Regeln aufzuzeigen.

1. Was sind „gute" Regeln?
2. Wie kann man mittels Visualisierungstechniken auf solche Regeln kommen?
3. Wie viele Assoziationsregeln können gemeinsam dargestellt werden?
4. Aus wie vielen Items bestehen die Regeln?

Dieses Kapitel soll die Antworten auf diese Fragen geben. Im Abschnitt 2.4 wurden einige Interessantheitsmaße vorgestellt. Diese werden jetzt genutzt, um *gute* von *schlechten* Assoziationsregeln zu unterscheiden. Es soll auch gezeigt werden, wie viele Regeln gemeinsam dargestellt werden können.

Außerdem soll eine allgemeine Definition von Visualisierung gegeben werden. Diese Definition sollte die Wichtigkeit der Visualisierung hervorheben.

3.1 Einführung

Visualisierung ist das Anzeigen von Information in graphischer oder tabellarischer Form. Erfolgreiche Visualisierung verlangt, dass die zugrunde liegenden Daten oder Informationen in sichtbare Form umgewandelt werden, so dass die Eigenschaften dieser Daten und die Beziehungen zwischen den Attributen analysiert werden können [34].

Eine andere Definition für Visualisierung geben Card et al. [5]:

> „The use of computer-supported, interactive, visual representations of data to amplify cognition."

Diese Definition beinhaltet einige essentiellen Begriffe. *Erkennen* bedeutet hier der Erwerb oder die Anwendung vom Wissen. Auf der anderen Seite steht *interaktiv* für eine wechselseitig beeinflussende Beziehung. Die *Darstellung* ist das Abbilden der Daten in visuelle Objekten, Attributen und Beziehungen.

Die Motivation für Anwendung von Visualisierung ist, dass Menschen große Mengen an visueller Information schnell wahrnehmen und dabei auch Muster finden können [34].

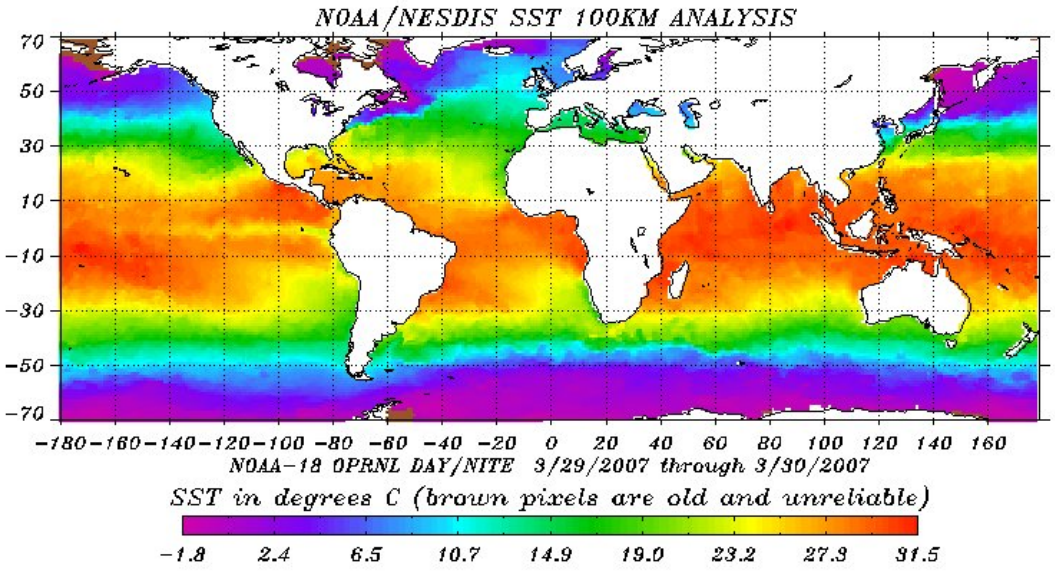


Abbildung 3.1: Temperatur der Meeresoberfläche

Abbildung 3.1[1] zeigt die Temperatur der Meeresoberfläche zwischen 29 und 30 März 2007. Auf dieser Weise kann man in wenigen Sekunden erkennen, dass die Temperatur des Wassers in der Nähe vom Äquator am höchsten ist [34].

[1] Quelle:http://www.osdpd.noaa.gov/PSB/EPS/SST/data/global100.cf.gif Zuletzt aufgerufen: 31.03.2007

Es gibt einen anderen Aspekt, der an dieser Stelle erwähnt werden muss. Es handelt sich dabei um das Wissen, „das in den Köpfen der Leute verschlossen ist" [34]. Es ist oft schwer oder unmöglich, solch ein Wissen in einem Algorithmus zu implementieren und voll auszunutzen. Hierfür können Experten die visuellen Darstellungen bestimmter Daten untersuchen. Dadurch lassen sich uninteressante Muster schneller entdecken und man kann sich auf das Wesentliche konzentrieren.

Schumann et al. [33] definieren das Ziel der Visualisierung folgendermaßen:

> „Die wissenschaftlich-technische Visualisierung, auch als *Scientific Visualization* bezeichnet, hat die Aufgabe, geeignete visuelle Repräsentationen einer gegebenen Datenmenge zu erzeugen, um damit eine effektive Auswertung zu ermöglichen".

Sie definieren auch allgemeine Anforderungen an Visualisierung [33]. Sie muss:

- expressiv,
- effektiv und
- angemessen

sein.

Bevor diese Begriffe näher erläutert werden, soll noch gesagt werden, was für Variablen es gibt. Im Allgemeinen können drei Typen von Variablen unterschieden werden:

1. Nominale Variablen stellen eine Menge von Variablen dar, deren Ausprägungen unterschiedlich sind, aber nicht in eine Rangfolge gebracht werden können. Eine solche Variable ist zum Beispiel die Variable *language in home* aus dem *AdultUCI* Datensatz vom Paket *arules* der Programmiersprache *R* [30]. Sie kann drei Ausprägungen annehmen, *english*, *spanish* oder *other language*.
2. Ordinalzahlen können eingeordnet werden. Zum Beispiel kann die Variable *education* aus demselben Datensatz die Ausprägungen *no college graduate* oder *college graduate* annehmen.
3. Quantitative Variablen haben einen numerischen Bereich. Die Variable *age* stellt ein Beispiel für eine quantitative Variable.

Die Anforderungen an eine Visualisierung werden jetzt näher erläutert. *Expressivität* bedeutet, dass die Daten unverfälscht dargestellt werden müssen. Zum Beispiel können Daten verfälscht dargestellt werden, wenn ein und derselbe Datensatz mittels zwei verschiedenen Visualisierungstechniken präsentiert wird. Abbildungen 3.2 und 3.3 sollen dies veranschaulichen. Auf der linken Abbildung wird mittels eines Balkendiagramms der Zusammenhang zwischen Automarken und deren Herkunftsland präsentiert. Auf der rechten Abbildung

wird dieselbe Information mittels Punkten vermittelt. Da in diesem Beispiel einen nominalen Zusammenhang abgebildet werden soll, eignet sich der Balkendiagramm zu diesem Zweck nicht. Die meisten Menschen würden die Länge der Balken als eine Verschlüsselung für Ordinalzahlen oder quantitative Variablen wahrnehmen. Dadurch kann der Eindruck entstehen, dass zum Beispiel Schweden besser als USA ist [5]. Diese Eigenschaft wird auch *Ausdrucksfähigkeit* genannt.

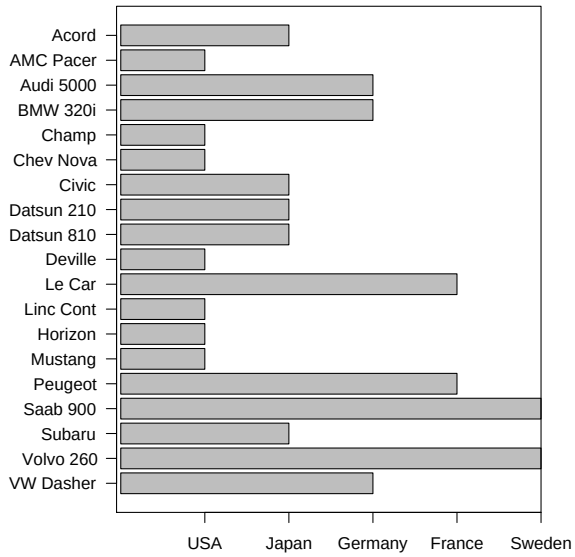


Abbildung 3.2: Falsche Darstellung

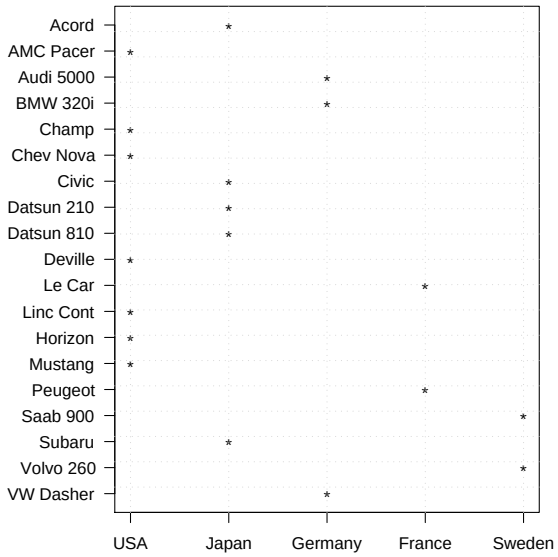


Abbildung 3.3: Richtige Darstellung

Auf der anderen Seite können für eine Datenmenge mehrere Visualisierungstechniken existieren, die diese Datenmenge präsentieren können. Es sollte daher diese Technik gewählt werden, die die Datenmenge unter Berücksichtigung der Zielsetzung am besten abbildet. Diese Eigenschaft wird *Effektivität* genannt.

Angemessenheit soll nicht die Qualität einer Visualisierung beschreiben. Es geht dabei viel mehr um den Nutzen, der durch eine bestimmte Visualisierung erzielt wird. Es werden Kosten und Nutzen einer Visualisierung gegenüber gestellt. Als Kosten kann zum Beispiel der Aufwand zur Interpretation einer Visualisierung monetär ausgedrückt werden. Eine Visualisierung, die nicht angemessen ist, kann daher nicht effektiv sein.

Je nachdem, was man präsentieren möchte, gibt es verschiedene Möglichkeiten Daten darzustellen. Wenn es nur ein Attribut gibt, dann werden die Objekte in Kategorien zusammengefasst. Sollte ein Objekt mehrere Attribute besitzen, dann kann dieses Objekt als Zeile (Spalte) einer Tabelle oder als eine Linie eines Graphs dargestellt werden. Eine andere Möglichkeit besteht darin, dass Objekte oft als Punkte in zwei- oder dreidimensionalem Raum präsentiert werden. Dabei kann der Punkt eine Figur (zum Beispiel Kreis, Kreuz oder Kästchen) sein [34].

Um die Analyse und Interpretation von Assoziationsregeln zu unterstützen, können entweder graphische oder wörtliche Visualisierungstechniken angewendet werden [6].

- Wörtliche Darstellung. Bei der wörtlichen Darstellung werden unter anderem verschiedene Zeichen und mathematischen Operatoren verwendet, um die Regeln zu präsentieren. Diese werden vom Anwender der Reihe nach verarbeitet. Der Vorteil dabei ist, dass sie von verschiedenen Anwender auf gleicher Weise wahrgenommen werden. Der Nachteil liegt darin, dass die wörtliche Darstellung nicht geeignet ist, die komplexen Ergebnisse eines Data-Miningprozesses darzustellen und zu analysieren. Beispiel:

Regel	*conf.*	*supp.*
$\{Edct = HSGr,\ Incm = Smll\} \rightarrow \{HrPW = FllT\}$	12.04	66.62

- Auf der anderen Seiten bieten die graphischen Methoden zur Visualisierung von Assoziationsregeln mächtigere Formen von Darstellung. Eine graphische Darstellung kann mehrere Formen annehmen, da die zugrunde liegenden Daten auf verschiedenen Arten (Position, Gestalt, Farbe und Größe) abgebildet werden können.

Um die Analyse und Interpretation von Assoziationsregeln zu ermöglichen, wurden verschiedene Visualisierungstechniken entwickelt. Jede dieser Methoden erfordert zwei wesentliche Komponenten: graphische Darstellung der Attributen und eine Technik zur Veranschaulichung der Beziehungen zwischen diesen Attributen. Diese Techniken können in zwei Klassen eingeordnet werden [6]:

- **Matrizen**
- **Graphen**

Diese werden an dieser Stelle näher erläutert.

3.2 Matrizen

Die Grundidee bei dieser Visualisierungstechnik ist, dass Regelrumpf und Regelkopf auf jeweils X und Y Achse abgebildet werden. Dabei wird eine mögliche Assoziation zwischen den Regelrumpf und Regelkopf durch die Anwesenheit eines Zeichens in der Schnittzelle gezeigt [6]. Dieses Zeichen kann entweder den Wert des Supports oder der Confidence darstellen. Abbildung 3.4 soll dies veranschaulichen. In diesem Beispiel stellt die Höhe der Linie im Schnittpunkt die Confidence der Regeln $\{Capital\ Gain = none\} \rightarrow \{Sex = Male\}$ und $\{Income = small\} \rightarrow \{Sex = Male\}$.

Möchte man die Assoziationsregeln $\{Capital\ Gain = none\ \&\ Income = small\} \rightarrow \{Sex = Male\}$, $\{Capital\ Gain = none\} \rightarrow \{Sex = Male\}$ und $\{Income = small\} \rightarrow \{Sex =$

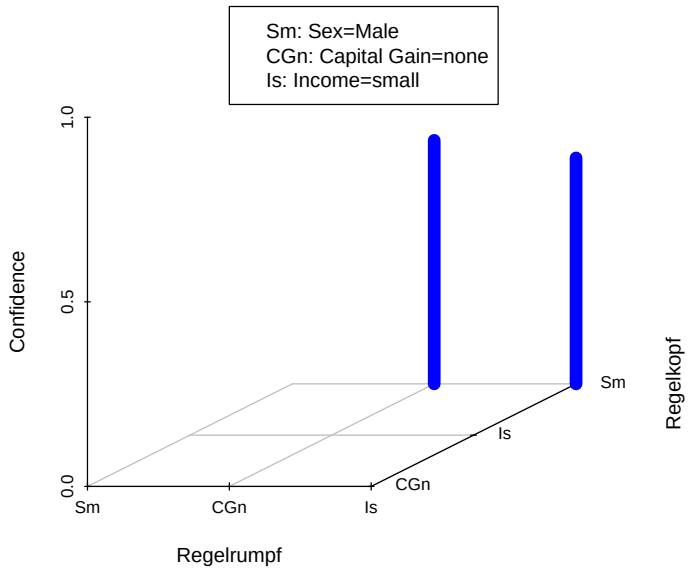


Abbildung 3.4: 3D Matrix

$Male\}$ gleichzeitig betrachten, bietet sich die Möglichkeit, ein weiteres Element auf der X-Achse hinzuzufügen. Dieses Element soll den Regelrumpf $\{Capital\ Gain = none\ \&\ Income = small\}$ (CGn-is) darstellen. Abbildung 3.5 zeigt alle drei Regeln.

Die Stärke dieser Visualisierungstechnik liegt darin begründet, dass Regeln, wo der Regelrumpf und Regelkopf jeweils aus einem *binären* Attribut bestehen, sehr gut dargestellt werden können. Sollten aber Regeln untersucht werden, die mehrere Attribute im Regelrumpf und/oder Regelkopf haben, dann ist dieser Ansatz nicht geeignet, da man durch die mehreren Balken nicht erkennen kann, was dahinter steht [26].

Eine erweiterte Form der zweidimensionalen Matrizen stellt das Gitter dar [26]. Abbildung 3.6 zeigt ein solches Gitter. Obwohl dabei wieder eine *ein-zu-ein* Beziehung abgebildet wird, werden Farben dazu benutzt, um die Ergebnisse effektiv darzustellen. Die Achsen bilden den Regelrumpf und Regelkopf ab. In jeder Zelle des Gitters wird durch Farbe den Support einer Regel angegeben. Zum Beispiel deuten die Rosafarben auf einen Supportwert von maximal 0.5 hin. Die hellblauen Zellen hingegen bedeuten einen hohen Supportwert. Dadurch kann eine schnelle Zusammenfassung der Ergebnisse bekommen werden. Diese Technik ist gut geeignet für Itemsets mit mehr Items, weil die Größe einer Zelle keine Eigenschaften abbildet und dadurch mehr Regeln visualisiert werden können.

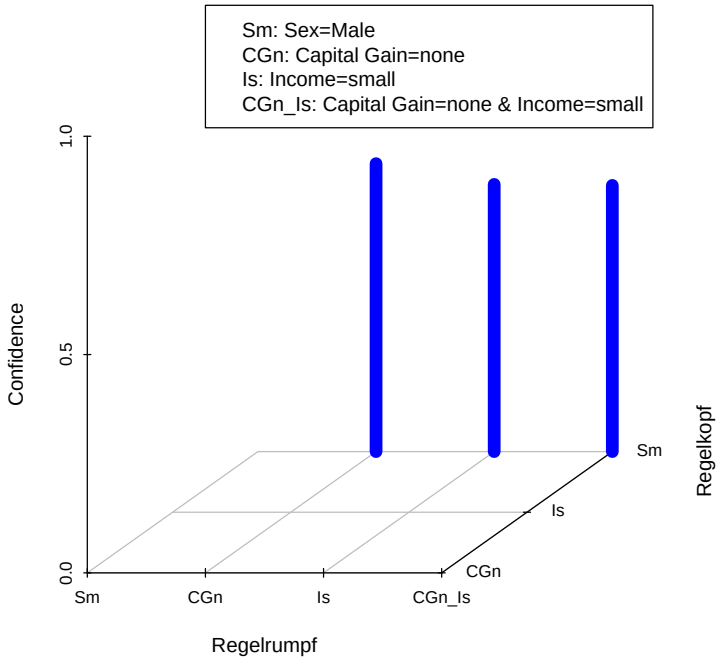


Abbildung 3.5: 3D Matrix

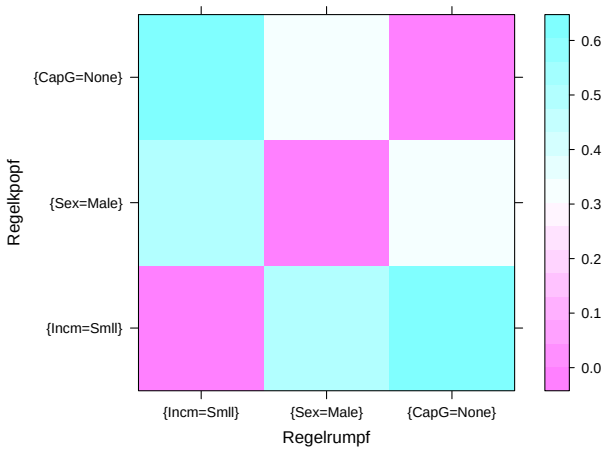


Abbildung 3.6: Gitter

		Income = Small	Income = Large
Sex = Male	Capital Gain = None	14441	5260
	Capital Gain = Low	675	480
	Capital Gain = High	12	922
Sex = Female	Capital Gain = None	9244	904
	Capital Gain = Low	304	64
	Capital Gain = High	8	211

Tabelle 3.1: Kontingenztabelle für AdultUCI Datensatz mit drei Attributen

3.3 Kontingenztabellen und Mosaik Plots

An dieser Stelle wird eine alternative Form der Matrixvisualisierung vorgestellt. Es handelt sich dabei um die so genannten *mosaic plots* [15]. Eine Zelle in einer (mehrdimensionalen) Kontingenztabelle beschreibt nur eine einzige Assoziationsregel. Dabei geben aber die anderen Zellen Informationen an, die für eine gründliche Analyse dieser Regel von Bedeutung sein können. Der Zusammenhang zwischen den Daten kann in einem Mosaik Plot leicht beobachtet werden.

Tabelle 3.1 zeigt Daten von dem Datensatz *AdultUCI*. Diese Daten sind nur auf drei Elementen dieses Datensatzes beschränkt worden, da man die Grundidee des Mosaikplots so besser verstehen kann:

1. Sex - mit Ausprägungen *Male* und *Female*
2. Capital Gain - mit Ausprägungen *None*, *Low* und *High*
3. Income - mit Ausprägungen *Small* und *Large*

Mit einem Mindest-Supportwert von 25% scheint die Regel

$$\{Sex = Male, Income = small\} \rightarrow \{CapitalGain = None\}$$

auf, während die Regel

$$\{Sex = Female, Income = small\} \rightarrow \{CapitalGain = None\}$$

nicht aufscheint. Die beide Regeln würden eine Korrelation zwischen *Income* und *Capital Gain* andeuten, die mittels eines χ^2 Tests der Unabhängigkeit ermittelt werden kann. Ein χ^2 Test ermittelt unter anderem, ob zwischen zwei Merkmale, hier *Sex* und *Capital Gain*, eine Korrelation vorhanden ist. Dieses Beispiel soll nur zeigen, dass durch Festlegen von Mindest-Support (hier 25%) die Menge der entdeckten Assoziationsregeln zwar reduziert wird, aber Regeln verloren gehen könnten, die für das Verständnis und die Analyse von Bedeutung sein könnten [15].

Ein ähnliches Problem kommt dann vor, wenn zwei Assoziationsregeln einen und derselben Regelkopf haben [15]. Das sind Regeln der Form:

$$X_1 \rightarrow Y$$
$$X_2 \rightarrow Y$$

An dieser Stelle kann nicht gesagt werden, ob hier zwei verschiedenen Gruppen entdeckt wurden, die auf Y schließen lassen, oder es mehr oder weniger dieselbe Gruppe sind. Dieses Problem kann nicht gelöst werden, indem weitere Assoziationsregeln beobachtet werden, da es möglich ist, dass beide Regeln einen großen Teil ein und derselben Population beschreiben, während dies wegen zu niedrigem Supportwert nicht ersichtlich wird [15]. Ein kleines Beispiel soll die Aussage veranschaulichen. Betrachtet werden folgende Regeln:

$$\{Sex = Female\} \rightarrow \{CapitalGain = None\} \text{ und}$$
$$\{Income = small\} \rightarrow \{CapitalGain = None\}$$

Diese scheinen bei einem Supportwert von 25% auf. Auf der anderen Seite scheint die „erklärende" Assoziationsregel

$$\{Sex = Female, Income = small\} \rightarrow \{CapitalGain = None\}$$

nicht auf.

Solche Probleme werden schlimmer, wenn eine große Untermenge der entdeckten Regeln gefiltert wurde, um so einen ersten Einblick in die Daten zu gewähren. Das bedeutet natürlich nicht, dass das Herausfiltern von interessanten Regeln eine schlechte Idee ist. Oder, dass Data Mining Systeme alle entdeckten Assoziationsregeln präsentieren müssen, einschließlich Regeln, die die Schwellenwerte für Support und Confidence nicht überschreiten aber vielleicht hilfreich sein könnten, um so andere Regeln zu verstehen [15]. Welche die interessanten Assoziationsregeln sind, ist zusätzlich noch subjektiv.

Man muss unterscheiden zwischen Werkzeuge, die *gute* und *interessante* Assoziationsregeln präsentieren und dem Anwender erlauben, sie zu verstehen und auf der anderen Seite solchen, mit deren Hilfe man *gute* und *interessante* Assoziationsregeln findet. In diesem Sinne stellen Kontingenztabellen und Mosaik Plots einen Teil der ersten Gruppe vor. Sie sollen helfen, die Regeln zu verstehen.

Es sei angenommen, dass eine Datenbank aus einer Tabelle r besteht, mit dem Schema $Z=\{A_1, \ldots, A_p\}$, wo A_j, $j = 1, \ldots, p$ ein kategorisches Attribut ist. Das bedeutet, dass jedes Attribut A_j eine Domain D_j mit verschiedenen Ausprägungen hat. Der verwendeter Ausschnitt des AdultUCI Datensatzes besteht momentan aus nur drei Attributen, *Sex*, *Income* und *Capital Gain*. Diese entsprechen A_1, A_2 und A_3. Jedes dieser drei Attributen hat eine Domain D. Diese sind:

	Income = Small	Income = Large
Sex = Male	15128	6662
Sex = Female	9592	1179

Tabelle 3.2: Kontingenztabelle für AdultUCI Datensatz mit zwei Attributen

1. D_{sex}: *Female* und *Male*
2. D_{income}: *Large* und *Small*
3. $D_{capitalGain}$: *None*, *Low* und *High*

Eine Kontingenztabelle von r ist grundsätzlich eine Tabelle, wo in jeder Zelle eine Anzahl vorhanden ist, die angibt, wie oft eine bestimmte Kombination von Attributen in der Datenbank r vorkommt. Der einfachste Fall ist eine zweidimensionale Kontingenztabelle so wie in Tabelle 3.2, wo alle 4 mögliche Kombinationen von den zwei Attributen jeweils eine Zelle annehmen [15].

Mosaik Plots wurden als graphisches Analogon der mehrdimensionalen Kontingenztabellen von J. Hartigan und B. Kleiner vorgestellt. Jede Zelle einer Kontingenztabelle wird durch ein kleines Kästchen in der Abbildung dargestellt. Die Größe eines Kästchens ist proportional zu der entsprechenden Anzahl in der Kontingenztabelle. Das bedeutet, dass größere Kästchen Kombinationen von Attributen abbilden, die öfter vorkommen [15].

Mit anderen Wörter vermitteln die Kästchen eines Mosaik Plots dieselbe Information wie die Zellen in einer Kontingenztabelle. Der Unterschied dabei ist, dass Mosaik Plots die (relative) Anzahl durch die Größe des Kästchens repräsentieren und nicht durch eine Nummer [15].

Tabelle 3.2 zeigt eine Zusammenfassung der Tabelle 3.1. Hier wurde der Datensatz mittels Aggregation auf zwei Attribute reduziert, *Sex* und *Income*.

Abbildung 3.7 zeigt zweidimensionales Mosaik Plots für die Daten aus Tabelle 3.2. Dieses Mosaik Plot stellt die graphische Darstellung von Kontingenztabellen dar [15]. Diese Abbildung gibt die graphische Ansicht von der gemeinsamen Verteilung ihrer Attributen an. Zum Beispiel gibt die Größe der Fläche aller vier Kästchen in der Abbildung die Anzahl aller Beobachtungen an. Ein einzelnes Kästchen gibt die relative Anzahl der Beobachtungen durch seine Größe an. Die Abbildung ist durch einen kleinen Abstand horizontal geteilt worden. Oberhalb dieses Abstandes befinden sich alle Beobachtungen $\{Income = small\}$ und unterhalb aller Beobachtungen $\{Income = large\}$. Dann wurden diese zwei Teile vertikal geteilt. Jeweils links von diesem vertikalen Abstand befinden sich alle Beobachtungen $\{Sex = Female\}$ und rechts von ihm alle Beobachtungen $\{Sex = Male\}$. Wird ein weiteres Element hinzugefügt (hier *Income*), dann werden die vier Kästchen wieder horizontal geteilt. Man sieht, dass es wenig Frauen mit $Income = large$ gibt (die Größe des Kästchens links unten).

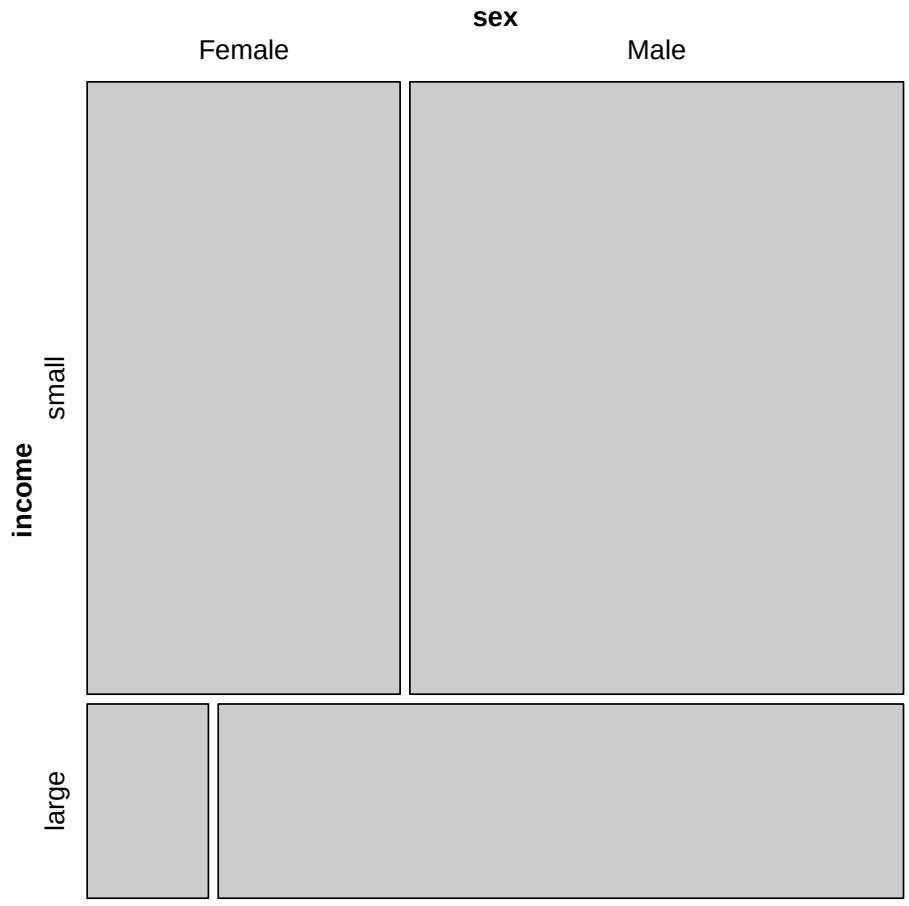


Abbildung 3.7: Zweidimensionales Mosaik Plot

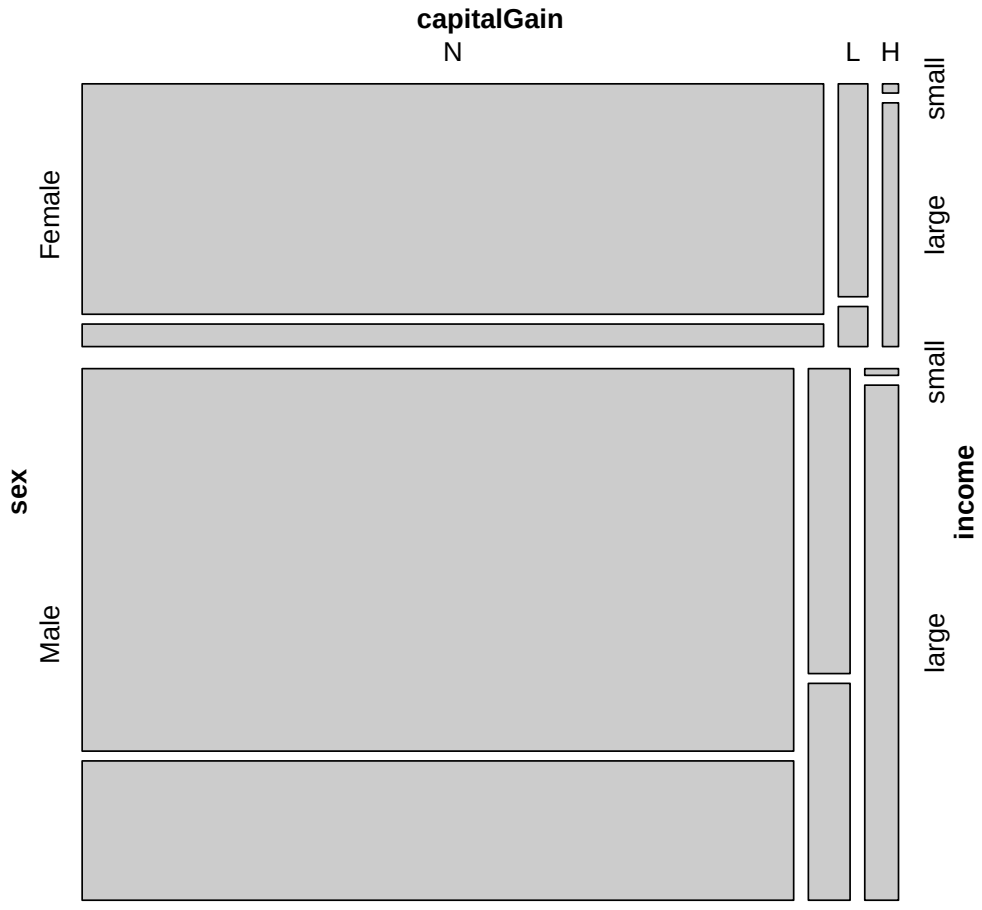


Abbildung 3.8: Dreidimensionales Mosaik Plot

		Income = Small	Income = Large
Sex = Female	Age = Middle-aged	4380	738
	Age = Old	357	37
	Age = Senior	2066	382
	Age = Young	2789	32
Sex = Male	Age = Middle-aged	7778	3627
	Age = Old	568	206
	Age = Senior	3274	2747
	Age = Young	3508	211

Tabelle 3.3: Kontingenztabelle für AdultUCI Datensatz mit drei Attributen

Abbildung 3.8 zeigt ein Mosaik Plot mit drei Elementen, wo allerdings zuerst nach dem Attribut *Sex* horizontal, dann nach *Capital Gain* vertikal und zum Schluss nach *Income* wieder horizontal geteilt wurde. Dadurch kann man in Abbildung 3.8 sehen, dass sich durch die Überschneidung von $\{Sex = Male\}$, $\{CapitalGain = High\}$ und $\{Income = small\}$ ein ganz kleines Kästchen ergibt, das einen ganz kleinen Anteil von der Gesamtfläche der Abbildung annimmt.

An dieser Stelle wird ein weiteres Element zu dem modifizierten *AdultUCI* Datensatz hinzugefügt. Das ist *Age* mit vier Ausprägungen, *Young*, *Middle-aged*, *Senior* und *Old*. Eine Möglichkeit alle Assoziationsregeln $\{Income\ \&\ Age\} \rightarrow \{Sex\}$ mittels Mosaik Plots darzustellen ist, indem alle Attribute im Regelrumpf einer Regel als erklärende Variablen zusammengefasst und in ein zweidimensionales Mosaik Plot abgebildet werden. Der Regelkopf (die abhängige Variable) wird in dem Mosaik Plot mittels Farbe hervorgehoben [15].

Tabelle 3.3 zeigt die Kontingenztabelle für den *AdultUCI* Datensatz mit den Attributen *Sex*, *Age* und *Income*. Das entsprechende Mosaik Plot wird durch Abbildung 3.9 dargestellt.

Abbildung 3.9 zeigt einen Überblick von möglichen Assoziationsregeln, die drei Attribute, *Income*, *Age* und *Sex* umfassen. Das zweite Kästchen von links in der unteren Reihe gibt ein Beispiel für eine Regel mit einer sehr hohen Confidence (die hervorgehobene dunkelgraue Fläche füllt fast das ganze Kästchen aus) an. Diese Regel ist:

$$\{Income = large, Age = Old\} \rightarrow \{Sex = Male\}$$

Auf der anderen Seite ist der Support dieser Regel relativ gering (das ist die Größe der Fläche dieses Kästchens gemessen an der Gesamtfläche). Das erste Kästchen in der oberen Reihe, das die Regel

$$\{Income = small, Age = Middle - aged\} \rightarrow \{Sex = Male\}$$

präsentiert, bildet die Regel mit dem größten Support ab, wobei allerdings jetzt die Confidence gering ist.

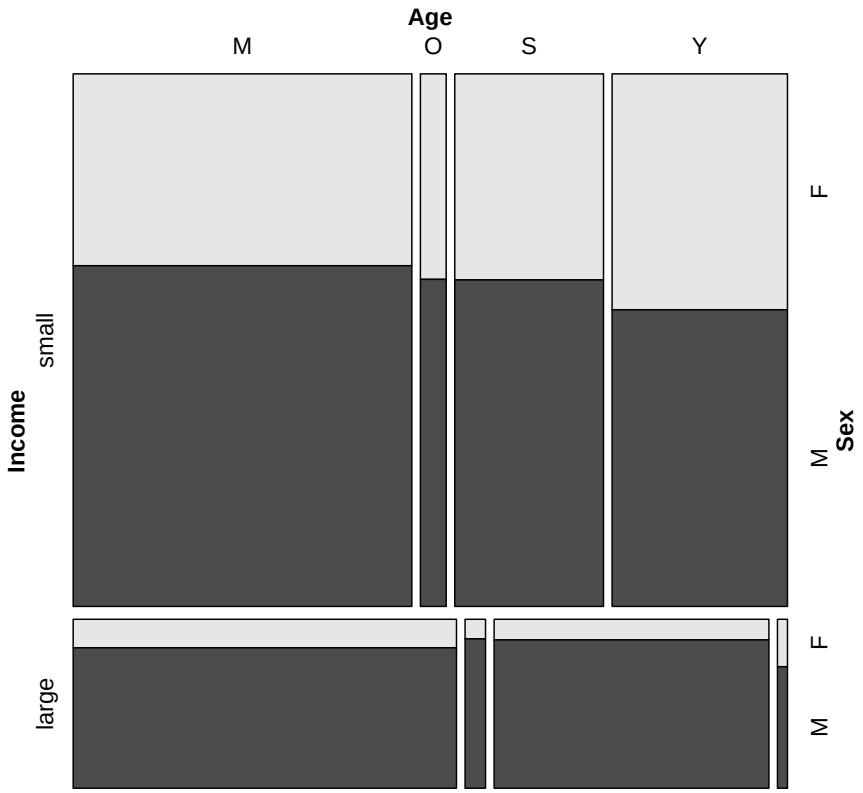


Abbildung 3.9: Mosaik Plot für die Regeln mit Regelkopf $Sex = Female$ oder $Sex = Male$

3.4 Double Decker Plots

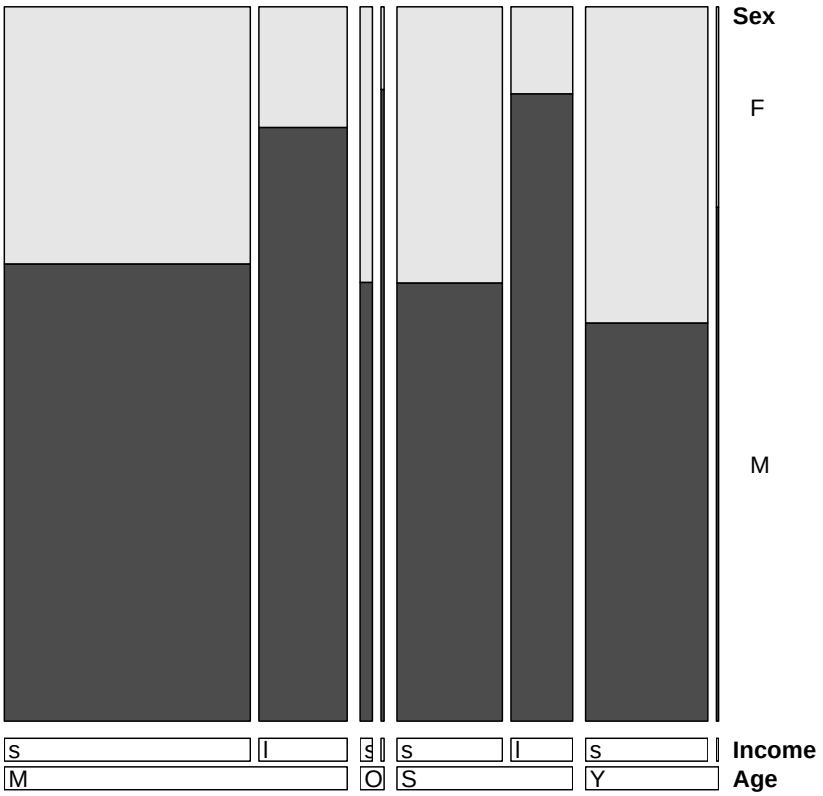


Abbildung 3.10: Double Decker Plot zum Mosaik Plot aus der Abbildung 3.9

Double Decker Plots sind Abbildungen, die ähnlich den Mosaik Plots sind. Der Unterschied dabei ist, dass die Fläche der Abbildung immer der Breite nach geteilt wird. Hervorgehoben wird jedoch, so wie bei den Mosaik Plots, vertikal [15].

Die Kombination von Attributen, die durch ein Kästchen dargestellt wird, kann aus der Beschriftung des Double Decker Plots gelesen werden (siehe Abbildung 3.10). Zum Beispiel gibt die erste Reihe die Kombination von $Income = small$, $Age = Middle - aged$ und $Sex = Male$. Dabei ist M im weißen Kästchen ganz unten links eine Abkürzung von *Middle-aged* und das kleine s oben von M steht für *small*. Ganz rechts kann man sehen, um welches Attribut es sich dabei handelt. Eine andere Möglichkeit anzuzeigen, welche Items in einer Kombination vorhanden sind, ist mit Hilfe von den Zahlen „0“ und „1“. Dabei würde 0 bedeuten, dass ein bestimmtes Item oder Itemset nicht vorhanden ist und 1, dass es vorhanden ist.

Um die Verhältnisse der hervorgehobenen Flächen leichter vergleichen zu können, ist es besser die so genannten *Double Decker Plots* zu verwenden [15]. Das sind Mosaik Plots, wobei die Fläche einer Abbildung immer nur vertikal geteilt wird (siehe Abbildung 3.10).

In diesem Abschnitt werden die *Double Decker Plots* präsentiert. Zu diesem Zweck wird an dieser Stelle in Anlehnung an [15] und [16] ein Beispiel vorgestellt. Der Datensatz AdultUCI wird jetzt vervollständigt, indem alle brauchbare Attribute hinzugefügt werden. Nach dem Aufbereiten des *AdultUCI* Datensatzes beinhaltet er 48842 Beobachtungen und 13 Attributen mit unterschiedlich vielen Ausprägungen [10]. Zum Beispiel sind *Sex* und *Income* solche Attribute. Ihre Ausprägungen sind *Female* und *Male* beziehungsweise *None*, *Low* und *High*. Insgesamt hat der Datensatz 115 Items. Die Beobachtungen können als Transaktionen und die einzelnen Items als Artikel angesehen werden. Durch diesen Datensatz können einige Eigenschaften gesehen werden, die für die Ergebnisse der Datenanalyse typisch sind. Zum Beispiel kann man an einem Doubledecker Plot die Qualität einer Regel beobachten.

3.4.1 Qualität einer Assoziationsregel

Betrachtet werden folgende Assoziationsregeln:

Regel	$conf.$	$supp.$
$\{Edct = HSGr,\ Incm = Smll\} \rightarrow \{HrPW = FllT\}$	12.04	66.62
$\{Edct = HSGr,\ HrPW = FllT\} \rightarrow \{Sex = Male\}$	13.48	64.50

Um die Qualität dieser Regeln verstehen zu können, wurden Double Decker Plots gebildet. Diese sind in Abbildungen 3.11 und 3.12 dargestellt. Diese Analyse passiert in Anlehnung an [15] und soll zeigen, wie man mittels Double Decker Plots *gute* von *schlechten* Assoziationsregeln unterscheiden kann.

Die erste Regel, $\{Edct = HSGr,\ Incm = Smll\} \rightarrow \{HrPW = FllT\}$ (ganz rechts in Abbildung 3.11), scheint gute zu sein. Keine andere Kombination von den beteiligten Artikeln weist eine größere Confidence auf.

Auf der anderen Seite stellt sich die zweite Assoziationsregel, $\{Edct = HSGr,\ HrPW = FllT\} \rightarrow \{Sex = Male\}$ (ganz rechts in Abbildung 3.12) als eine eher schwache. Es gibt zwei andere Kombinationen, die etwas größere Konfidenzwerte aufweisen. Und das, obwohl die zwei Regeln ungefähr die gleichen Werte für Support und Confidence aufweisen. Tatsächlich zeigt Abbildung 3.12, dass *hours-per-week=Full-Time* eine negative Wirkung auf die Regel hat. Dieses Double Decker Plot zeigt, dass 3 weitere Assoziationsregeln aus den Ergebnissen benötigt werden, um diese Regel bewerten zu können. In Abbildung 3.12 können folgende Assoziationsregeln mit einem Minimum Confidence von etwas weniger als 70% beobachtet werden:

$$\{not\ Edct = HSgr\ \&\ not\ HrPW = FllT\} \rightarrow \{sex = Male\}$$
$$\{Edct = HSGr\ \&\ not\ HrPW = FllT\} \rightarrow \{sex = Male\}$$

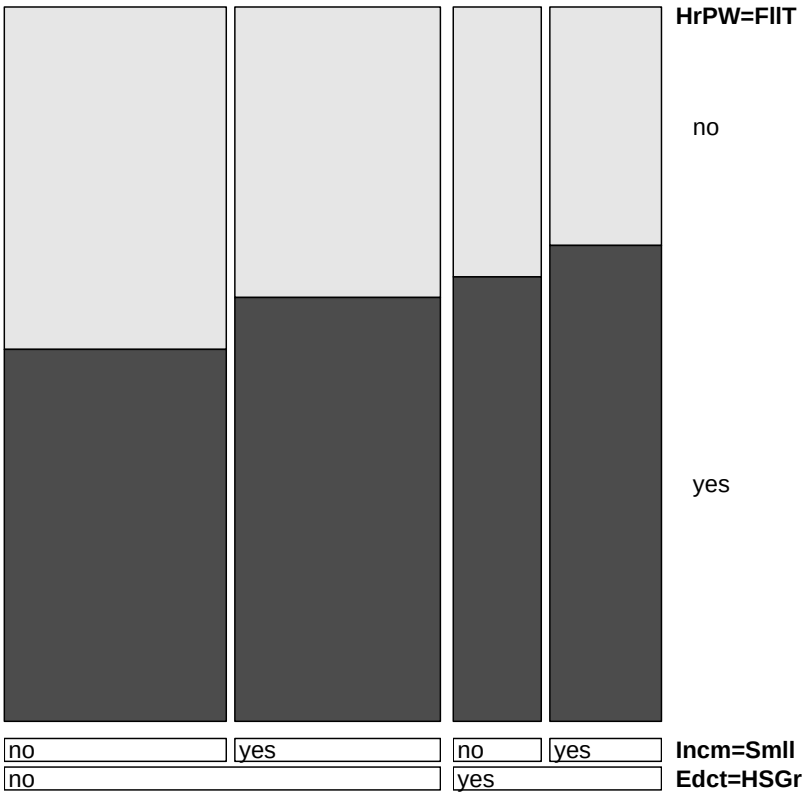


Abbildung 3.11: Beispiel einer *guten* Assoziationsregel

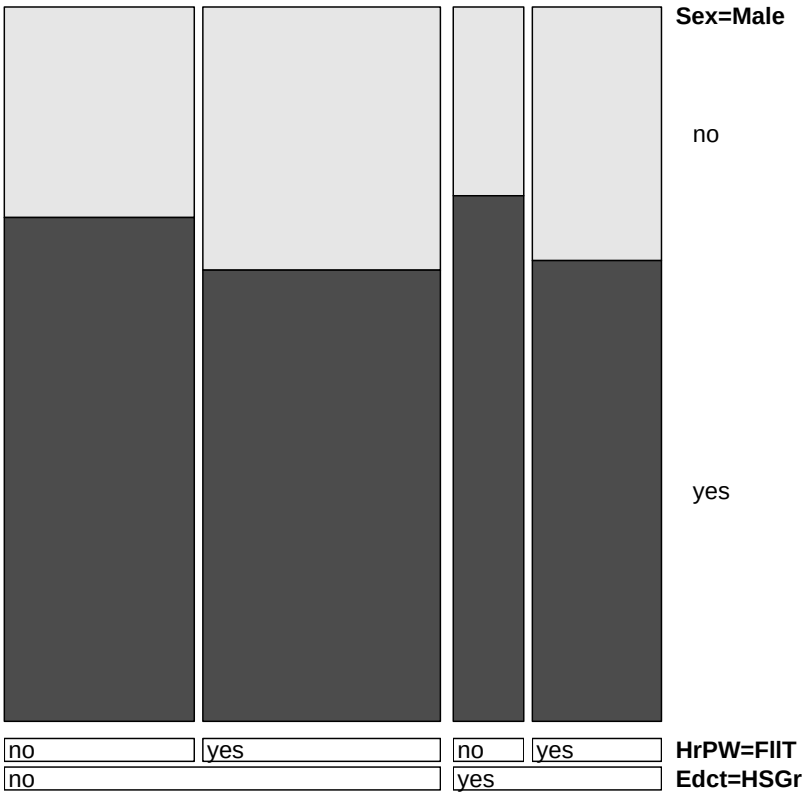


Abbildung 3.12: Beispiel einer *schlechten* Assoziationsregel

Genau so wie Assoziationsregeln sind auch Mosaik Plots (oder Double Decker Plots) nicht nur auf binäre Daten beschränkt. Daher können sie auch dann eingesetzt werden, wenn ein der Attribute des Regelrumpfs mehrere Ausprägungen hat [15]. In Abbildung 3.10 sieht man zum Beispiel, dass das Item *Age* vier verschiedene Ausprägungen hat (*middle-aged, old, senior* und *young*).

3.4.2 Differenz der Confidence

Jetzt soll eine weitere Eigenschaft der Double Decker Plots dargestellt werden. Sie steht im Zusammenhang mit einem Interessantheitsmaß, *Differences of Confidences (DOC)* (siehe Abschnitt 2.4.3). An dieser Stelle soll nur wiederholt werden, wann eine Assoziationsregel gemessen an DOC gut ist. Dieses Maß kann Werte zwischen -1 und 1 annehmen. Umso größer der Wert ist desto stärker eine Assoziationsregel [16]. Für eine Regel $X \rightarrow Y$ ist DOC:

$$doc(X \rightarrow Y) = conf(X \rightarrow Y) \text{ - } conf(\neg X \rightarrow Y)$$

Differences of Confidences kann in einem Double Decker Plot direkt gesehen werden (siehe Abbildung 3.13).

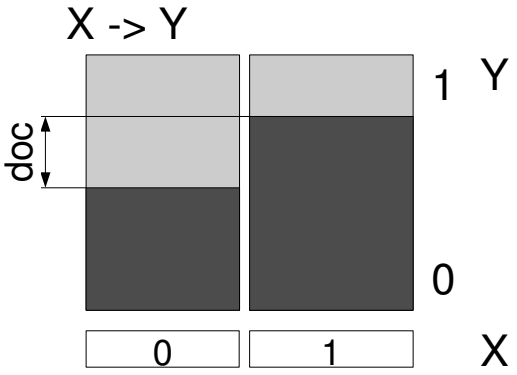


Abbildung 3.13: Difference of Confidence

Der Support von $X \rightarrow Y$ ist direkt proportional zu dem hervorgehobenen Rechteck auf der rechten Seite, seine Höhe gibt die Confidence von $X \rightarrow Y$ an. Ähnlich gibt die Höhe des hervorgehobenen Rechteck auf der linken Seite die Confidence von $\neg X \rightarrow Y$. Die Differenz von beiden gibt DOC an [16].

Tabelle 3.4 zeigt sechs Assoziationsregeln geordnet nach DOC. Diese Regeln sind in Abbildung 3.14 dargestellt. Die ersten zwei Regeln weisen eine relativ große DOC auf. Allerdings sind Assoziationsregeln von der Form $X = 0 \rightarrow Y = 1$ unbedeutend. Daher *gehen* die DOC

	Regelrumpf	Regelkopf	Support	Confidence	DOC
1	{CptG=None}	{Incm=Larg}	0.13	0.14	−0.28
2	{Race=Whit}	{Sex=Feml}	0.27	0.31	−0.14
3	{Sex=Male}	{CptL=None}	0.63	0.95	−0.02
4	{Incm=Smll}	{Edct=SmCl}	0.12	0.24	0.03
5	{NtvC=UntS}	{Race=Whit}	0.79	0.88	0.23
6	{Rltn=OwnC}	{MrtS=NvrM}	0.14	0.89	0.66

Tabelle 3.4: Sechs Assoziationsregeln geordnet nach DOC

der ersten zwei Regeln in Abbildung 3.14 eher in die „falsche" Richtung [16]. Das bedeutet, dass Confidence von $\neg\ X \rightarrow Y$ größer als die von $X \rightarrow Y$ ist. Die mittleren zwei Regeln weisen kaum DOC auf. Das deutet eher auf schwache Regeln hin. Die letzten zwei Assoziationsregeln zeigen DOC, die jetzt in die „richtige" Richtung gehen.

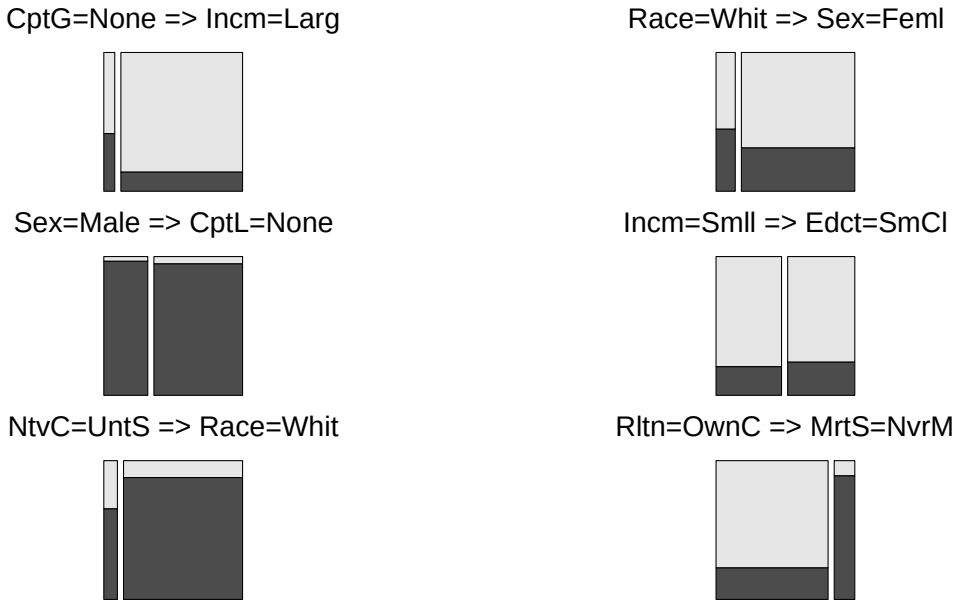


Abbildung 3.14: Difference of Confidence - Sechs Regeln geordnet nach DOC

3.4.3 Überschneidung zweier Assoziationsregeln

Es soll eine weitere Eigenschaft der Double Decker Plots untersucht werden. Wenn zwei Assoziationsregeln sich den Regelkopf teilen, etwa so $X_1 \rightarrow Y$ und $X_2 \rightarrow Y$, ist es durchaus möglich, dass beide Regeln einen großen Teil derselben Gruppe beschreiben. Nun soll an dieser Stelle in Anlehnung an [16] gezeigt werden, wie Mosaik Plots beziehungsweise Double Decker Plots verwendet werden können, um die zugrunde liegenden Strukturen zu analysieren. Dies geschieht, indem drei Abbildungen für jede der Regeln $X_1 \rightarrow Y$ und $X_2 \rightarrow Y$ und ihre Überschneidung $X_1 \wedge X_2 \rightarrow Y$ gebildet werden.

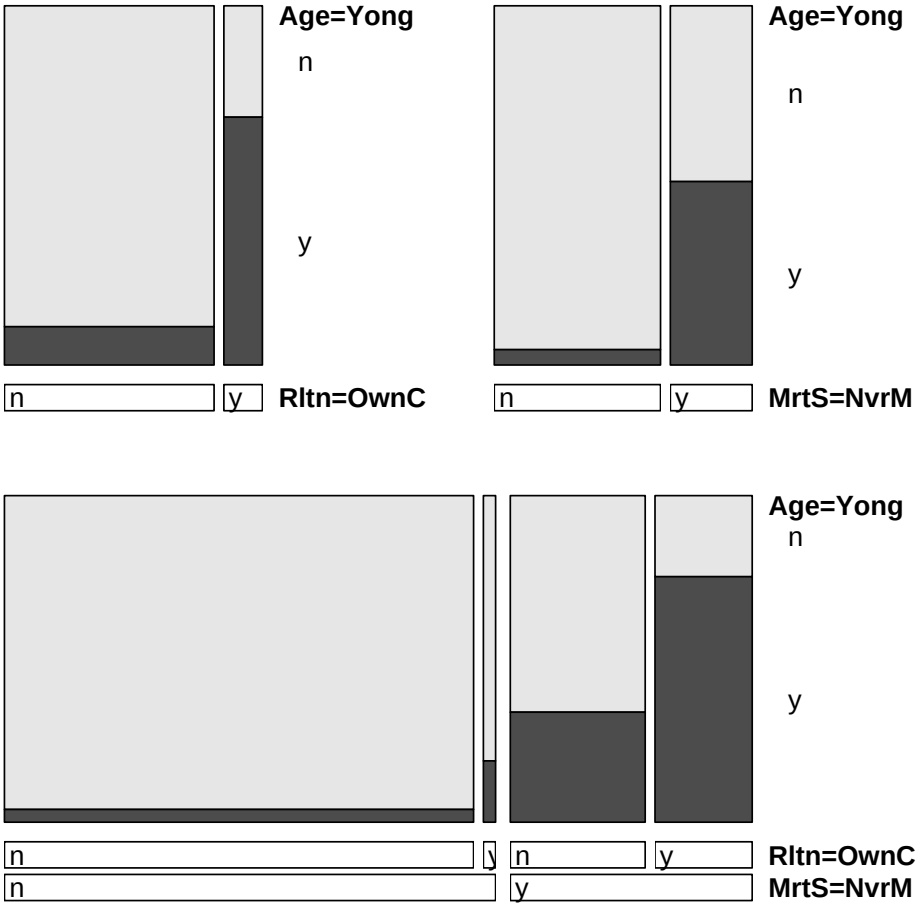


Abbildung 3.15: Überschneidung von zwei Assoziationsregeln

Es werden folgende zwei Regeln beobachtet:

	Regel		*conf.*	*supp.*
$\{Rltn = OwnC\}$	$\rightarrow$	$\{Age = Yong\}$	*10.71*	*68.99*
$\{MrtS = NvrM\}$	$\rightarrow$	$\{Age = Yong\}$	*16.85*	*51.06*

Abbildung 3.15 zeigt in der oberen Reihe die Double Decker Plots dieser Assoziationsregeln und in der unteren Reihe ihre Überschneidung. Von der unteren Reihe wird ganz klar, dass ein großer Teil der Leute die nie verheiratet waren ein eigenes Kind sind. In der Tat sind etwas mehr als 75% der jungen Leute($\{Age = Yong\}$), die sowohl ein Kind sind($\{Rltn = OwnC\}$) als auch nie verheiratet waren($\{MrtS = NvrM\}$). Daraus lässt sich schließen, dass die ersten zwei Regeln nicht unabhängig sind.

3.5 Gerichtete Graphen

Bei dieser Visualisierungstechnik werden die Items oder Itemsets als Knoten und die Assoziationen als Kanten dargestellt. Die Gewichtung der Kanten kann dabei entweder Support oder Confidence abbilden. Abbildung 3.16 veranschaulicht dies in Anlehnung an [19]. Hier entsprechen die Gewichtungen dem Support.

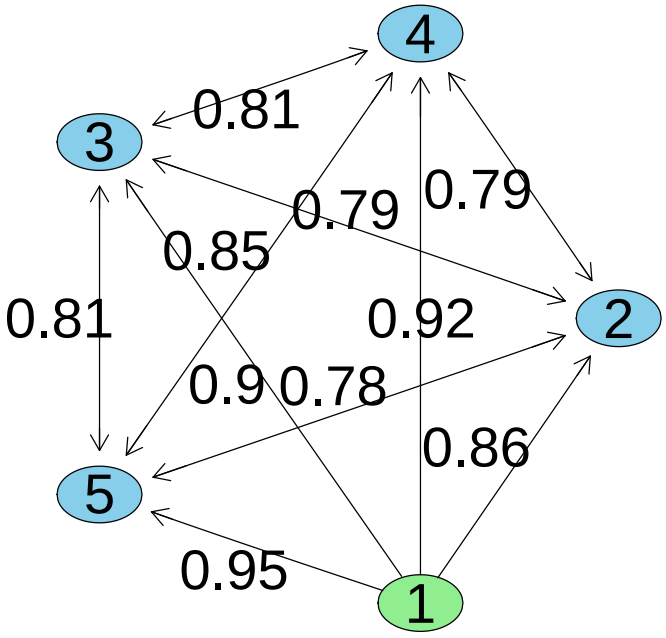


Abbildung 3.16: Gerichtete Graphen

Ein wesentlicher Nachteil dieser Visualisierungstechnik ist, dass mit steigender Anzahl an Knoten, die hier die einzelnen Assoziationsregeln darstellen, die Visualisierung unüberschaubar wird. Dadurch wird auch die Interpretation bedeutend beeinträchtigt [6].

Ein Vorteil dieser Technik besteht darin, dass gleichzeitig mehrere Assoziationsregeln abgebildet werden können. Andere Merkmale und zusätzliche Information können mittels Farbe dargestellt werden [19]. Zum Beispiel wird in Abbildung 3.16 die grüne Farbe dazu benutzt, um zu zeigen, dass ein bestimmter Knoten nur als Regelrumpf vorkommt. Auf der anderen Seite deutet die blaue Farbe darauf hin, dass diese Knoten sowohl Regelrumpf als auch Regelkopf sein können. Die Nummer eines jeden Knoten entspricht dabei einem bestimmten Itemset. Da aber diese aus mehrere Itemsets bestehen können, könnte dies zur Unübersichtlichkeit führen.

Mittels gerichteten Graphen können auch häufige Itemsets visualisiert werden. Abbildung 3.17 zeigt dies in Anlehnung an [7]. Insgesamt wurden aus dem *AdultUCI* Datensatz 67

häufige Itemsets mit einem Mindestsupport von 0.45 generiert. In der obersten Reihe bilden die Knoten alle häufige Itemsets mit Länge 1 ab, in der zweiter Reihe sieht man alle Itemsets mit Länge 2 usw. Die Kante zwischen zwei Knoten zeigt, dass der obere Knoten eine Untermenge des unteren ist. Somit sieht man, dass Itemsets Nummer 66 und 58 das Itemset Nummer 1 bilden. Außerdem zeigt Abbildung 3.17, dass Itemset Nummer 67 zwar das Minimum an Support erreicht hat, aber keine weitere Itemsets bildet. Das erkennt man daran, dass keine Kante aus diesen Knoten ausgeht.

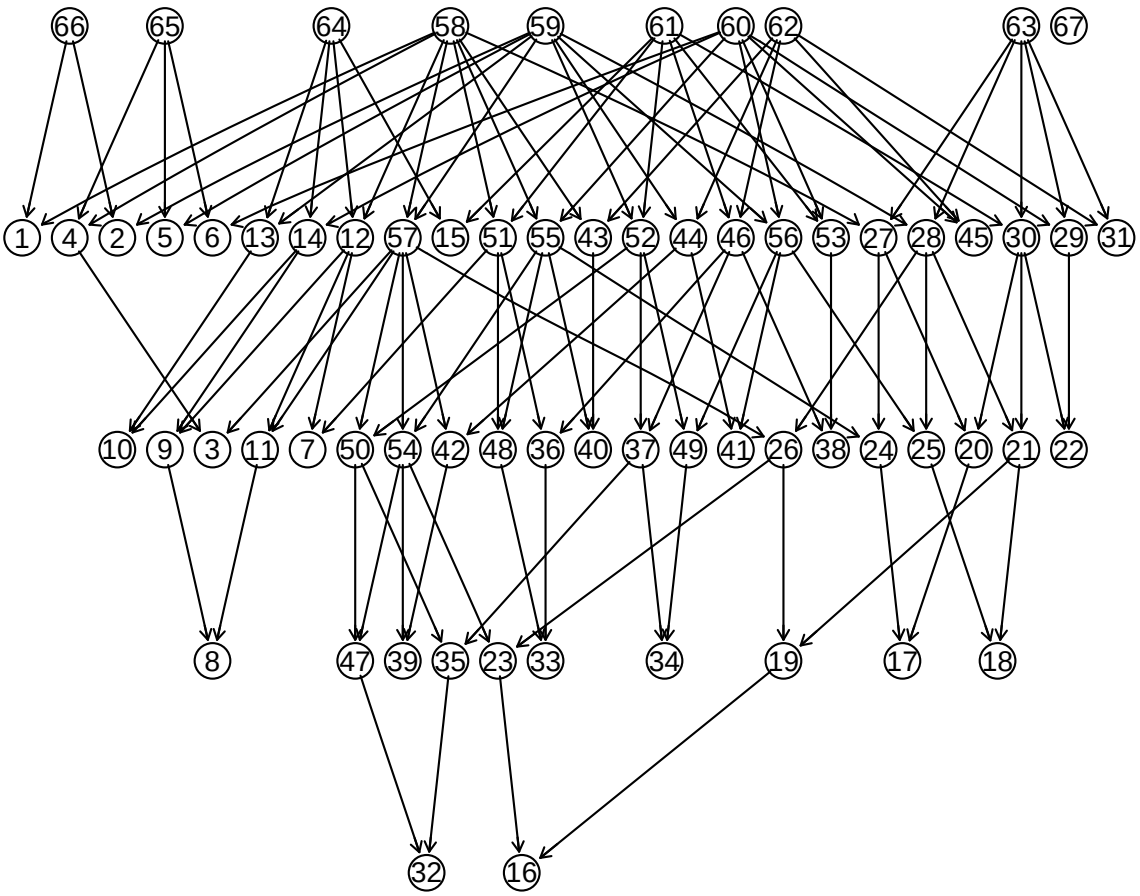


Abbildung 3.17: 67 Häufige Itemsets aus AdultUCI
Die erste Reihe von oben beinhaltet alle häufigen 1-Itemsets, die zweite alle häufigen 2-Itemsets usw.

3.6 Andere Visualisierungstechniken

In diesem Abschnitt werden weitere Visualisierungstechniken vorgestellt. Zuerst wird *VisAR* [35] dargestellt. Das ist eine Technik, die als ein Mischling zwischen Matrizen und gerichteten Graphen gesehen werden kann.

Eine andere Technik ist die Erforschung der entdeckten Assoziationsregeln als ein *OLAP* (On-line Analytical Processing) Problem [22]

3.6.1 VisAR

Der Prozess der Visualisierung im VisAR–System [35] wird in vier Stufen aufgeteilt. Diese sind:

1. Verwalten der Assoziationsregeln
2. Herausfinden interessanter Regeln
3. Visualisieren ausgewählter Regeln
4. Interaktive Visualisierung

In der ersten Stufe werden die Assoziationsregeln festgelegt und geladen. Dann werden sie nach dem Supportwert geordnet. Das Ziel in der zweiten Stufe ist anzugeben, welche Items präsentiert werden sollen. In der dritten Stufen werden diese Assoziationsregeln abgebildet, die die in Stufe zwei ausgewählten Items beinhalten. Die vierte Stufe erlaubt dem Anwender Interaktion. Dabei können Einzelheiten angezeigt werden.

Abbildung 3.18 zeigt das VisAR–System. Auf der linken Seite können Items gewählt werden, wobei es zwei Möglichkeiten gibt, die Assoziationsregeln anzuzeigen. Durch die Operationen *AND* oder *OR* kann bestimmt werden, ob Regeln angezeigt werden sollen, die *nur* die gewählten Items im Regelrumpf beinhalten oder aber auch Regeln, wo auch andere Items vorkommen. Regelrumpf und Regelkopf werden auf der Y–Achse dargestellt. Die Linie in der Mitte grenzt die Regelrumpfitems von den Regelkopfitems ab, wobei die ersten oberhalb der Linie stehen. Die Assoziationsregeln werden entlang der X–Achse abgebildet. In Abbildung 3.18 sind die Items *cd*, *rice*, *battery*, *soya sauce* und *sweets* im Regelrumpf. Die Items *newspaper*, *battery*, *soya sauce* und *sweets* sind im Regelkopf der Regeln. Eine Assoziationsregel wird durch eine parallele zu der Y–Achse Linie abgebildet, wobei die Punkte die Items in einer Regeln darstellen.

Zum Beispiel präsentiert die erste vertikale Linie in Abbildung 3.18 eine Assoziationsregel mit fünf Items. Die ersten vier Punkten sind im Regelrumpf und präsentieren die Items *cd*, *rice*, *battery* und *soya sauce*. Der letzte Punkt, *newspaper*, stellt den Regelkopf. Somit wird durch diese erste Linie die Assoziationsregel $\{cd,\ rice,\ battery,\ soya\ sauce\} \rightarrow \{newspaper\}$ dargestellt.

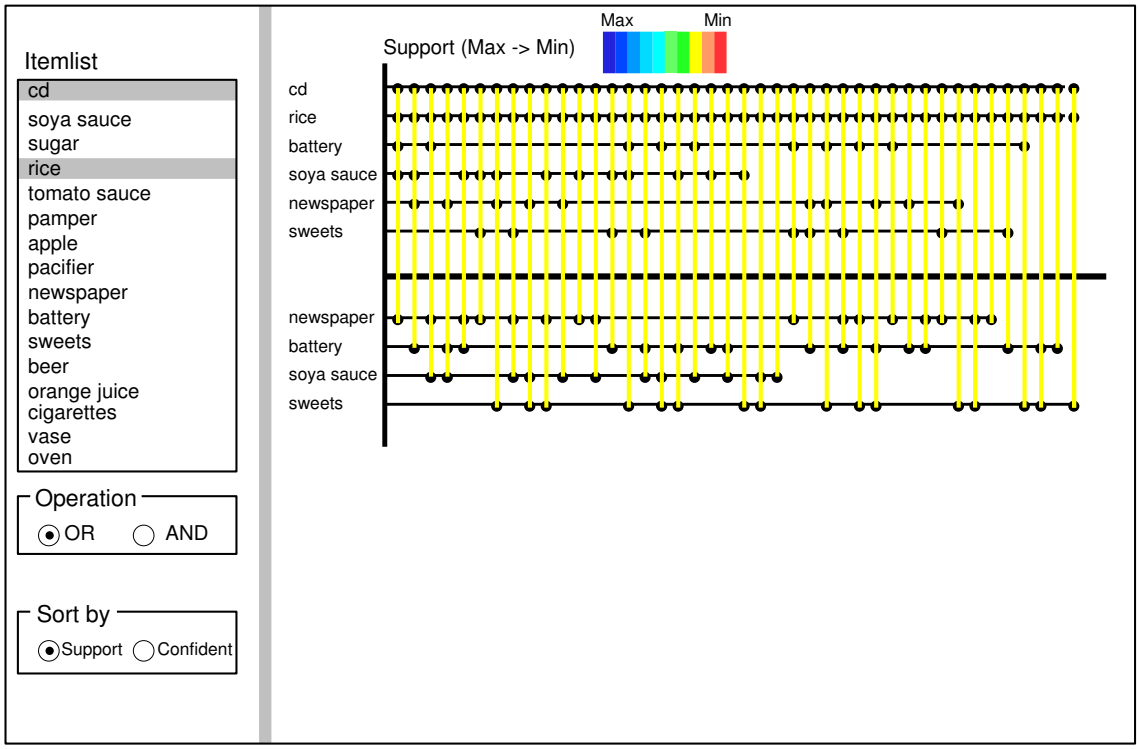


Abbildung 3.18: Das VisAR System
in Anlehnung an [35]

Die Regeln in Abbildung 3.18 sind nach dem Supportwert geordnet. Im VisAR werden Farben verwendet, um entweder den Support oder die Confidence anzugeben. Es werden zehn verschiedene Farbe benutzt. Diese geben jeweils ein 10% Intervall, wobei dunkel blau den höchsten Wert (90–100%) zum Ausdruck bringen soll und rot den kleinsten (0–10%). In Abbildung 3.18 sind alle Regeln im Bereich 20–30%. In Abbildung 3.18 wird Confidence dargestellt.

Abbildungen 3.19 und 3.20 zeigen Assoziationsregeln, die nur die Items *cd* und *rice* im Regelrumpf haben. Bei der ersten Abbildung sind die Regeln nach der Confidence und in der zweiten nach Support geordnet. Einige Vorteile dieses Systems sollen noch erwähnt werden. Es geht darum, dass viele Regeln gleichzeitig dargestellt werden können. Dabei können sowohl Regelrumpf als auch Regelkopf mehrere Items beinhalten. Durch die Anwendung von Farben können Assoziationsregeln mit ähnlich großen Confidence oder Support leicht wahrgenommen werden. Es können auch nur bestimmte Items gewählt werden. Dadurch können sich Anwender nur auf diese Regeln konzentrieren.

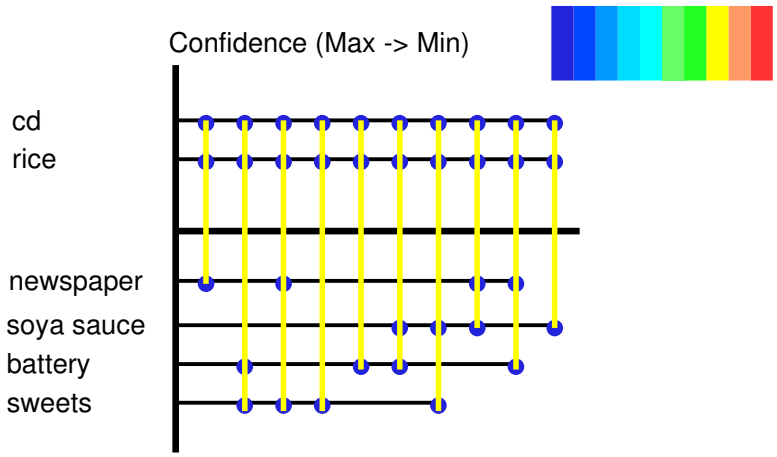


Abbildung 3.19: VisAR Quelle
in Anlehnung an [35]

Ein Vorteil dieses Systems ist unter anderem die Tatsache, dass es erlaubt, Assoziationsregeln mit mehreren Items sowohl im Regelrumpf als auch im Regelkopf zu visualisieren. Außerdem kann der Anwender wählen, welche Items visualisiert werden sollen [35].

Auf der anderen Seite können nur diese Regeln abgebildet werden, die die davor gewählten Items beinhalten. In diesem System werden nur zehn Farben zur Abbildung von Confidence oder Support verwendet. Dadurch kann man eine Regel mit einem Supportwert von 20.01 % von einer mit 29.99 % nicht unterscheiden.

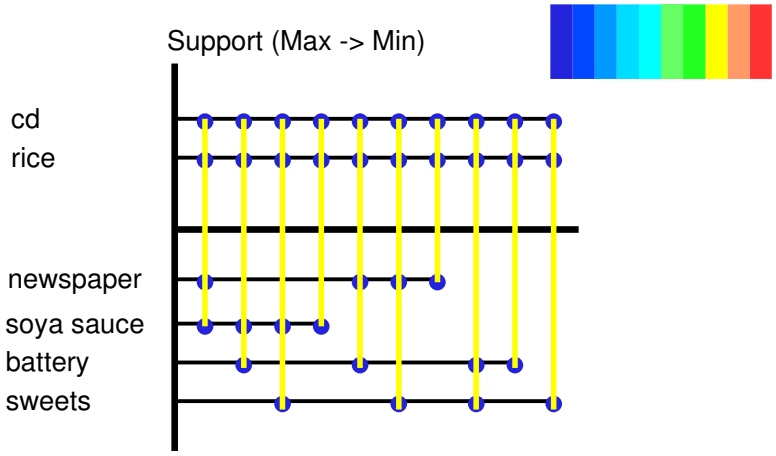


Abbildung 3.20: Das VisAR System
in Anlehnung an [35]]

3.6.2 OLAP

Liu et al. [22] gehen davon aus, dass Assoziationsregeln mit mehr als zwei Attributen im Regelrumpf nicht verwendbar sind. Aufgrund langjähriger Erfahrung können aus solchen Regeln keine Maßnahmen abgeleitet werden. Liu et. al sagen auch, dass durch die Anwendung von Minimalwerten, wie Support und Confidence, bei der Gewinnung von Assoziationsregeln zur Informationsverlust führen könnte.

Deswegen zeigen sie, dass die Analyse der Assoziationsregeln als ein *OLAP* (On-line Analytical Processing) Problem betrachtet werden kann. Dabei werden so genannte *Würfel* verwendet. Liu et at. [22] nennen diese Würfel Regel-Würfel. Dadurch können bekannte Techniken aus *OLAP* verwendet werden. Diese Techniken sind:

- Ausschneiden (engl. *slicing*). Es wird angenommen, dass ein Datensatz aus drei Items besteht, die jeweils mehrere Ausprägungen haben können. Ausschneiden bedeutet, dass eine der Dimensionen auf nur eine Ausprägung reduziert wird [24]. Zum Beispiel wird *Work Time* = *workaholic* gesetzt.

- Würfeln (engl. *dicing*). Hier werden zwei oder mehr Dimensionen auf zwei oder mehr Ausprägungen reduziert. Dadurch entsteht ein kleiner Würfel.

- Hereinzoomen[2] (engl. *drilling down*). Durch diese Operation bietet sich eine detaillierte Betrachtungsweise auf den Datensatz. Zum Beispiel möchte man wissen, wie sich das Item *Work Time* zusammensetzt.

[2]Siehe: http://de.wikipedia.org/wiki/Drill-Down

- Herauszoomen[3] (engl. *rolling up*). Das ist das Gegenteil von Hereinzoomen. Dadurch enthält man einen allgemeinen Überblick über den Datensatz. Zum Beispiel können die Ausprägungen des Items *Age*, *young, middle-aged, senior* und *old*, wieder zu einer Gruppe zusammengefasst werden. Also die Dimension *Age* wird nicht betrachtet.

Ein Beispiel soll dies in Anlehnung an [22] veranschaulichen. Zu diesem Zweck wird der Datensatz *AdultUCI* auf drei Attribute reduziert, wobei eines davon das Klassen-Attribut ist, *Sex*, und hat nur zwei Ausprägungen, *Female* und *Male*. Die anderen Attributen sind *Work time* und *Age*. *Work time* hat vier mögliche Ausprägungen (*part time, full time, over time* und *workaholic*), *Age* hat auch vier Ausprägungen (*young, middle, senior* und *old*). Der Datensatz besteht aus 48842 Aufzeichnungen. Abbildung 3.21 zeigt einen Regel-Würfel mit 32 (4 $Items$ x 4 $Items$ x 2 $Items$) Assoziationsregeln.

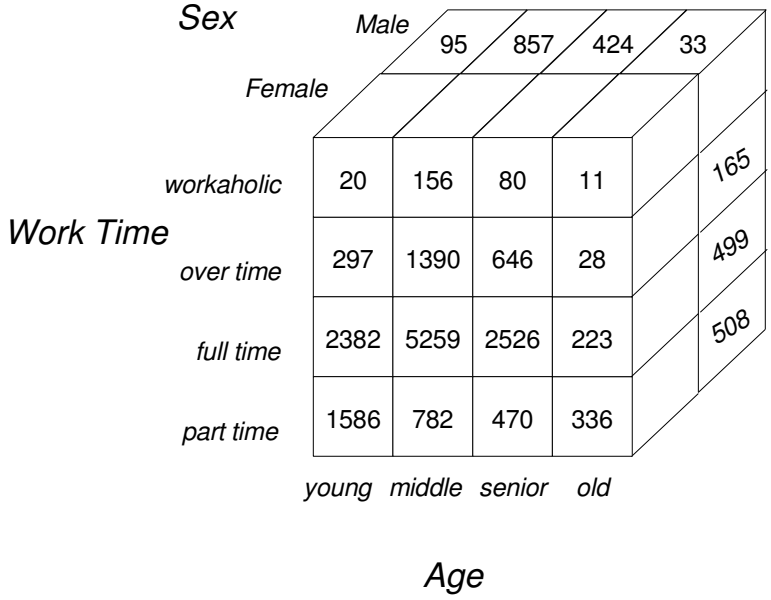


Abbildung 3.21: 3D Regel-Würfel

Zum Beispiel hat die Assoziationsregel

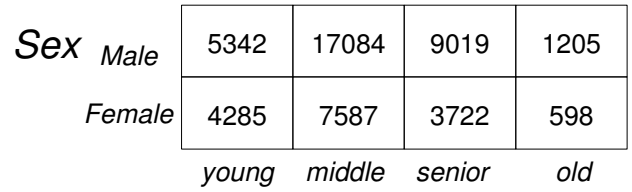
$$\{WrkT = Wrkh,\ Age = Yong\} \rightarrow \{Sex = Feml\}$$

Support von 20/48842 und Confidence von 20/(20+95). Daher ist es einfach, diese Kennzahlen zu berechnen.

Nun sollen jetzt die einzelnen Operationen kurz vorgestellt werden. Durch das Herauszoomen oder Roll-Up wird eine Gruppierung durchgeführt. Zum Beispiel können die Ausprägungen von Attribut *Work time part time* und *full time* zu einer Gruppe *normal* und

[3]Siehe: http://de.wikipedia.org/wiki/Drill-Down

die Ausprägungen *over time* und *workaholic* zu einer Gruppe *unusual* zusammengefasst werden. Dies kann auf Basis bestehenden Wissen passieren. Eine andere Art von Roll-Up wäre, wenn die ganze Dimension entfernt wird. Wenn nun das Attribut *Work time* entfernt wird, entsteht der Unter-Würfel in Abbildung 3.22. Dort sind auch die Assoziationsrgelen abgebildet, die sich durch Diese Operation ergeben.



{Age=young} => {Sex=Female}
{Age=middle} => {Sex=Female}
{Age=senior} => {Sex=Female}
{Age=old} => {Sex=Female}

{Age=young} => {Sex=Male}
{Age=middle} => {Sex=Male}
{Age=senior} => {Sex=Male}
{Age=old} => {Sex=Male}

Abbildung 3.22: Unter-Würfel nach einer Roll-Up Operation (*Worktime* entfernen) sowie dazugehörigen Regeln

Das Hereinzoomen oder Drill-Down ist das Gegenteil von Herauszoomen. Wenn der Würfel jetzt nur zwei Dimensionen hat, wird durch Hereinzoomen detaillierte Information abgebildet. Würde man ausgehend von Abbildung 3.22 eine weitere Dimension hinzufügen wollen, zum Beispiel *Work time*, würde ein Regel-Würfel wie in Abbildung 3.21 entstehen.

Durch Ausschneiden oder Slice wird eine Dimension ausgeschnitten. Beispielsweise entsteht der Unter-Würfel in Abbildung 3.23, wenn *Work Time = workaholic*. Wieder sind die dazugehörigen Assoziationsregeln abgebildet.

Das Würfeln oder Dice lässt einen kleineren Würfel entstehen, da zwei oder mehr Dimensionen ausgewählt werden. Zum Beispiel wird folgende Selektion gemacht: (*Worktime = workaholic* oder *Worktime = over time*) und (*Age = young* oder *Age = middle*). Dadurch entsteht der Würfel in Abbildung 3.24.

Es existieren auch andere *OLAP* Operationen, allerdings reichen diese aus, um interessantes Wissen zu finden. Liu et al. zählen einige Gründe auf, warum die Analyse mittels *OLAP* Operationen hilfreich ist:

- Assoziationsregeln können in einem Zusammenhang beobachtet werden, da alle wichtige Regeln in demselben Würfel enthalten sind.

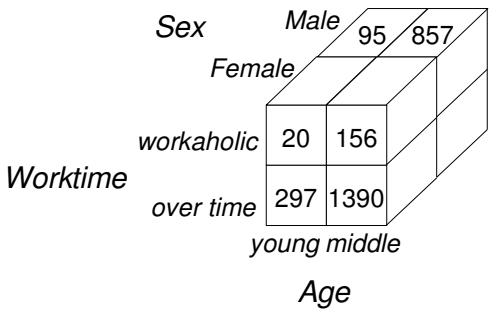
Abbildung 3.23: Unter-Würfel nach einer Slice Operation($Worktime = workaholic$) sowie dazugehörigen Regeln

- Dieses System beruht auf Matrizen, die für die Visualisierung von zwei und drei dreidimensionalen Würfeln geeignet sind
- Die *OLAP* Operationen erlauben den Anwender die entdeckten Regeln systematisch zu erforschen.

Die Visualisierung der Assoziationsregeln in diesem Modell basiert auf Matrizen. Es gibt zwei Modi, einen allgemeinen und einen detaillierten. Im Allgemeinen werden auf der Y-Achse die Klassen abgebildet und auf der X-Achse die Items (Abbildung 3.25). Für jedes Item (eine Spalte) zeigt jedes Gitter alle Regeln, die eine Bedingung haben. Die einzelnen Assoziationsregeln werden dabei als kleinen Balken dargestellt. Die Höhe eines Balkens gibt die Confidence einer Regel an. Man sieht in der obersten Reihe der Abbildung 3.25 die Verteilung der Ausprägungen der Items. Dadurch kann man sehen, dass es mehr Leute gibt, die Überstunden machen (O steht für *Over time*) als solche, die Teilzeit arbeiten (P steht für *Part time*). Außerdem fällt auf, dass an dieser Studie relativ wenig alte Leute teilgenommen haben. Die Plots auf der linken Seite zeigen das Verhältnis zwischen Männer und Frauen. Die Balkendiagramme in der zweiten Reihe sollen die Werte der Konfidenzen der Regeln mit $\{Sex = Female\}$ im Regelkopf zum Ausdruck bringen. Die dritte Reihe gibt dieselbe Information wie die zweite allerdings für die Regeln mit $\{Sex = Male\}$ in Regelkopf.

Der allgemeine Modus zeigt somit drei Eigenschaften:

1. Die Verteilung der Ausprägungen jedes Items wird in der obersten Reihe abgebildet und für die Klasse sind es die zwei Plots ganz links



{WrkT=Wrhc, Age=yng} => {Sex=Fem} *{WrkT=Wrhc, Age=yng} => {Sex=Male}*
{WorkT=Wrhc, Age=midd} => {Sex=Fem} *{WorkT=Wrhc, Age=midd} => {Sex=Male}*
{WrkT=OvrT, Age=yng} => {Sex=Fem} *{WrkT=OvrT, Age=yng} => {Sex=Male}*
{WrkT=OvrT, Age=midd} => {Sex=Fem} *{WrkT=OvrT, Age=midd} => {Sex=Male}*

Abbildung 3.24: Regel-Würfel nach dem Ausschneiden ($Worktime = workaholic$ oder $Worktime = over\ time$) und ($Age = young$ oder $Age = middle$) und dazugehörige Regeln

2. Assoziationsregeln mit einer Bedingung werden für alle Items visualisiert. Dabei werden auch alle anderen Regeln für dieses Item auch abgebildet (entlang der X-Achse) sowie alle andere Klassen (entlang der Y-Achse).

3. Tendenzen können leicht entdeckt werden. Allerdings hängen diese Tendenzen davon ab, wie die Items sortiert wurden (zum Beispiel alphabetisch).

Der detaillierte Modus soll dem Anwender helfen, die Assoziationsregeln weiter zu erforschen. Abbildung 3.26 soll dies veranschaulichen. Sie bildet die Konfidenzen der Regeln ab, die durch die Operation Roll-Up entstanden sind (siehe Abbildung 3.22). Zum Beispiel sieht man, dass die graue Fläche im Schnittpunkt von *Male* und *Middle* etwa 2/3 der Gesamtfläche des Kästchen annimmt. Das entspricht einer Confidence von etwa 66%. In der obersten Reihe sieht man auch die Verteilung der einzelnen Ausprägungen.

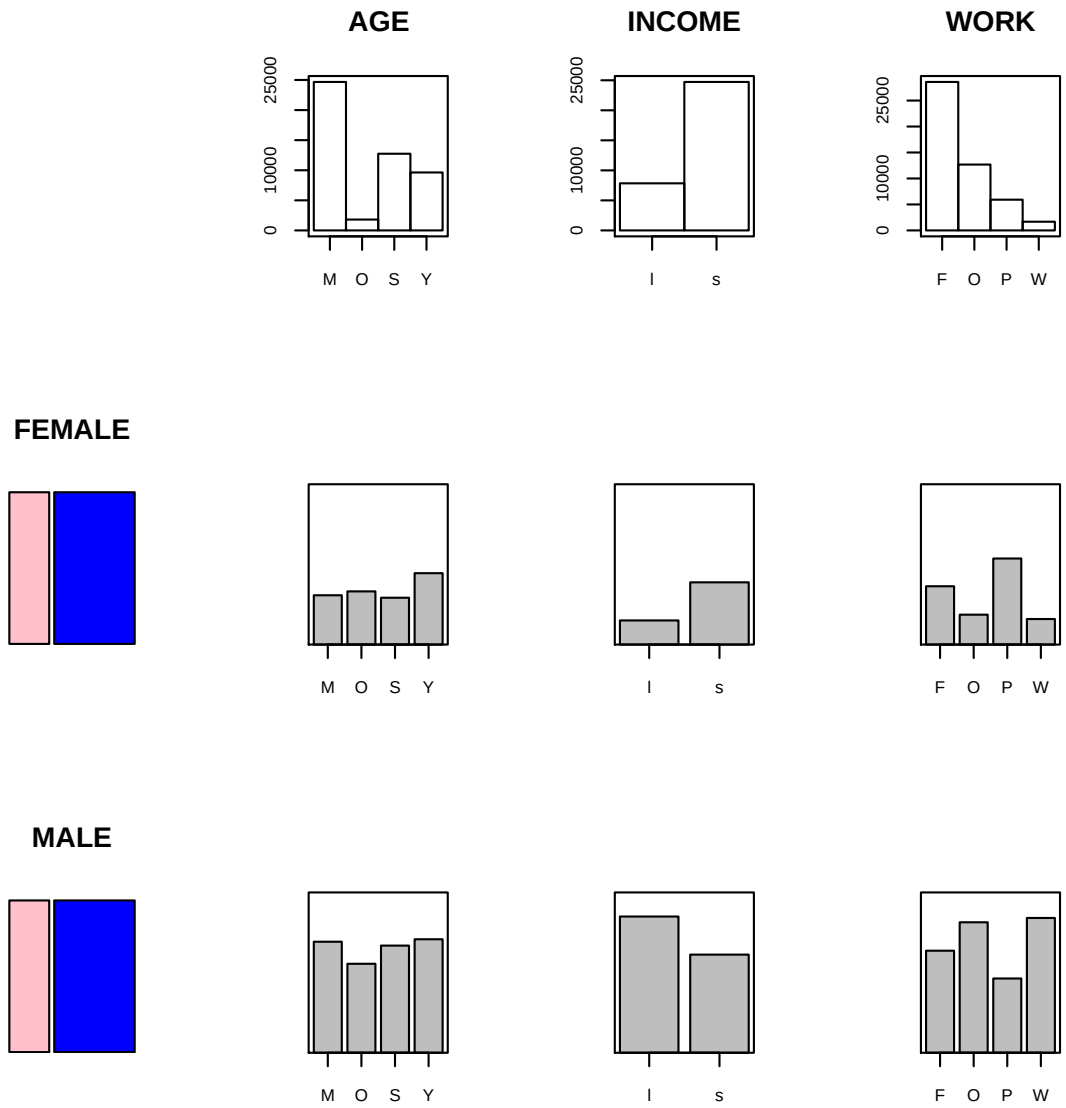


Abbildung 3.25: Regel-Würfel: Allgemeiner Modus
in Anlehnung an [22]

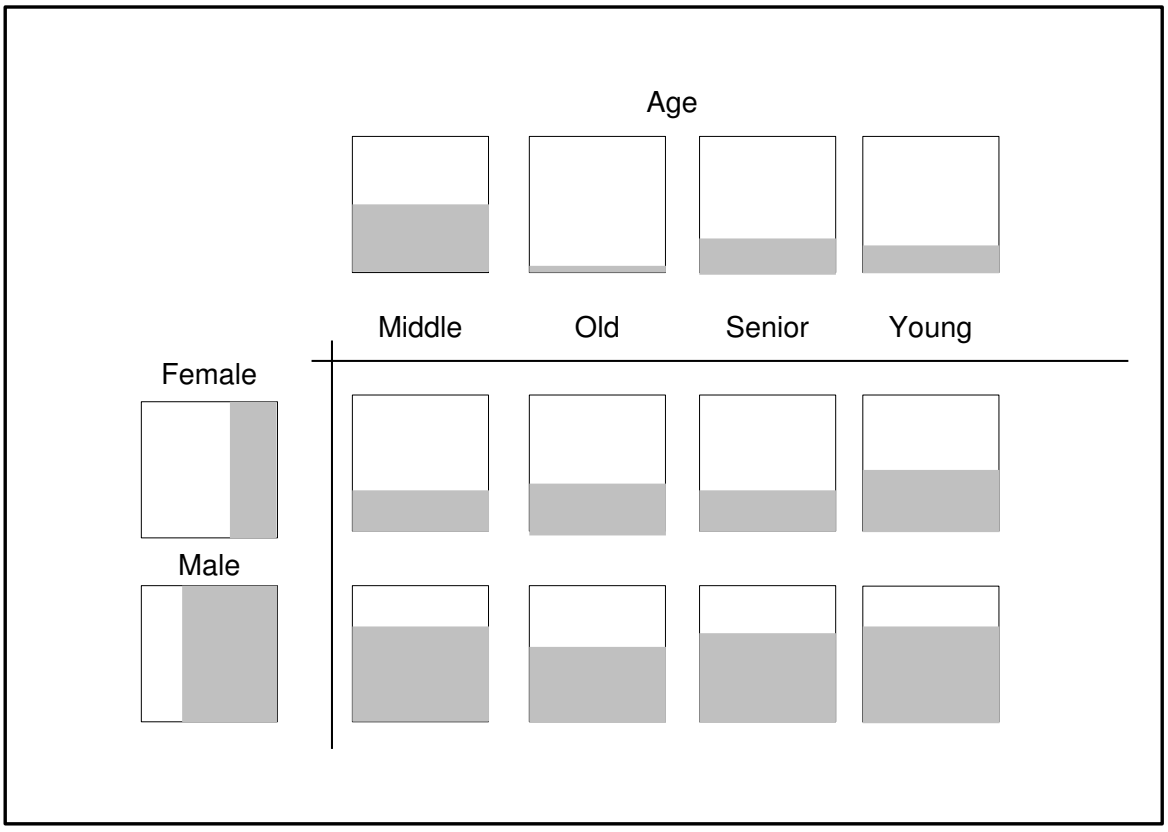


Abbildung 3.26: Regel-Würfel: Detaillierter Modus
in Anlehnung an [22]

Kapitel 4

Visualisierungstechniken mit R

In diesem Kapitel werden einige von den im vorherigen Kapitel vorgestellten Visualisierungtechniken mittels der Programmiersprache `R` implementiert. Zu diesem Zweck soll zuerst die Programmiersprache `R` vorgestellt werden. Dabei werden, neben den zahlreichen Möglichkeiten Daten zu präsentieren, einige für das Ziel dieser Arbeit sehr wichtige Pakete vorgestellt. Diese sind unter anderem `arules`, `vcd` und `Rgraphviz` (siehe dazu [10; 25; 9]).

Nach einer kurzen Vorstellung von `R` werden in Anlehnung an [10], [25] beziehungsweise [9] oben erwähnten Paketen diskutiert. Danach folgt ein Abschnitt über die Problemstellung und Implementierung. Es werden wichtige für die Implementierung bestimmter Visualisierungstechniken Eigenschaften analysiert. Abschließend werden bestimmte Techniken implementiert.

4.1 Programmiersprache R

R ist eine Sprache und Umgebung für statistische Berechnungen und deren Vizualisierung[31]. Sie ist eine objekt-orientierte Programmiersprache. R basiert hauptsächlich auf S. Sie wurde von John Chambers entwickelt. Mit dieser Sprache lassen sich viele Problemstellungen einfach lösen. R bietet eine Vielzahl von statistische und graphische Techniken, wobei diese auch erweiterbar sind. R kann kostenlos unter GNU GENERAL PUBLIC LICENSE[1] bezogen werden.

R ist ein vollständiges Softwarepaket zur Datenverarbeitung, Berechnungen und graphischen Visualisierung von Daten. Es umfasst unter anderem:

- Eine effektive Behandlung der Daten
- Eine Menge an Operatoren für Berechnungen mit Arrays
- Graphische Ausgabe der Berechnungen entweder auf dem Bildschirm oder auf der Festplatte
- Schnittstelle zu anderen Programmiersprachen (zum Beispiel C,C++, FORTRAN)
- R bietet auch die Möglichkeit Schleifen, Bedingugen und rekursive Funktionen zu verwenden.

Ein sehr gutes Einführungsbuch ist das Buch von Uwe Ligges [20]. Andere Literatur zu diesem Thema findet man auf der Homepage von R (http://www.r-project.org/) unter dem Punkt Manuals. Auf der Homepage findet man auch den Source Code für alle Betriebssysteme.

Für R sind Erweiterungspakete verfügbar. Das bedeutet, dass bestimmte Funktionen nach Funktionalität zusammengepackt werden und als Paket zur Verfügung gestellt werden. Es ist ein Vorteil dieser Programmiersprache, dass es bereits über 1000 zusätzliche Pakete gibt(Stand Juni 2007). Einige von diesen Paketen haben für die vorliegenede Arbeite eine essentielle Bedeutung.

Ein weiterer Vorteil der Programmiersprache R sind die so genannte *generische Funktionen*. Das bedeutet, dass eine Funktion sich unterschiedlich verhält, je nachdem was für eine Klasse das Objekt hat, mit dem sie aufgerufen wurde. Ein Beipiel stellt die Funktion summary() dar. Dazu kommen wir aber später noch.

4.2 Visualisierungen in R

R bietet eine Vielzahl an Möglichkeiten zur Generierung von Graphiken und deren Ausgabe. Grundsätzlich soll man zwischen zwei Typen von Funktionen in R unterscheiden. Die eine

[1] http://www.gnu.org/copyleft/gpl.html

sind so genannte *High–Level* Funktionen, die andere *Low–Level* [31]. Funktionen zur Visualisierung der ersten Gruppe(zum Beispiel `plot()`) generieren die Abbildungen in `R` und durch die Funktionen der zweiten Gruppe können diese Abbildungen zusätzlich bearbeitet werden(zum Beispiel eine Legende hinzufügen).

Außerdem unterscheidet man zwei Typen von Diagrammen. Die eine bauen auf Funktionalität vom Basispaket `graphics` (zum Beispiel `hist()`) auf. Die andere, und diese sind viel mächtiger, bauen auf Funktionalität vom Paket `grid` beziehungsweise `lattice`. Das Paket `grid` stellt dabei die Funktionen der *Low–Level* bereit (zum Beispiel `grid.xaxis()`). Das Paket `lattice` auf der anderen Seite stellt einige *High–Level* Funktionen zur Verfügung (zum Beipiel `levelplot()`).

4.2.1 Pakete arules, vcd und Rgraphviz

Diese drei Pakete bieten viele Funktionen, die es erlauben Assoziationsregeln zu generieren oder diese zu präsentieren. Das Paket `arules` bietet durch die Funktionen `apriori()` und `eclat()` die Möglichkeit Assoziationsregeln beziehungsweise häufige Itemsets zu generieren. Die Funktionen `mosaic()` und `doubledecker()`, die durch das Paket `vcd` bereitgestellt werden, bieten die Möglichkeit, einige von den im vorherigen Kapiteln vorgestellten Visualisierungstechniken, zu implementieren.

Durch das Paket `Rgraphviz` werden Graphen visualisiert, die durch das Paket `graph` generiert wurden. Dieses Paket ist eine Schnittstelle zwischen `R` und dem Programm `Graphviz`. Das ist ein Open Source Programm zur Visualisierung von strukturierte Information [2].

4.3 Problemstellung und Implementierung

Folgende Visualisierungstechniken sollen mit Hilfe von `R` implenetiert werden:

1. Dreidimensionalen Matrizen
2. Gitter
3. Mosaik Plots
4. Doubledecker Plots
5. Gerichtete Graphen

Bevor aber mit der eigentlichen Implentierung begonnen werden kann, müssen Assoziationsreglen gebildet werden. Dabei wird ihre Struktur analysiert.

[2] http://www.graphviz.org

4.3.1 Assoziationsregeln mit R

In Anlehnung an [10] wird die Generierung von Assoziationsregeln vorgeführt. Das Paket `arules` wird zuert geladen. Dabei werden auch andere Pakete mitgeladen, da `arules` auf diese aufbaut.

```
> library("arules")
```

Wenn das Paket geladen ist, wird der Datensatz `Adult` geladen. Dann wird die Funktion `class()` angewendet. Diese Funktion gibt die Klasse zurück, zu der das Objekt `Adult` gehört.

```
> data(Adult)
> class(Adult)

[1] "transactions"
attr(,"package")
[1] "arules"
```

Der Datensatz gehört zur Klasse `transactions`. Das bedeutet, dass Funktionen wie `apriori()` und `eclat()` gleich anwendbar sind. Außrdem kann man die Funktion `inspect()` anwenden.

```
> inspect(Adult[1])

  items
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}
```

Man sieht, wie die einzelnen Transaktionen aufgebaut sind. Der ersten Transaktion kann man entnehmen, dass es sich dabei um einen weißen Mann mit kleinem Einkommen handelt, der Vollzeit beschäftigt ist.

Es wird auch die Funktion `abbr3()` angewendet. Diese kürzt die Transaktionen ab. Die erste Transaktion wird zwei mal betrachtet (vor der Abkürzung und danach).

```
> Adult <- abbr3(Adult)

> inspect(Adult[1])

  items
1 {Age=MddA,
   Wrkc=SttG,
   Edct=Bchl,
   MrtS=NvrM,
   Occp=AdmC,
   Rltn=NtIF,
   Race=Whit,
   Sex=Male,
   CptG=Low,
   CptL=None,
   HrPW=FllT,
   NtvC=UntS,
   Incm=Smll}
```

Dann wird die Funktion `summary()` angewendet, um mehr Informationen über den Datensatz zu erhalten.

```
> summary(Adult)

transactions as itemMatrix in sparse format with
 48842 rows (elements/itemsets/transactions) and
 115 columns (items)

most frequent items:
CptL=None CptG=None NtvC=UntS Race=Whit Wrkc=Prvt   (Other)
    46560     44807     43832     41762     33906    401333

element (itemset/transaction) length distribution:
sizes
    9    10    11    12    13
   19   971  2067 15623 30162

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   9.00   12.00   13.00   12.53   13.00   13.00

includes extended item information - examples:
    labels variables levels
```

```
1 Age=Yong        Age   Yong
2 Age=MddA        Age   MddA
3 Age=Senr        Age   Senr
```

Der Datensatz besteht aus 48842 Transaktionen und wird repräsentiert als eine dünnbesetzte Matrix mit 48842 Zeilen und 115 Spalten, die die einzelnen Items repräsentieren. Man sieht unter anderem auch die häufigsten Items, die Verteilung der Länge der Transaktionen und die erweiterte Item-Information. Diese Information zeigt, welche Variablen und welche Ausprägung benutzt wurden, um ein Item zu erzeugen. Zum Beispiel enstand das Item $age = Young$ aus der Variable *age* mit der Ausprägung *Young*.

Als Nächstes wird die Function `apriori` mit aufgerufen. Diese Funktion generiert aus einem Datensatz Assoziationsregeln.

```
> rules <- apriori(Adult, parameter = list(support = 0.1,
+     confidence = 0.1))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen
        0.1    0.1    1 none FALSE            TRUE     0.1      1
 maxlen target   ext
      5  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 [0.16s].
sorting and recoding items ... [31 item(s)] done [0.02s].
creating transaction tree ... done [0.26s].
checking subsets of size 1 2 3 4 5 done [0.26s].
writing ... [8271 rule(s)] done [0.01s].
creating S4 object  ... done [0.08s].

> rules

set of 8271 rules
```

Zuerst werden die verwendeten Parameter ausgegeben. Abgesehen von den vorgegebenen Werten für Support und Confidence haben alle anderen Parameter ihre Standartwerte. Es

wurden 8271 Assoziationsregeln generiert. Es wurden nur Regeln mit maximaler Länge 5 erzeugt. Sollen längere Regeln generiert werden, so muss der Parameter `maxlen` höher gesetz werden. Um einen Überblick über die Regeln erhalten zu können, wird wieder die Funktion `summary()` angewendet.

```
> summary(rules)

set of 8271 rules

rule length distribution (lhs + rhs):sizes
   1    2    3    4    5
  31  370 1512 3028 3330

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   4.000   4.000   4.119   5.000   5.000

summary of quality measures:
    support         confidence          lift
 Min.   :0.1000   Min.   :0.1008   Min.   :0.6646
 1st Qu.:0.1188   1st Qu.:0.5324   1st Qu.:0.9882
 Median :0.1413   Median :0.8112   Median :1.0253
 Mean   :0.1765   Mean   :0.7130   Mean   :1.1388
 3rd Qu.:0.1994   3rd Qu.:0.9224   3rd Qu.:1.1011
 Max.   :0.9533   Max.   :1.0000   Max.   :3.9827
```

Die Ausgabe zeigt unter anderem die Verteilung der Länge der Regeln sowie Zusammenfassungen der einzelnen Interesantheitsmaße.

Mit Hilfe der Funktion `subset()` kann die Menge der erzeugten Regeln reduziert werden. Es werden zwei Untermengen erzeugt. Die eine beinhaltet alle Regeln mit `HrPW=Ovrt` im Regelkopf, die andere mit `HrPW=FllT`.

```
> rulesOverTime <- subset(rules, subset = rhs %in% "HrPW=OvrT")
> rulesFullTime <- subset(rules, subset = rhs %in% "HrPW=FllT")
```

Dann können jeweils die ersten drei Regeln beider Untermengen geordnet nach Confidence betrachtet werden.

```
> inspect(SORT(rulesOverTime, by = "confidence")[1:3])

  lhs            rhs           support confidence     lift
1 {Age=MddA,
   Race=Whit,
```

```
   Sex=Male,
   NtvC=UntS} => {HrPW=OvrT} 0.1113386  0.3987388 1.536384
2 {Age=MddA,
   Race=Whit,
   Sex=Male}  => {HrPW=OvrT} 0.1184022  0.3882772 1.496074
3 {Rltn=Hsbn,
   Race=Whit,
   Sex=Male,
   NtvC=UntS} => {HrPW=OvrT} 0.1312190  0.3877193 1.493924
```

```
> inspect(SORT(rulesFullTime, by = "confidence")[1:3])
```

```
  lhs            rhs           support confidence     lift
1 {Age=MddA,
   Sex=Feml,
   CptG=None} => {HrPW=FllT} 0.1018795  0.7005491 1.197334
2 {Age=MddA,
   Sex=Feml,
   CptL=None} => {HrPW=FllT} 0.1040498  0.6954975 1.188700
3 {Age=MddA,
   Sex=Feml}  => {HrPW=FllT} 0.1076737  0.6931594 1.184704
```

Daraus kann man entnehmen, dass eher Männer die Überstunden machen.

4.3.2 Dreidimensionale Matrizen

Um diese Visualisierungtechnik implementieren zu können, braucht man das Paket `scatterplot3d` [21]. Damit kann man multivariate Daten in einem dreidimensioanlen Raum abbilden. An dieser Stelle wird eine Funktion definiert, der man zwei Parameter übergibt. Die Funktion erzeugt dann die Abbildung. Zuerst wird das Paket geladen.

```
> library(scatterplot3d)
```

Dann wird eine Funktion `matrix3D()` definiert. Diese benötigt mindestens ein Argument der Klasse `rules`. Als zweites Argument kann angegeben werden, welches Interessantheitsmaß verwendet werden soll. Es muss ein Objekt der Klasse `character`. Standarteinstellung ist `support`. Andere Möglichkeiten sind `confidence` oder `lift`. Sollten andere Interessantheitsmaße definiert worden sein, können diese auch verwendet werden.

Wie bereits in Abschnitt **??** diskutiert wurde, eignet sich diese Technik nicht, wenn die generierten Assoziationsregeln viel sind. Zu diesem Zweck wird eine Untermenge der Regeln gebildet. Diese beinhaltet alle Regeln, die Support größer als 0.65 haben.

```
> subrules <- rules[quality(rules)$support > 0.65]
> subrules

set of 36 rules
```

Die so enstandenen Regeln werden an die Funktion `matrix3D()` übergeben. Sie nimmt sich die Namen der Elementen der linken und rechten Seite sowie den Supportwert. Daraus macht sie ein Objekt der Klasse `data.frame`, das sie an die Funktion `scatterplot3d` übergibt. Des Weiteren gibt sie den Namen der einzelnen Itemsets an.

```
> matrix3D <- function(rules, measure = "support") {
+     leftSideLabels <- labels(lhs(rules))$elements
+     rightSideLabels <- labels(rhs(rules))$elements
+     support <- quality(rules)[[measure]]
+     rulesASdataframe <- data.frame(Regrelrumpf = leftSideLabels,
+         Regelkopf = rightSideLabels, Support = support)
+     scatterplot3d(rulesASdataframe, lwd = 1.5, zlim = c(0,
+         max(support)), pch = "", type = "h", color = "blue",
+         lab = c((length(unique(leftSideLabels)) + 2),
+             (length(unique(rightSideLabels)) + 2)), lab.z = 6,
+         cex.axis = 1, cex.lab = 0.7, angle = 15, zlab = measure)
+     writeLines("Itemsets im Regelrumpf")
+     print(unique(leftSideLabels))
+     writeLines("Itemsets im Regelkopf")
+     print(unique(rightSideLabels))
+ }
```

Die Funktion wird nun aufgerufen.

```
> matrix3D(subrules)

Itemsets im Regelrumpf
 [1] "{}"
 [2] "{Wrkc=Prvt}"
 [3] "{CptL=None}"
 [4] "{Race=Whit}"
 [5] "{NtvC=UntS}"
 [6] "{CptG=None}"
 [7] "{Race=Whit,NtvC=UntS}"
 [8] "{Race=Whit,CptG=None}"
 [9] "{CptG=None,NtvC=UntS}"
[10] "{Race=Whit,CptL=None}"
```

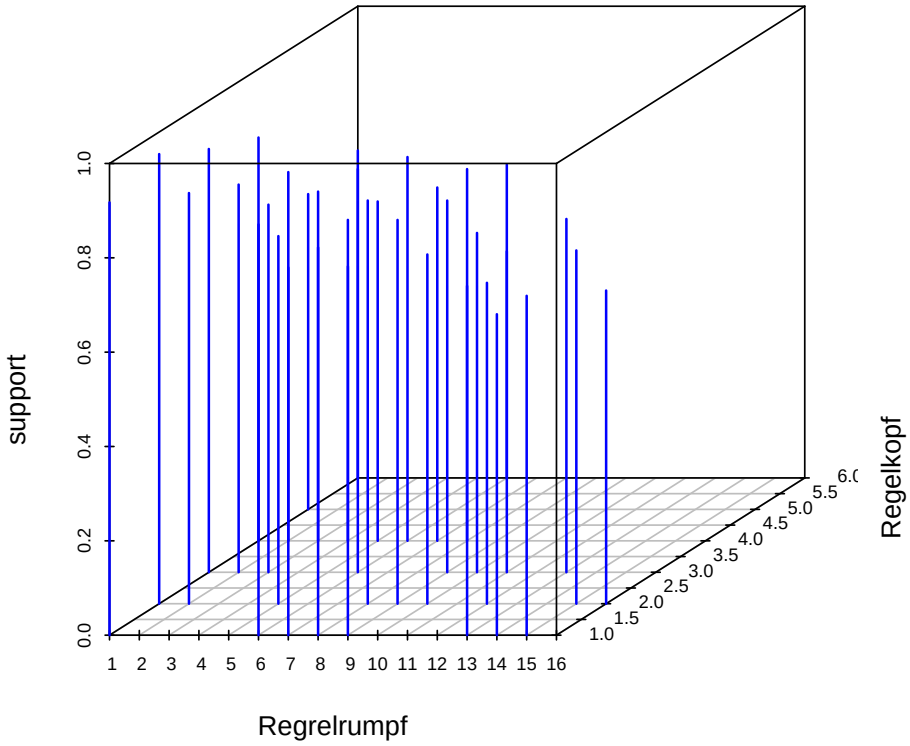


Abbildung 4.1: Assoziationsregeln als dreidimensionale Matrix

```
[11] "{CptL=None,NtvC=UntS}"
[12] "{CptG=None,CptL=None}"
[13] "{Race=Whit,CptG=None,NtvC=UntS}"
[14] "{Race=Whit,CptL=None,NtvC=UntS}"
[15] "{Race=Whit,CptG=None,CptL=None}"
[16] "{CptG=None,CptL=None,NtvC=UntS}"
Itemsets im Regelkopf
[1] "{Sex=Male}"  "{Wrkc=Prvt}" "{Race=Whit}" "{NtvC=UntS}"
[5] "{CptG=None}" "{CptL=None}"
```

Wie man in Abbildung 4.1 sieht, bestehen `subrules` aus 16 Itemsets im Regelrumpf und 6 im Regelkopf. Was man allerdings nicht sieht, ist welche Itemsets welche Nummer haben.Zu diesem Zweck muss man sich die Ausgabe in der Konsole anschauen. Auf der anderen Seite kann man erkennen, dass mit steigender Nummer der Itemsets im Regelrumpf (entlang der X-Achse) den Supportwert (die Höhe der blauen Balken in Abbildung 4.1) abnimmt. Daraus lässt sich schießen, dass das längere Itemsets sein sollen.

4.3.3 Gitter

Grundsätzlich gibt es in `R` zwei Funktionen, mit deren Hilfe Assoziationsregeln als Gitter abgebildet werden können. Die erste findet man im Paket `graphics` [31]. Das ist die Funktion `image()`. Die zweite Funktion ist `levelplot()`. Man findet sie im Paket `lattice` [32]. Während das erste Paket(`graphics`) beim Starten von `R` gleich mitgeladen wird, muss das Paket `lattice` zusätzlich geladen werden. Das Paket `arules` hängt vom Paket `lattice` ab und daher wird das zweite mitgeladen.

Nun wird eine Funktion definiert, die entweder `image()` oder `levelplot()` aufruft, um Assoziationsregeln als Gitter abzubilden. Man kann auch angeben, welches Interessantheitsmaß abgebildet werden soll. Standarteinstellung ist, dass `support` mittels der Funktion `levelplot()` visualisiert wird.

```
> gitter <- function(rules, kind = "levelplot", measure = "support") {
+     leftSideLabels <- labels(lhs(rules))$elements
+     rightSideLabels <- labels(rhs(rules))$elements
+     support <- quality(rules)[[measure]]
+     rulesAsDataFrame <- data.frame(Regelrumpf = leftSideLabels,
+         Regelkopf = rightSideLabels, Support = support)
+     rightSideLabels <- as.factor(rightSideLabels)
+     leftSideLabels <- as.factor(leftSideLabels)
+     m <- matrix(ncol = length(unique(as.integer(rightSideLabels))),
+         nrow = length(unique(as.integer(leftSideLabels))))
+     for (i in 1:nrow(rulesAsDataFrame)) m[as.integer(leftSideLabels)[i],
+         as.integer(rightSideLabels)[i]] <- support[i]
+     m[is.na(m)] <- 0
+     if (kind == "image") {
+         image(m, col = gray.colors(164), xlab = "Antecedent",
+             ylab = "Consequent", main = paste("Assotiation
+                rules with Image. Measure:", measure))
+     }
+     if (kind == "levelplot") {
+         levelplot(m, xlab = "Antecedent", ylab = "Consequent",
+             main = paste("Assotiation rules with Levelplot. Measure:",
+                 measure), aspect = "xy", cuts = 90)
+     }
+ }

> gitter(rules, measure = "confidence")
```

Das Ergebnisse sieht man in Abbildungen 4.2 und 4.3. Die Funktion erzeugt eine Matrix mit genauso viele Spalten, wie die Anzahl der unterschiedlichen Itemsets auf der rechten Seite

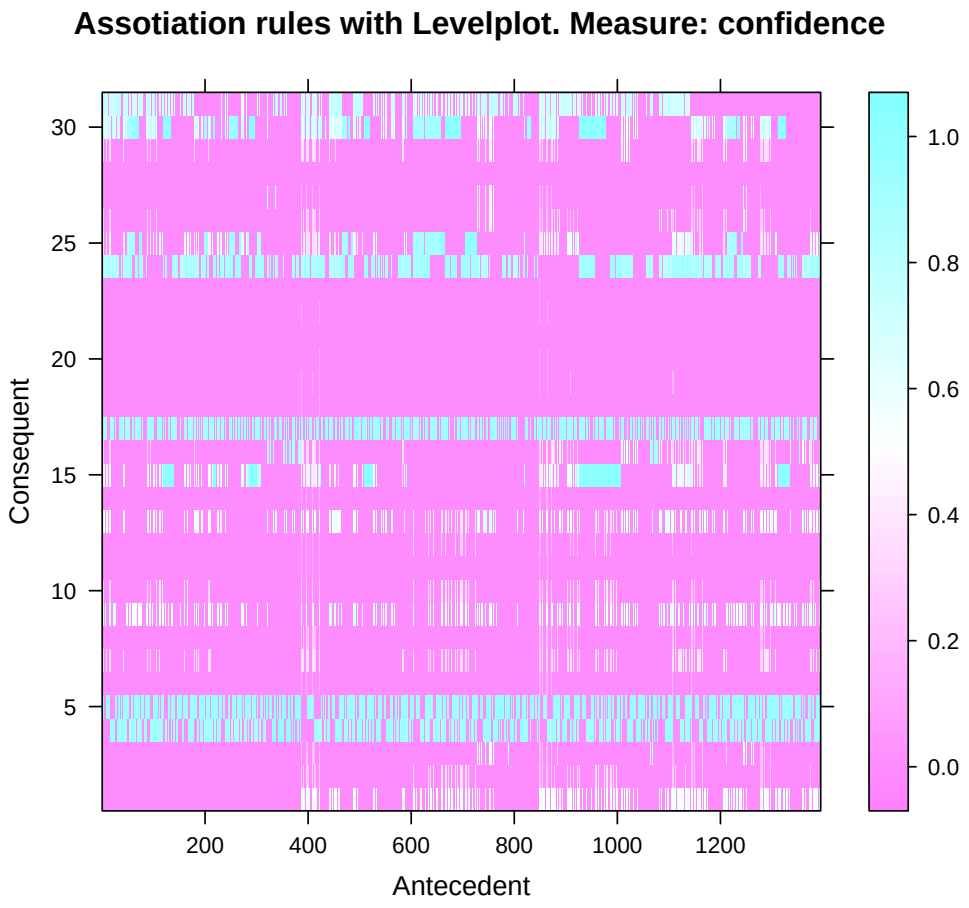


Abbildung 4.2: Assoziationsregeln mit Levelplot

```
> gitter(rules, kind = "image", measure = "lift")
```

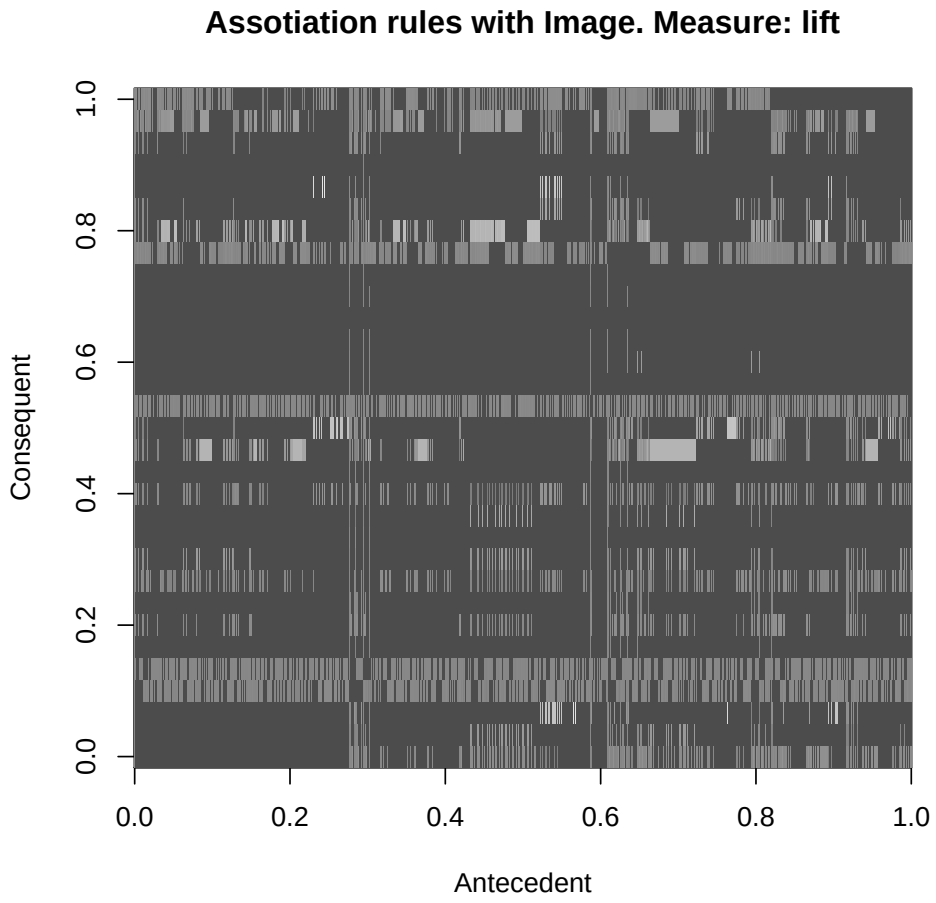


Abbildung 4.3: Assoziationsregeln mit Image

von `rules`. Die Anzahl der Zeilen der Matrix entspricht der Anzahl der unterschiedlichen Itemsets auf der linken Seite von `rules`. Die Funktion `as.integer()` macht das Wesentliche dabei. Sie wandelt die Namen der einzelnen Itemsets in Zahlen um. Zum Beispiel:

```
> rightSideLabels2 <- labels(rhs(rules))$elements
> rightSideLabels2 <- as.factor(rightSideLabels2)
> rightSideLabels2[1]

[1] {Occp=OthS}
31 Levels: {Age=MddA} {Age=Senr} {Age=Yong} ... {Wrkc=Prvt}

> as.integer(rightSideLabels2[1])

[1] 21

> levels(rightSideLabels2)

 [1] "{Age=MddA}"  "{Age=Senr}"  "{Age=Yong}"  "{CptG=None}"
 [5] "{CptL=None}" "{Edct=Bchl}" "{Edct=HSGr}" "{Edct=SmCl}"
 [9] "{HrPW=FllT}" "{HrPW=OvrT}" "{HrPW=PrtT}" "{Incm=Larg}"
[13] "{Incm=Smll}" "{MrtS=Dvrc}" "{MrtS=MrCS}" "{MrtS=NvrM}"
[17] "{NtvC=UntS}" "{Occp=AdmC}" "{Occp=CrfR}" "{Occp=ExcM}"
[21] "{Occp=OthS}" "{Occp=PrfS}" "{Occp=Sals}" "{Race=Whit}"
[25] "{Rltn=Hsbn}" "{Rltn=NtIF}" "{Rltn=OwnC}" "{Rltn=Unmr}"
[29] "{Sex=Feml}"  "{Sex=Male}"  "{Wrkc=Prvt}"
```

Man sieht, dass das erste Item von `rightSideLabels2` die Zahl 21 zurück gibt, wenn die Funktion `as.integer()` angewendet wird. Das bedeutet, dass die Werte für Support aller Regeln, die dieses Item (`{Occp=OthS}`) im Regelkopf haben, sich in der 21. Zeile der Matrix `m` befinden.

Man sieht der Unterschied zwischen `image()` und `levelplot()` in Abbildungen 4.2 und 4.3 . Die Abbildung mit `levelplot()` hat eine Maßeinteilung der Farben auf der linken Seite. Die Zahlen auf der X bzw. Y Achse geben die Nummer des jeweiligen Itemsets im Regelrumpf bzw. Regelkopf. Man sieht zum Beispiel, dass Itemsets 4 und 5 auf der Y Achse (also im Regelkopf) relativ viele Regeln bilden, die das Minimum an Support übersteigen. Während Itemsets von 18 bis 24 kaum Regeln bilden. Das erkennt man an fast durchgehende Rosafarbe in Spalten 18 bis 24.

4.3.4 Mosaik Plots

Mosaik Plots kann man mit `R` mittels der Funktionen `mosaicplot()`, `mosaic()` und `imosaic()` visualisieren. Diese Funktionen findet man in Paketen `graphics`, `vcd` beziehungsweise `iplots` (siehe dazu [25; 36; 31]). Die Funktion `imosaic()` Die Funktion `imosaic()` generiert eine interaktive Abbildung, die dem Anwender erlaubt, sie zu ändern oder zu rotieren. Leider können diese Effekte nicht auf Papier übertragen werden. Die Funktion `mosaic()` baut auf die Function `mosaicplot` auf und stellt eine verbesserte Version. Daher wird an dieser Stelle vorgeführt, wie man Assoziationsregeln mittels der Funktion `mosaic()` visualisieren kann.

Zuerst wird das Paket `vcd` geladen. Andere Pakete werden mitgeladen.

```
> library(vcd)
```

Die Funktion `mosaic()` benötigt als Argument eine Tabelle. Zu diesem Zweck werden zuerst einige Funktionen definiert, die aus einem Objekt der Klasse `rules` ein Objekt der Klasse `table` machen.

```
> getTable <- function(rule, data) {
+     regelrumpf <- unlist(LIST(lhs(rule), decode = FALSE))
+     regelkopf <- unlist(LIST(rhs(rule), decode = FALSE))
+     transactions <- data[, c(regelrumpf, regelkopf)]
+     ruleAsDataFrame <- as.data.frame(as(transactions,
+         "matrix"))
+     for (i in 1:ncol(ruleAsDataFrame)) {
+         ruleAsDataFrame[[i]] <- factor(ruleAsDataFrame[[i]],
+             levels = c(0, 1), labels = c("no", "yes"))
+     }
+     table(ruleAsDataFrame)
+ }
```

Die Funktion `getTable` macht zuerst aus einer Regel ein Objekt der Klasse `data.frame`. Zuerst werden die an der Regel beteiligten Transaktionen herausgeholt. Daraus wird eine Matrix erzeugt, deren Elemente entweder den Wert `0` oder `1` haben können. Aus dieser Matrix wird ein Objekt der Klassen `data.frame` erzeugt. Danach werden dessen Elementen umgewandelt. Für jeden Vektor im `rulesASdataFrame` wird `0` mit `no` und `1` mit `yes` ersetzt. Daraus wird ein Objekt der Klasse `table` gemacht, was zurückgegeben wird.

```
> rulesAStable <- function(rules, data) {
+     tables <- list()
+     for (i in 1:length(rules)) {
+         tables[[i]] <- getTable(rules[i], data)
```

```
+     }
+     tables
+ }
```

Die Funktion `rulesAStable()` benötigt als Argumente zwei Attribute. Das erste Argumen soll der Klasse `rules` angehören (es muss nicht ein einziges Regel sein).Das zweite soll der Datensatz sein, mit dem die Assoziationsregeln generiert wurden. Die Funktion gibt ein Objekt der Klasse `list` zurück, dessen Elemente Objekte der Klasse `table` sind. Das bedeutet, dass jedes Element der Funktio `mosaic` übergeben werden kann. Zuerst werden die einzelnen Regeln und der Datensatz an die Funktion `getTable()` übergeben. Das Objekt, das zurückgegeben wird, wird in einer Liste gespeichert.

Wie bereits erwähnt wurde (siehe Abschnit 3.3), können mit Mosaik Plot nur einzelne Assoziationsregeln visualisiert werden. Trotzdem werden jetzt zwei Regeln mit Hilfe der Funktion `sample()` gewählt. Damit soll gezeigt werden, dass es möglich ist, mehrere Regeln als Objekte der Klasse `table` in einer Liste gespeichert werden können. Diese können dann visualisiert werden.

```
> set.seed(9)
```

```
> twoRules <- sample(rules, 2)
```

Die Regel können mittels `inspect()` betrachtet werden.

```
> inspect(twoRules)
```

```
  lhs            rhs           support confidence     lift
1 {Wrkc=Prvt,
   HrPW=FllT} => {CptL=None} 0.4064330  0.9620063 1.009156
2 {Sex=Male}  => {HrPW=OvrT} 0.2111912  0.3159265 1.217299
```

Jetzt kann die Funktion definiert werden, die diese Regeln als Mosaik Plot visualisiert werden soll.

```
> mosaic.arules <- function(table) {
+     mosaic(table, highlighting = length(dim(table)),
+         main = "Mosaicplot with mosaic")
+ }
```

Diese Funktion bekommt als einziges Argument ein Objekt der Klasse `table`. Zuerst werden aber `twoRule Adult` an die Funktion `rulesAStable` übergeben.

```
> RulesList <- rulesAStable(twoRules, Adult)
> RulesList

[[1]]
, , CptL=None = no

         HrPW=FllT
Wrkc=Prvt    no   yes
      no    398   409
      yes   691   784

, , CptL=None = yes

         HrPW=FllT
Wrkc=Prvt    no   yes
      no   6596  7533
      yes 12580 19851


[[2]]
        HrPW=OvrT
Sex=Male    no   yes
     no  13831  2361
     yes 22335 10315
```

Das erste Element von `RulesList` wird an die Funktion `mosaic.arules()` übergeben.

```
> mosaic.arules(RulesList[[1]])
```

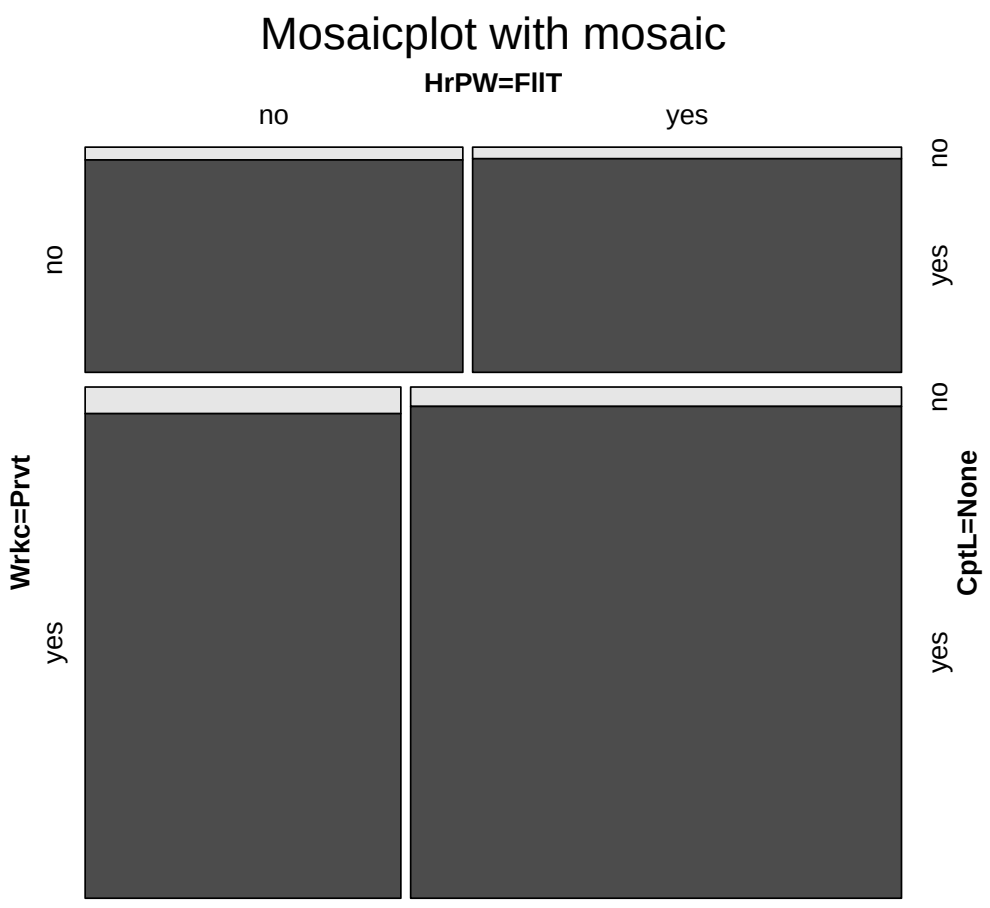


Abbildung 4.4: Assoziationsregeln mit mosaic

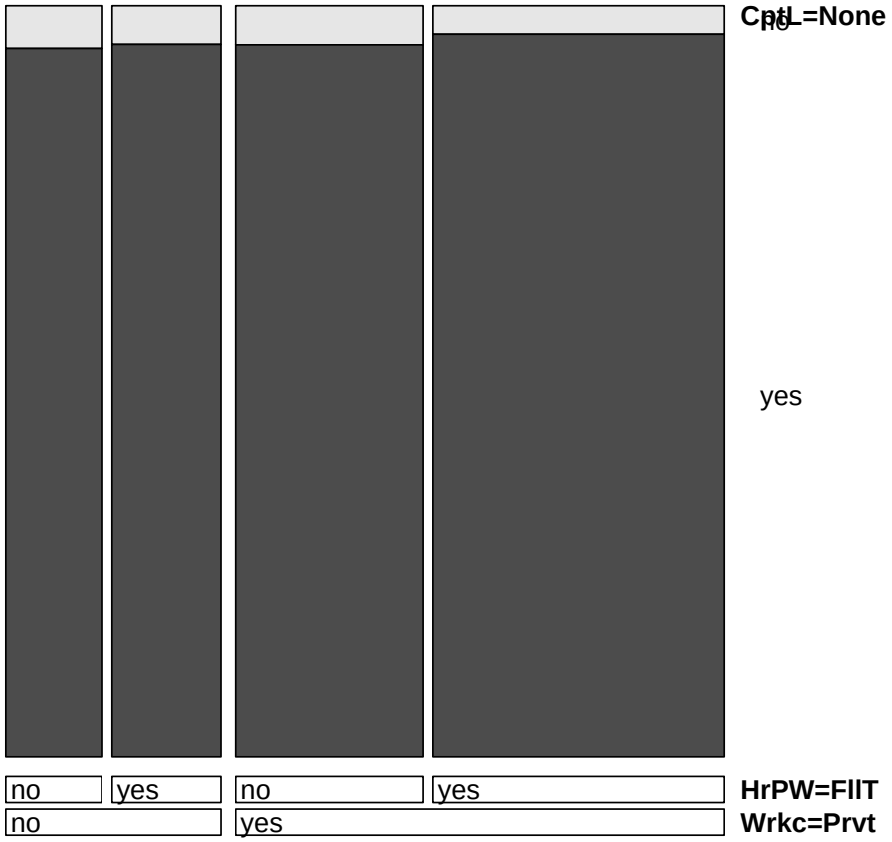


Abbildung 4.5: Zufällige Regel mit doubledecker

4.3.5 Doubledecker Plots

Wie aus dem Abschnitt 3.4 bekannt ist, stellen Doubledecker Plots Mosaic Plots dar, mit dem Unterschied, dass die Fläche einer Abbildung bis auf das Hervorheben immer vertikal geteilt wird. In `R` existiert bereits eine Funktion namens `doubledecker()`. Man findet sie im Paket `vcd`. Diese Funktion benötigt auch ein Objekt der Klasse `table`. Daher sind die im vorherigen Abschnitt definierten Funktion auch für Doubledecker Plots anwendbar. Abbildung 4.5 zeigt dieselbe Regel von vorhin.

```
> doubledecker(RulesList[[1]], margin = c(2, 6,
+              length(dim(RulesList[[1]])) +  2))
```

4.3.6 Graphen

Graphen kann man in R mit dem Paket Rgraphviz [9] darstellen. Dieses Paket findet man auf der Homepage von *BIOCONDUCTOR* (http://www.bioconductor.org). Das ist ein Open Source Projekt für Analyse von Biodaten [8].

Mit gerichteten Graphen lassen sich sowohl häufige Itemsets als auch Assoziationsregeln darstellen. Das Paket wie üblich mit `library()` geladen.

```
> library(Rgraphviz)
```

Es werden wieder einige Funktionen definiert, mit denen häufige Itemsets abgebildet werden können. Davor sollen aber diese generiert werden. Das geschieht mit Hilfe der Funktion `eclat()`.

```
> itemsets <- eclat(Adult, parameter = list(support = 0.55))

parameter specification:
 tidLists support minlen maxlen            target   ext
    FALSE    0.55      1      5 frequent itemsets FALSE

algorithmic control:
 sparse sort verbose
      7   -2    TRUE

eclat - find frequent item sets with the eclat algorithm
version 2.6 (2004.08.16)         (c) 2002-2004   Christian Borgelt
create itemset ...
set transactions ...[115 item(s), 48842 transaction(s)] done [0.16s].
sorting and recoding items ... [7 item(s)] done [0.02s].
creating bit matrix ... [7 row(s), 48842 column(s)] done [0.01s].
writing  ... [34 set(s)] done [0.01s].
Creating S4 object  ... done [0.00s].

> itemsets

set of 34 itemsets
```

Zuerst werden die verwendeten Parameter ausgegeben. Es wurden 34 häufige Itemsets generiert.

```
> removeBracket <- function(itemsetsLabels) {
+     itemsetsLabels <- gsub("\\{", "", itemsetsLabels)
+     itemsetsLabels <- gsub("\\}", "", itemsetsLabels)
+     itemsetsLabels
+ }
```

Die Funktion `removeBracket` entfernt die geschweiften Klammern aus dem Vektor mit den Namen der Itemsets und gibt ihn wieder. Damit soll erreicht werden, dass später die Funktion `grep()` problemlos angewendet wird.

```
> createGraph <- function(itemsets) {
+     itemsetsLabels <- removeBracket(labels(itemsets))
+     graph <- new("graphNEL", nodes = itemsetsLabels,
+         edgemode = "directed")
+     graph <- addEdges.items(graph, itemsets)
+     nodes(graph)[1:length(nodes(graph))] <- paste(1:length(nodes(graph)))
+     graph
+ }
```

Diese Funktion macht aus den Itemsets einen Graph. Zuerst werden die geschweiften Klammmern entfernt. Dann erzeugt sie den Graph mit `new()`. Danach wird die Funktion `addEdges.items()` aufgerufen. Sie fügt Kanten zwischen den Knoten. Da die Namen der einzelnen Knoten zu lang sein können, werden sie mit Nummern ersetzt. Die Funktion gibt einen Graph zurück.

```
> addEdges.items <- function(graph, itemsets) {
+     itemsLabels <- nodes(graph)
+     for (i in 1:length(nodes(graph))) {
+         graph <- addEdge(nodes(graph)[i], nodes(graph)[just(i,
+             grep(nodes(graph)[i], nodes(graph)), itemsets)],
+             graph, 1)
+     }
+     graph <- removeEdge(nodes(graph), nodes(graph), graph)
+     graph
+ }
```

Die Funktion `addEdges.items()` fügt Kanten zwischen Itemsets der Länge i und $i + 1$. Dafür sorgt die Funktion `just()`.

```
> just <- function(cell, numbers, itemsets) {
+     if (length(numbers) == 1)
+         numbers
```

```
+     else {
+         numbers <- numbers[size(itemsets[cell])
+            >= size(itemsets[numbers]) - 1]
+     }
+ }
```

Diese Funktion bekommt eine ganze Zahl (`cell`), einen Vektor mit ganzen Zahlen (`numbers`) und ein Set von Itemsets (`itemsets`). Die `just()` Funktion gibt einen Vektor zurück. Sie stellt sicher, dass nur diejenige Elementen von `numbers` zurückgegeben werden, die folgende Bedingung erfüllen:

```
size(itemsets[cell])>=size(itemsets[numbers])-1
```

Damit werden nur die häufige Itemsets verbunden, die gemessen an ihrer Länge, sich um nur 1 unterscheiden. Das bedeutet, dass zum Beispiel zwischen einem häufigen Itemset der Länge 2 und einem der Länge 4 keine Kante enstehen darf.

Es wird eine Funktion definiert, die die Itemsets als Graph visualisiert.

```
> graph_viz_freq <- function(itemsets) {
+     graph <- createGraph(itemsets)
+     nA <- list()
+     shape <- rep("circle", length(itemsets))
+     names(shape) <- nodes(graph)
+     nA$shape <- shape
+     fivecolors <- colors()[c(589, 530, 417, 568, 554)]
+     fillcolor <- fivecolors[size(itemsets)]
+     names(fillcolor) <- nodes(graph)
+     nA$fillcolor <- fillcolor
+     fontsize <- rep("18", length(itemsets))
+     names(fontsize) <- nodes(graph)
+     nA$fillcolor <- fillcolor
+     width <- paste(round(quality(itemsets)$support, 2) *
+         2)
+     names(width) <- nodes(graph)
+     nA$width <- width
+     height <- paste(round((quality(itemsets)$support) *
+         2/3, 2) * 2)
+     names(height) <- nodes(graph)
+     nA$height <- height
+     plot(graph, nodeAttrs = nA, main = "Frequent itemsets")
+ }
```

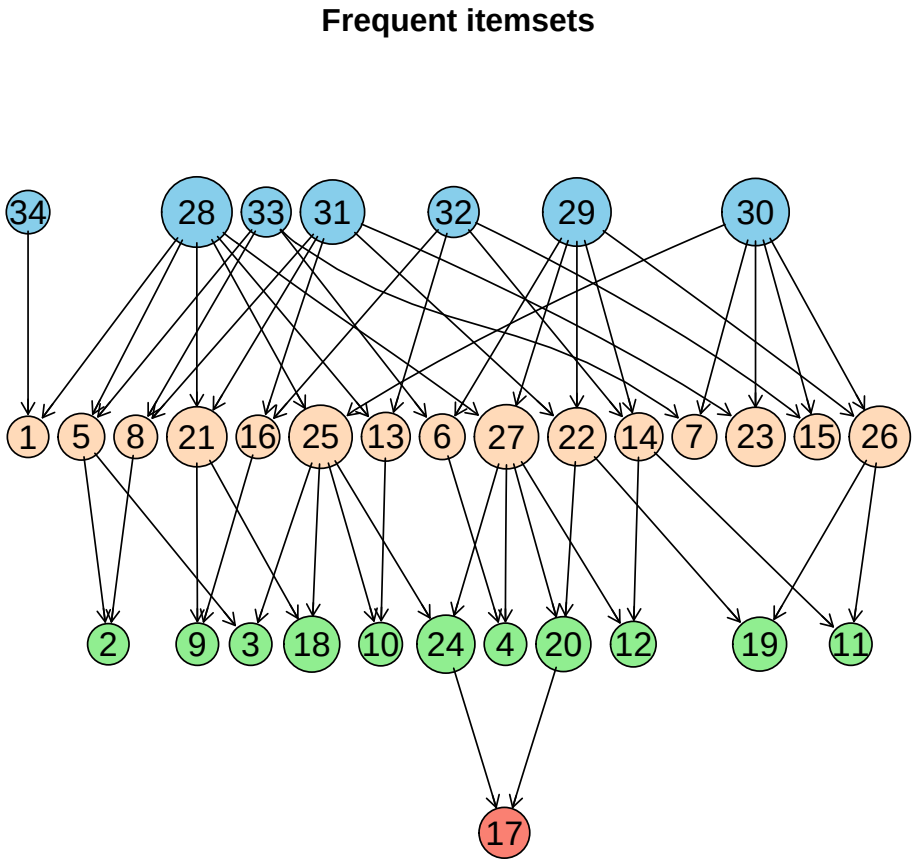


Abbildung 4.6: Häufige Itemsets mit Rgraphviz

Die Funktion `graph_viz_freq` wird mit dem Objekt `itemsets` aufgerufen. Dabei werden auch Farben verwendet, um die verschiedene Länge der Itemsets zum Ausdruck zu bringen. Die Größe der einzelnen Knoten gibt den Support an. Das bedeutet, dass größere Knoten Itemsets mit einem höheren Wert des Supports darstellen.

```
> graph_viz_freq(itemsets)
```

Man sieht das Ergebnis des Aufrufes in Abbildung 4.6. Es fällt auf, dass keine Knoten einer unteren Ebene größer ist als seine Vorgänger.

Jetzt soll gezeigt werden, wie Assoziationsregeln mit Graphen visualisiert werden können. Dazu werden `subrules` verwendet.

```
> subrules

set of 36 rules

> graph_viz <- function(rules, measure = "support") {
+     leftSide <- labels(lhs(rules))$elements
+     rightSide <- labels(rhs(rules))$elements
+     support <- quality(rules)[[measure]]
+     nodes1 <- unique(rightSide)
+     fromto <- cbind(leftSide, rightSide)
+     graph <- ftM2graphNEL(fromto, edgemode = "directed")
+     eAttrs <- list()
+     edgeLabels <- paste(round(support, 2))
+     edgeLabels <- edgeLabels[setdiff(seq(along = edgeLabels),
+         removedEdges(graph))]
+     names(edgeLabels) <- edgeNames(graph)
+     eAttrs$label <- edgeLabels
+     labelfontsize <- rep("1", length(edgeNames(graph)))
+     names(labelfontsize) <- edgeNames(graph)
+     eAttrs$labelfontsize <- labelfontsize
+     farben <- colors()[unique(round(runif(length(edgeNames(graph)),
+         1, length(colors()))))]
+     farben2 <- colors()[unique(round(runif(length(edgeNames(graph)),
+         1, length(colors()))))]
+     farben3 <- colors()[unique(round(runif(length(edgeNames(graph)),
+         1, length(colors()))))]
+     farb <- c(farben, farben2, farben3)
+     farb <- unique(farb)
+     color <- farb[1:length(edgeNames(graph))]
+     names(color) <- edgeNames(graph)
+     eAttrs$color <- color
+     fontcolor <- farb[1:length(edgeNames(graph))]
+     names(fontcolor) <- edgeNames(graph)
+     eAttrs$fontcolor <- fontcolor
+     twocolors <- c("lightgreen", "skyblue")
+     nodeType <- 1 + (nodes(graph) %in% nodes1)
+     nA = makeNodeAttrs(graph, fillcolor = twocolors[nodeType],
+         label = 1:length(nodes(graph)), shape = "circle")
+     sg1 = subGraph(nodes1, graph)
```

```
+     sgL = list(list(graph = sg1, cluster = FALSE,
+                     attrs = c(rank = "sink")))
+     att = list(graph = list(rankdir = "LR", rank = ""))
+     plot(graph, nodeAttrs = nA, attrs = att, edgeAttrs = eAttrs,
+         subGList = sgL, "dot")
+ }
```

Die Funktion `graph_viz` holt sich zuerst die Namen der Elementen im Regelrumpf und Regelkopf aus den Assoziationsregeln, mit denen sie aufgerufen wurde (dazu auch ein Interessantheitsmaß - Standarteinstellung ist Support). Aus den Namen macht sie dann eine Matrix (`fromto`), die sie an die Funktion `ftM2graphNEL()` übergibt. Diese erzeugt einen Graphen mit Kanten. Die Matrix `fromto` beinhaltet die Information, welche Knoten mit welchen Knoten verbunden werden sollen. Dann werden verschiedene Eigenschaften der Knoten und Kanten eingestellt (zum Beispiel Form, Farbe, Schrift). Für Knoten gibt es eine eigene Funktion, die das macht (`makeNodeAttrs()`). Für die Kanten wird eine Liste definiert (`eAttrs`). Dieser Liste werden dann benannte Folgen als einzelne Elementen zugewiesen. Es wird auch ein Subgraphen definiert (`sg1`). Das hilft beim Visualisieren, indem die Elementen im Regelrumpf und Regelkopf auf zwei verschiedenen Seiten stehen. Die Liste `att` sorgt für die Ausrichtung der verschiedenen Knoten. Am Ende werden alle Listen an die Funktion `plot()` übergeben. Das Ergebnis sieht man in Abbildung 4.7.

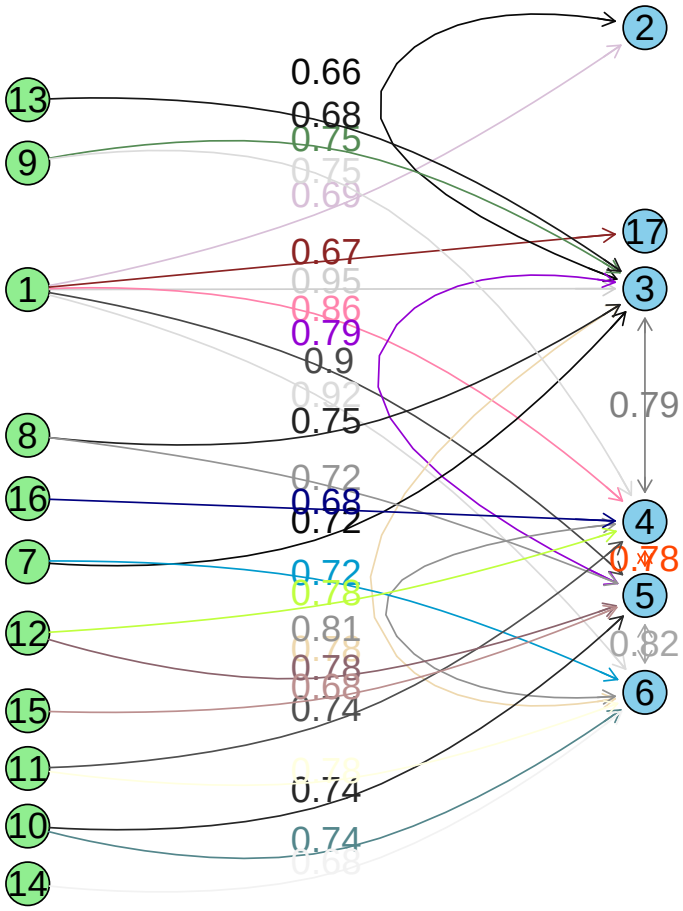


Abbildung 4.7: Assoziationsregeln mit Rgraphviz

```
> graph_viz(subrules)
```

Kapitel 5

Zusammenfassung und Ausblick

In dieser Arbeit wurden Assoziationsregeln und deren Visualisierung diskutiert. Der Schwerpunkt lag dabei auf Visualisierung mit Hilfe der Programmiersprache `R`.

Es wurden verschiedene Merkmale einer Assoziationsregel vorgestellt. Einige von diesen Interessantheitsmaße wurde dann in späteren Abschnbitten verwendet, um die Anzahl der generierten Regeln zu reduzieren. Der `Apriori` Algorithmus wurde auch ausführlich beschrieben.

Dann wurden diverse Visualisierungstechnicken vorgestellt, mit denen sich Assoziationsregeln darstellen lassen. Unter anderem wurden dreidimensionale Matrizen, Gitter, Mosaik Plots und gerichte Graphen präsentiert.

In Kapitel Implementierung wurde versucht, einige dieser Techniken mit Hilfe der Programmiersprache `R` zu implenetieren. Diese Visualisierungstechniken sind:

1. Dreidimensionalen Matrizen mit Hilfe des Pakets `scatterplot3d`
2. Gitter mit Hilfe der Pakete `graphics` mit der Funktion `image()` sowie `lattice` mit der Funktion `levelplot()`
3. Mosaik und Doubledecker Plots mit Hilfe des Pakets `vcd`
4. Gerichtete Graphen mit Hilfen des Pakets `Rgraphviz`

Es hat sich gezeigt, dass sich die Programmiersprache `R`, die zugleich auch Datenbearbeitungsprogramm ist, sehr gut geeignet ist, solche Probleme zu lösen.

Im Hinblick auf die zuküftigen Überlegungen könnten die in dieser Arbeit vorgestellten Implementierungen zu einem `R` Paket zusammengefasst werden. Dabei können die in dieser Arbeit etwickelten Funktionen weiter verbessert werden.

Literaturverzeichnis

[1] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, 1993.

[2] Wolfgang Behme and Marko Multhaupt. Text mining im strategischen controlling. *Handbuch der maschinellen Datenverarbeitung*, 207:103–114, 1999.

[3] Michael J. Berry and Gordon Linoff. *Data Mining Techniques: For Marketing, Sales, and Customer Support*. John Wiley & Sons, Inc., New York, NY, USA, 1997.

[4] Toni Bollinger. Assoziationsregeln - analyse eines data mining verfahrens. *Informatik-Spektrum*, 19:257–261, 1996.

[5] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman. *Readings in Infomation Visualization Using Vision to Think*. Morgan Kaufmann Publishers, 1999.

[6] Aaron Ceglar, John Roddick, Paul Calder, and Chris Rainsford. Visualising hierarchical associations. *Knowledge and Information Systems*, 8(3):257–275, 2005.

[7] Guerdal Ertek and Ayhan Demiriz. A framework for visualizing association mining results. In *Lecture Notes in Computer Science*, volume 4623/2006, pages 593–602. Springer Berlin / Heidelberg, 2006.

[8] Robert C Gentleman, Vincent J. Carey, Douglas M. Bates, et al. Bioconductor: Open software development for computational biology and bioinformatics. *Genome Biology*, 5:R80, 2004.

[9] Jeff Gentry, Li Long, Robert Gentleman, and Seth Falcon. *Rgraphviz: Provides plotting capabilities for R graph objects*, 2007. R package version 1.14.0.

[10] Michael Hahsler, Bettina Grün, and Kurt Hornik. arules – A computational environment for mining association rules and frequent item sets. *Journal of Statistical Software*, 14(15):1–25, October 2005.

[11] Jiawei Han and Micheline Kamber. *Data Mining: Conepts and Techniques*. Morgan Kaufmann Publishers, 2001.

[12] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. Algorithms for association rule mining — a general survey and comparison. *SIGKDD Explorations*, 2:58–64, 2000.

[13] Hajo Hippner, Ulrich Küsters, and Matthias Meyer. *Handbuch Data Mining im Marketing: Knowledge Discovery in Marketing Databases*. Vieweg, 2001.

[14] Heike Hofmann. Generalized odds ratios for visual modeling. *Journal of Computional and Graphical Statistis*, 10/4:628–640, 2001.

[15] Heike Hofmann, Arno P. J. M. Siebes, and Adalbert F. X. Wilhelm. Visualizing association rules with interactive mosaic plots. In *KDD '00: Proceedings of the sixth ACM SIGKDD International conference on Knowledge discovery and data mining*, pages 227–235, 2000.

[16] Heike Hofmann and Adalbert Wilhelm. Visual comparison of association rules. *Computational Statistics*, 16(3):399–415, 2001.

[17] Martin Hubert. Pfadanalysen führen zum kunden. *Absatzwirtschaft*, 11:108, 1999.

[18] Bayardo R. J. and R. Agrawal. Mining the most interesting rules. In *Proc of. the Fifth ACM-SIGMOD International Conference on Knowledge Discovery and Data Mining.*, 1999.

[19] Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, Hannu Toivonen, and A. Inkeri Verkamo. Finding interesting rules from large sets of discovered association rules. In *CIKM '94: Proceedings of the third international conference on Information and knowledge management*, 1994.

[20] Uwe Ligges. *Programmieren mit R.* Springer-Verlag, Heidelberg, 2005.

[21] Uwe Ligges and Martin Mächler. Scatterplot3d - an r package for visualizing multivariate data. *Journal of Statistical Software*, 8(11):1–20, 2003.

[22] Bing Liu, Kaidi Zhao, Jeffrey Benkler, and Weimin Xiao. Rule interestingness analysis using olap operations. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 297 – 306. ACM Press, 2006.

[23] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Efficient algorithms for discovering association rules. In *AAAI Workshop on Knowledge Discovery in Databases (KDD-94)*, 1994.

[24] Wolfgang Martin. *Data Warehousing Data Mining - OLAP.* International Thomson Publishing, 1998.

[25] David Meyer, Achim Zeileis, and Kurt Hornik. Framework: Visualizing multi-way contingency tables with vcd. *Journal of Statistical Software*, 17(3):1–48, 2006.

[26] Kian-Huat Ong, Kok-Leong Ong, Wee-Keong Ng, and Ee-Peng Lim. Crystalclear: Active visualization of association rules. *IEEE*, 15:15–25, 2002.

[27] Ralf Otte, Viktor Otte, and Volker Kaiser. *Data Mining für die industrielle Praxis*. Carl Hanser Verlag, 2004.

[28] Helge Petersohn. *Data Mining: Verfahren, Prozesse, Anwendungsarchitektur*. Oldenbourg Verlag, 2005.

[29] Gregory Piatetsky-Shapiro and William J. Frawley. *Knowledge Discovery in Databases*. AAAI Press / The MIT Press, 1991.

[30] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2006. ISBN 3-900051-07-0.

[31] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2007. ISBN 3-900051-07-0.

[32] Deepayan Sarkar. *lattice: Lattice Graphics*, 2007. R package version 0.15-8.

[33] Heidrum Schumann and Wolfgang Müller. *Visualisierung-Grundlagen und allgemeine Methoden*. Springer, 2000.

[34] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Pearson / Addison-Wesley, 2006.

[35] Kesaraporn Techapichetvanich and Amitava Datta. Visar: A new technique for visualizing mined association rules. In *Lecture Notes in Artificial Intelligence*, volume 3584/2005, pages 88–95. Springer Berlin / Heidelberg, 2005.

[36] Simon Urbanek and Tobias Wichtrey. *iplots: iPlots - interactive graphics for R*, 2007. R package version 1.0-7.