The Decision Exploration Lab

Supporting the business analyst in understanding automated decisions

Bertjan Broeksema

The work in this thesis has been carried out under the auspices of the IBM France Center for Advanced Studies (CAS), the Scientific Visualization and Computer Graphics (SVCG) research group, and the Laboratoire Bordelais de Recherche en Informatique (LaBRI). SVCG is affiliated with the university of Groningen. LaBRI is affiliated with the university of Bordeaux. The work was financed by IBM France and through two IBM Ph.D. Fellowship awards.

ISBN: 978-90-367-6786-6
ISBN: 978-90-367-6785-9 (Electronic version)

Document prepared with LaTeX and typeset by pdfTeX
Cover deisgn:
Printed by:

rijksuniversiteit groningen

The Decision Exploration Lab

Supporting the business analyst in understanding automated decisions

Proefschrift

ter verkrijging van de graad van doctor aan de
Rijksuniversiteit Groningen
op gezag van de
rector magnificus, prof. dr. E. Sterken
en volgens het besluit van het College voor Promoties.

De openbare verdediging zal plaatsvinden op

maandag 3 maart 2014 om 11:00 uur

door

Albertus Hendrik Johan Broeksema

geboren op 10 april 1983
te Enschede

Promotores:
Prof. dr. A. C. Telea
Prof. dr. G. Melançon

Copromotor:
Dr. T. Baudel

Beoordelingscommissie:
Prof. dr. J.-D. Fekete
Prof. dr. A. Kerren
Prof. dr. M. J. McGuffin

Thesis summary

A Decision Management System (DMS) provides means to model and automate enterprise decisions and they are applied in a wide range of industries, among which health care, commerce, insurance, finance and transportation. These systems make millions of decisions each day without direct human supervision, impacting the life of millions of people and impacting economies at a large scale. The multiplicative effect of decision automation provides the opportunity to fine-tune the decision system. By analyzing its global and emerging properties rather than focusing on the details of each decision, the system as a whole can be better adapted to the reality it models.

Like expert systems, DMSs provide a clear separation of decision logic, information related to individual decisions and decision execution. These data spaces contain a wealth of information related to the structure and functioning of a DMS. In this thesis various ways are explored to visualize and analyze this data in order to help a business user to gain a deeper understanding of automated decisions.

To address the problem of understanding the global and emerging properties of automated decision making systems, we combine interactive analysis of the decision data with analysis of the decision logic. We present a visual analytics system, the Decision Exploration Lab (DEL), which provides a verbal analysis mode and a visual decision exploration mode. In verbal mode the user can make selections on past decisions using controlled natural language. In visual decision exploration mode, the decision data is analyzed using Multiple Correspondence Analysis (MCA). The analysis results are visualized using interactive techniques to show the important structure of the decision data to the user. Correlated concepts can be clustered at a level of granularity that suits the needs of the business analyst. Clustered concepts can next be linked to the rules of the decision logic that are relevant for the subset of decisions which match these concepts. We evaluated our approach with two use case scenarios from the car insurance industry.

Apart from the above, we propose a number of technical contributions, enhancements and extensions to information visualization methods, for multivariate categorical data. Firstly, we present a generic algorithm to generate all well-known treemap layouts as well as other rectangular space-filling layouts. Secondly, we present explanatory and interactive visualization techniques to support interpretation and usage of MCA. Thirdly, we present labeling and scale adjustment techniques in order to improve the usability of 2D-plots.

Samenvatting

Een Decision Management System (DMS) is een systeem om dagelijkse enterprise besluiten te modeleren en te automatiseren. Dit soort systemen worden toegepast in verschillende industrieën zoals de gezondheidszorg, de handel, het verzekeringswezen, het bankwezen en de transport sector. Deze systemen maken elke dag milioenen besluiten met betrekking tot de bedrijfsvoering, en hebben zodoende invloed op zowel de levens van milioenen individuen alswel op economiën. Het vermenigvuldigende effect van het automatiseren van dagelijkse besluiten biedt de mogelijkheid tot een preciese afstemming van DMSs. Door het analyzeren van de globale eigenschappen die zich voordoen, in plaats van de aandacht te richten op individuele besluiten, kan het systeem als geheel beter aangepast worden aan de realiteit die het modeleert.

Net als expert systemen, hebben DMSs een duidelijke scheiding van de besluit logica, informatie die betrekking heeft op individuele besluiten en de uitvoer van besluiten. Elk van deze drie data domeinen bevat een grote hoeveelheid informatie die aspecten van de structuur en het functioneren van een DMS beschrijven. In dit proefschrift onderzoeken we verschillende manieren om deze informatie te visualizeren en te analyzeren om een business analist een dieper inzicht te geven in het functioneren van geautomatiseerde besluiten.

Voor het verkrijgen van dit inzicht, combineren we interactieve analyze van besluit data met analyze van de besluit logica. We presenteren een visual analytics systeem, het Decision Exploration Lab (DEL), dat een verbale analyze modus and een visuele besluit analyze modus biedt. In de verbale modus kan de gebruiker selecties maken van historische besluiten door gebruik te maken van gecontrolleerde natuurlijke taal. In de visuele besluit exploratie modus, wordt de besluit data geanalyseerd met Multiple Correspondence Analysis (MCA). De resultaten van de analyze worden gevisualiseerd, gebruik makend van interactieve technieken, om de belangrijke structuren van de besluit data inzichtelijk te maken voor de gebruiker. Gecorreleerde concepten kunnen geclusterd worden op een niveau dat geschikt is voor de vragen die de business analist probeert te beantwoorden. Geclusterde concepten kunnen vervolgens gelinkt worden aan regels van de besluit logica die relevant zijn voor de besluiten die overeenkomen met de concepten in het geselecteerde cluster. We hebben onze aanpak geëvalueerd met behulp van twee scenario's uit de autoverzekering branche.

Naast de boven genoemde contributie, presenteren we verschillende technische contributies, verbeteringen en uitbreidingen van informatie visualisatie technieken voor multivariate categorische data. Allereerst presenteren we een generiek algoritme om alle bekende treemap en andere ruimte vullende, rechthoekige, layouts te genereren. Ten tweede, we presenteren verklarende en interactieve visualisatie technieken om het gebruik en de interpretatie van MCA te ondersteunen. Ten derde, we presenteren labeling en schaal aanpassings technieken om de bruikbaarheid van 2D-plots te verbeteren.

To my amazing wife Agnes
and to my beautiful daugther Venne

Where is the Life we have lost in living?
Where is the wisdom we have lost in knowledge?
Where is the knowledge we have lost in information?

— T.S. Elliot, The Rock, 1934 —

Contents

List of Figures

List of Tables

Listings

Acknowledgments

Even though the journey towards this thesis was lonely at times, I have not been walking on my own. There have been many people who have made this journey lighter, more interesting and more exciting. Moreover, these people have kept me pushing forward, made me push my boundaries and encouraged me not to give up.

My supervisors, Alex Telea from the university of Groningen and Thomas Baudel from IBM France, have been instrumental for the development of my academic and professional capabilities the last years. Alex has been supervising me first during my final master internship and the last years during my Ph. D. Working with him was one of the best things that could have possibly happened in the process of pursuing a Ph. D. He has teached and explained to me a lot of things which would have been complete magic for me years ago. Thomas has been supervising me from the business side of things. His knowledge on decision automation, information visualization and his reluctance to accept a world that is run by automated intelligent(?) systems have truly shaped my thinking. What I appreciated even more was his constant gentle, though persisting pressure to reach further, put the bar higher. You have made me push my boundaries to a much larger extent than I would have done on my own. I also truly enjoyed the many, many discussions I have had with both of you on research, life, the universe and everything. I also owe thanks to Guy Melançon, who kindly agreed to arrange the French side of my research setup. We had the pleasure to meet and exchange ideas, although not as often as I would have liked.

Research is always performed in a community. Therefore I would like to thank all reviewers who have been reading and commenting on my work, mostly anonymous. In particular I would like to thank the three reviewers of this thesis, prof. dr. J.-D. Fekete, prof. dr A. Kerren and prof. dr. M. J. McGuffing. Their remarks and questions led to a large number of improvements of this thesis.

There have been many colleagues at IBM that have provided me with invaluable input on general context and background of decision management, technical issues, customer needs and even code. In particular I would like to thank Patrick Albert, Pierre Chardin, Paolo Crisafulli, Christian Deutsch, Adil El Ghali, Pierre Feillet, Laurent Grateau, Alain Neyroud, Pierre-andre Paumelle, Jean Pommier, Christian de Sainte Marie and Nicolas Sauterey. Additionally, there always has been a coming and going of Ph. D. students and interns who made life at IBM pleasant by means of fun conversations, picnics and games of pool. So, thanks, Amina, Nicolas, Penelope, Olva, Samir, Lexi, Pierre André, Alioune, Abdessamad, Valerio, it was really fun to have you around.

Many friends and family have made our stay in Paris possible, easier and joyful, too many to name them all. All those of you who have visited us, called us, send us postal cards, emails or other signs of life, thank you. Two of you I want to thank in particular, as you basically became part of our furniture: Luuk and Inge. Dad, Wim, Henk, you all have been a great help to get

us settled in Paris. I would also like to thank the members of the Eglise Réformée Néerlandais à Paris, who have welcomed me and Agnes with open arms from the very beginning. In particular, thanks to all members of "Kring Parijs", I really enjoyed the monthly meetings and yearly barbecues.

I would like to thank my parents in law, Piet and Marleen. You both have been of great support over the years, even though I took your daughter to Paris. In particular I would like to thank Piet. You have been a great inspiration for me and encouraged me to keep expanding my knowledge.

Fifteen years ago I asked my parents why I would ever need lo learn German or French when I knew that I wanted to pursue Computing Science. Now I know... and now I wonder why you never used the rod on me at the time (Proverbs 22:15). It was because whatever path I would chose, you would never and have not given up your loving support. Dad, mom, I dearly love you! Finally, I'd like to dedicate this thesis to my wife Agnes. You would probably not believe me when I say that this thesis would not have happened without you, but really, it wouldn't. Life is a wonderful ride with you on my side and provokes talents which I never expected to have. Your love, your support, your patience and even you correcting me at times give brilliance to every moment we are together, and make me long for home every moment we are apart. I dearly hope to enjoy your companionship for many years to come.

Bertjan Broeksema
Paris September 29, 2013

Chapter 1

Introduction

It's a mystery to me - the game commences
For the usual fee - plus expenses

Private Investigations, MARK KNOPFLER

The use of Information Technology (IT) to support enterprises in becoming more effective, efficient and customer oriented has increased significantly the last decades. In the sixties and seventies, centralized systems were developed and found their way in enterprises in the form of mainframes. These systems enabled multiple users to send their enterprise critical tasks to the mainframe. Additionally, scale enlargement could take place due to automated mass processing of large amounts of data. In the eighties, personal computers became more and more common. Business users were now supported individually and could structure and improve their business activities by using spreadsheets, text documents and databases. Networking capabilities took a large step forward with the speed increases of network traffic in the nineties. Sharing work across business users by means of email and file sharing became ubiquitous. It made large scale business-to-business trading possible and opened new ways for interaction between businesses and customers.

One of the main challenges that businesses faces these days is the integration of business strategy and goals with their IT-systems. This has led to the emergence of software systems that operates on the intersection between business and technology. Various systems have been developed to model and automate business processes, handle business events and automate enterprise decisions. Business Process Modeling software enables business users to gain understanding of complex business processes. It allows them to reason about these processes, automate the processes and better align them with business goals and client needs. Complex Event Processing combines various sources of information streams in order to analyze and structure business events and the business's reaction to these events. A Decision Management System (DMS) allows enterprises to model and automate their enterprise decisions.

With these developments, it also became clear that for agile and adaptive business automation, direct involvement of domain experts is crucial. As opposed to IT-departments which are merely concerned with the technical aspects of IT, domain experts have a clear understanding of their sector and the enterprise's policies. Directly involving experts from sectors such as finance, legal or marketing is of importance for a revenue generating, legal and targeted automation of business. To decrease the implementation time of new processes and decisions and the adaptation of existing ones, business automation software systems have become tools that are increasingly put into the hands of such users. Effectively, one could say that business users became programmers, though the underlying hardware of their 'programs' is not a computer but the enterprise itself. Continuing this metaphor, business users also need to 'debug' and analyze their automated business processes. However, unlike in software engineering, the questions business users deal with are open ended. Markets, regulations and business policies change all the time and business needs are often not precisely defined. These considerations lead to the need for new exploratory tooling that helps business users understand their processes and decisions, find clues for improvements and new business opportunities.

In this thesis we particularly focus on DMS. DMSs are applied to a wide range of domains such as banking, retail and insurance. They help enterprises to make complex decisions in a consistent and automated way, manage risk, reduce time-to-market and increase customer satisfaction. One has to realize that many of these decisions involve actual money, e.g. loan applications, insurance quotes, cross-selling offers, and are made in large volumes. Clearly, these systems can have a huge financial impact, both on the enterprise and on society.

1.1 Automating business decisions

Enterprises make decisions all the time, both major (e.g. in which country to do business? Do we start the development of a new product?) as well as minor (e.g. do we offer a customer a discount? Is this transaction fraudulent?). Some of the decisions will be made only once or a couple of times while others might be taken thousands of times a day. Each of the business critical decisions that are taken stem from a business policy dictated by the responsible management.

Suppose an enterprise decides that each customer with a portfolio value above one million should be contacted on a monthly basis. Although this is a sufficient clear policy in itself, it rises various questions. What to do with the decision? How can management and the responsible persons know that the policy exists? How can management know that this policy is enforced in a consistent manner? It also brings up deeper questions such as: Is it the right decision? And, if this decision changes, what is the impact of this change?

These questions pertain to the remark of James Taylor, stating that business decisions should be treated as corporate assets [4, p19-20]. He continues to describe that corporate assets are *strategic*, *managed*, *visible*, *reusable* and *improving*. An asset is strategic when it is considered how it can be used to reduce costs, increase revenue and expand business. An asset is managed, meaning that it is reviewed and improved on a regular base to be kept in good order. Assets must be visible to management in order to be used correctly. They are normally not left idle but reused and leveraged as much as possible. Finally, assets are improved constantly, thus in the case of decision making the learning-improvement loop must be closed. Based on results, decision outcomes are analyzed and improved.

DMSs are designed to meet these goals as they enable enterprises to model and automate their decisions. With decisions being modeled in a central place, they become visible to management and thus can be used as strategic tools. The decisions can be reviewed on a regular basis in order to verify whether they are still aligned with the business goals. Each decision is made as a service to all other business applications, ensuring that the policy is enforced consistently and that it is reused throughout the application portfolio of the enterprise.

Let us come back to our former example of the portfolio value policy. These kinds of policies are simple enough to be encoded in an executable format. A DMS enables this and Business Process Management software allows for hooking the policy in the overall business process. Each time the portfolio value of the customer changes, we have what we call a business case that requires an action. This business case is passed into the decision service that was designed to decide *what* action should be taken. The resulting decision flows back into the IT-infrastructure, ensuring that the actual action is performed. In this case it could be the enabling of a monthly email reminder, sent to the consultant responsible for dealing with the respective customer.

Although a DMS addresses many of the above questions, still important questions are left open: How can management learn whether or not the right decisions are made and what happens if a decision changes? How do customers with a portfolio value of more than a million develop, given that they are contacted once per month? What would happen if they where contacted twice a month or once each two months? Do we need to adapt the schedule of our manufacturing process? Is there a problem with the medicine prescription of this patient? What price should

be offered to electricity buyers on a certain moment on the day? These questions are even more urgent when decisions are taken at a large scale.

A wrong decision might not be harmful in an individual case, but when many of these wrong decisions add up, they can pose great risks. Often it cannot even be known on forehand if a decision is a good decision, such as in the case of eligibility. Only when new information comes in over time, such as persons that where eligible for a loan who became defaulters, it becomes clear that the decision was wrong. Learning about and understanding the causes of the aggregated effects of these kind of phenomena are crucial in modern global economy.

1.2 Thesis Problem and Approach

This thesis focuses on the central problem of designing a system that allows business analysts to understand the aggregated effects of massive amounts of decisions being made by DMSs. Before we can introduce the main research question of this thesis, we first need to define the datasets which together define the structure, logic and execution of automated decisions. A DMS involves the management of large datasets of different kinds with strong relations to each other. Specifically, these datasets consist of a Decision Model (DM) and a collection of decisions that have been made over time. A DM models a business decision in an executable way by modeling the business objects that are subject to the decision and the rules that describe how a decision is being taken. We therefore have three related datasets of interest for a particular business decision:

1. *Domain Model*: description of the business domain. An ontology that describes the attributes and allowed values of the instances that serve as input for a decision and instances that represent the actual decision being made for a given input. The domain model is part of the DM.

2. *Decision Logic*: In all generality this logic is a program. In practice it is a set of rules describing business policies, regulations and constraints. These rules define how a decision is being made and what actions should be taken. They may be structured in control flows and decision tables. The decision logic is part of the DM.

3. *Decisions*: Each decision being made results in the storage of three objects: a trace of executed business rules, an instance of the Domain Model for the input object(s) and an instance for the decision outcome(s).

The domain model and the decision logic are both part of the Decision Model (DM) and can be analyzed statically (i.e., withouth any decision being taken). Making sense of and manipulating such datasets already raises challenges when they reach over a few hundred artifacts, as demonstrated by existing Semantic Web research [5]. In practice, a DM will have hundreds of concepts (terms describing objects from the business domain) and thousands of attributes in the domain model, thousands of rules that describe the decision logic and millions of executions. Certain events, such as changes in regulations or changing physical circumstances, force enterprises to make changes to the DM's of affected decisions. However, as pointed out by Taylor, managing decisions means that supervision of decision takes place to ensure that improvement and optimization of decisions are ongoing and proactive [4, p. 42]. Active supervision and understanding of the decision performance, enables an enterprise to pro-actively deal with risk and changing markets, and to find new business opportunities. Summarizing the above, we can state our central research question as follows:

Research question: *How can we support a business analyst in supervising automated decisions, such that he gains a better understanding of the operating and effectiveness of these decisions?*

To understand the operation of an automated decision an analyst must improve his insight in the structure of the DM and the effect of changes. To assess the effectiveness of a DM the analyst must understand if the aggregated results of many decisions actually lead to the expected business performance. We further refine this research question into various concrete challenges:

Q1 How can an analyst gain insight in the overall structure of the domain model and decision logic?

Q2 When a Decision Model is changed, how can a business analyst find out about the impact of the change and whether there are related parts of the Decision Model that might need to change as well?

Q3 What is the structure of the business cases and decisions? Are there correlations between concepts, trends, outliers?

Q4 How do these concepts relate to the decisions being made?

Q5 Which parts of the decision logic are relevant for a given selection of decisions?

Q6 How can a business analyst find opportunities to enlarge the scope of decisions based on captured data?

Q7 How can a business analyst determine the kind of changes that are needed to optimize the business performance of a DM?

To address these problems, we base our research on prior work in information visualization, visual analytics, software visualization, mathematics and statistics and business software to identify partial solutions that help solving stated challenges. We develop a number of generalizations and improvements for visualization techniques involving high-dimensional categorical data in general. Next, these improvements are integrated in a framework that supports the understanding of automated decisions.

1.3 Thesis Contributions

The contributions of this thesis are as follows.

1.3.1 Analytics contributions

- **Model assessment:** We introduce a new and interesting problem domain from industry to the InfoVis and VAST research communities, namely Decision Management Systems. Specifically, we highlight the dual nature of the data that can be extracted from these systems. On the one hand we have the DM which represents a certain view on the business reality that it tries to capture and react to. On the other hand we have an accumulation of business cases which is a concrete view of what has happened until a certain point in time. In the ideal case, what has happened matches with the expectations underlying the DM, but this is unlikely to happen. The relation between explicit knowledge and facts (in the form of business cases) backing up this knowledge, brings up interesting questions and challenges that fit a Visual Analytics (VA) approach.

- **Visual Analytics:** We introduce the Decision Exploration Lab (DEL), an interactive and integrated approach that enables querying and understanding of decisions. Interaction and statistical techniques are added to enable gaining insight in the relation between the business cases, decided actions and decision logic.

- **Evaluation:** We present the evaluation of our approach which was done to assess its operating and effectiveness. This evaluation consists of two parts. First, we did a preliminary user study to evaluate our novel technique for exploratory analysis of multivariate categorical data. Second, we evaluate our VA solution with two scenarios from the car insurance industry.

1.3.2 Technical contributions

The development of DEL resulted in the contribution of various visualization and exploration techniques for high-dimensional categorical data. These contributions can be summarized as follows:

- A generic algorithm for rectangular space-filling layouts. We review various layout algorithms in this category and extract design principles. Using these principles we construct a generic layout algorithm that can be configured to produce a wide range of well-known and new layouts. In addition we show how the complexity of this algorithm is linear in most cases and quadratic worst case with certain configurations.

- We introduce a new interactive approach to analyze high-dimensional categorical data. This approach is based on dimensionality reduction and provides various visual and interactive features to overcome the typical interpretation problems that come with dimensionality reduction.

- Two improvements for scatter plots that have a large number of points. We developed a static labeling method which is able to label areas of a scatter plot in a meaningful way without clutter and loss of information at the cost being able to link individual points and labels. We also developed two methods to determine appropriate adjustments of the scales for a scatter plot in order to reduce the influence of outliers to the visual appearance of the plot.

1.4 Thesis Outline

CHAPTER 2 describes decision management systems in more detail. It introduces the important concepts and the various technical artifacts that are available for analysis and the challenges that these systems bring. This is followed by a discussion of the analytical requirements related to the identified challenges.

CHAPTER 3 discusses related work in the three main areas that revolve around the application domain of DMS and provide solutions for similar (sub)problems: Decison management systems, program comprehension and visual analytics approaches.

CHAPTER 4 presents and discusses information visualization directions to solve the particular problem of this thesis. It presents methods to visualize both the domain model and the business logic using treemaps. Treemaps proved to be an interesting research direction in itself and resulted in a generic algorithm for space-filling layouts. Additionally, we present our work in visualizing rule execution and visualization of the impact of changes to either the business logic or the domain model. The chapter concludes with the observation that visualization of the

available technical artifacts and/or data is not enough to solve stated problems and that analytics is needed.

CHAPTER 5 starts with a discussion on two early approaches of applying analytics to decision management systems based on serendipity. Next an exploratory VA approach is presented for categorical data in general and business cases in particular based on multiple correspondence analysis. This is followed by a discussion on and analytical approach for the analysis of business rule triggering patterns.

CHAPTER 6 presents an integrated solution for the challenge we address and which allows a business analyst to explore decision data. First, the overall architecture of the Decision Exploration Lab is discussed. This is followed by the presentation of the core components of the system: the querying and filtering mechanism and visual decision exploration. The constructions of this systems led to various refinements to earlier proposed sub-solutions, which are discussed here as well.

CHAPTER 7 evaluates the system presented in CHAPTER 6. To this extent we first discuss what it takes to evaluate a solution for DMSs. This is followed by two scenarios from the car insurance industry that demonstrate how the system can be used to understand the functioning of a DMS.

CHAPTER 8 concludes the thesis by summarizing the contributions of this thesis and outlining open challenges with respect to the understanding of the functioning of DMSs.

Chapter 2

Decision Management Systems

> Waiting hurts. Forgetting hurts. But not knowing which decision to take can sometimes be the most painful.
>
> *MI VIDA*, JOSÉ N. HARRIS

Decisions are becoming an increasingly important asset in modern business. In highly competitive complex and rapidly changing markets, it is important that business decisions are traceable and adaptable [6, 7]. Decision automation is one approach to reach those goals [8]. The value proposition of a Decision Management System (DMS) is that it separates the business *logic* from the enterprise *applications* implementing business processes. It encompasses encoding of business knowledge in an executable Decision Model (DM). As opposed to the more traditional approaches, these models may not be created and maintained by IT-departments, but by business analysts, or more generally knowledge engineers.

DMSs are commonly used in combination with Business Process Modeling systems. The latter are used to model and track the end-to-end business process, while DMSs are used to model and automate individual decisions that are part of the overall business process. From a high-level view a DM can be thought of as a function that takes an input and results in an action.

This chapter gives a high-level overview of DMSs, based on IBM's Operational Decision Manager (ODM) [9]. However, the presented concepts apply to similar systems such as Drools [10] and FICO Blaze Advizor [11] as well.

2.1 Applications of DMSs

Not every decision lends itself for modeling in a way that allows execution. The decisions involved when designing a new building, for example, are highly specific and often one-time decisions. Decisions suitable for automation with a DMS have different properties, the most important being that they are highly repetitive. They also slightly change on a frequent basis due to events such as changes in regulations or customer preferences. As a result, DMSs can be found in a wide range of applications such as health care, tax processing, insurance and fraud detection.

When prescribing drugs to a patient a number of things must be checked in order to make sure that the prescribed drugs does not cause any harm. Questions like: Is the dose not too high? Does the prescribed drug interplay with other drugs that the patient is currently using? And has the patient any allergies? must be answered properly before the drug is actually prescribed. These kind of questions can by directly answered by a human specialist but are prone to human error. Systems that have access to a patients dossier can answer these questions automatically and raise a warning to the right persons.

Reducing fraud is an ongoing topic for financial institutes such as tax authorities and banks. Often data mining is used on past data to learn about the properties of fraudulent transactions. Based on these findings decision rules can be found that help to identify new transaction that have a high probability of being fraudulent. This helps focusing the fraud detection and research resources and helps reducing fraud in a more efficient way.

Another typical use case for DMSs is the insurance industry. Based on customer input, an insurer has to find a balance between the risk this new customer would bring and the offer he has to make in order to convince the customer to take the insurance. This balance often results in a eligibility check and in a quoting decision. The former decides whether or not the person can become a customer, while the latter decides the quote and possible deductions and surcharges that make up the final quote.

In order to make our discussion, with respect to the problems we want to address, more concrete, we use a scenario from the car insurance industry. This scenario will serve as a running example through the remainder of this thesis. Let us now first consider this scenario, the processing of car insurance requests, in more detail.

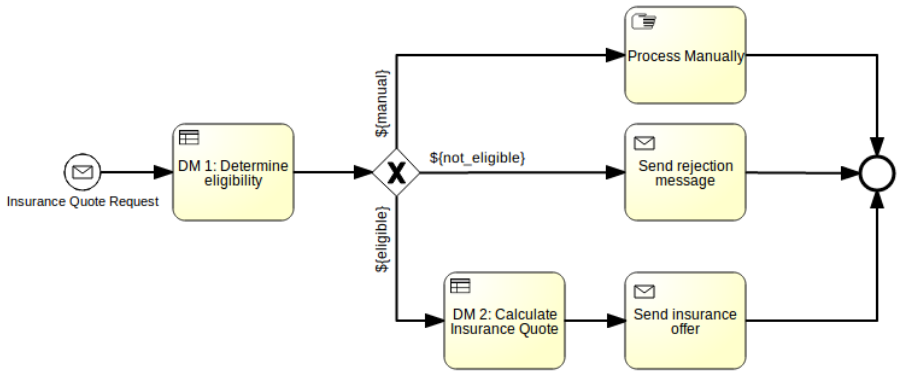


Figure 2.1: Overview of the car insurance scenario: A request for a car insurance, containing information about the driver, the vehicle and the requested insurance is processed by two decision models (DM 1, DM2) resulting in either manual processing, a rejection or a pricing offer.

2.2 Scenario: Car insurance request processing

The scenario describes the business process of determining the eligibility of a person and the quote of the insurance. Given some customer input, an insurance quote is calculated or the customer is notified that he is not eligible. This process is supported by two decision models that deal with risk management ($DM1$) and pricing ($DM2$). Figure 2.1 gives an overview of this scenario using the Business Process Modeling Notation (BPMN) [12].

Both models have as input an $AutoQuoteRequest$ object, which details information about the driver, the car and the requested insurance. This object is created by a front-end such as a web-form that requires the customer to fill in the required information. The risk management decision model $DM1$ determines if a driver is eligible for an insurance. Its output is one of the following:

- *Eligible*: the system could automatically determine that an insurance can be given.
- *Ineligible*: the system found reasons to deny the request.
- *Manual processing*: the system could not take a decision, and a human operator should intervene.

This model represents a risk trade-off. If too many drivers, having a high chance of being involved in an accident, get an insurance, the enterprise will lose profit due to the high amount of claims. On the other hand, when too many drivers are rejected it might not be profitable at all. The goal of decision automation in this case is to make the best risk trade-off in a consistent and automated way, therefore the number of manual processed requests should be minimal.

Once $DM1$ has determined eligibility, the pricing decision model $DM2$ is triggered. This DM calculates the final quote of the insurance. First it determines a base premium based on the value of the car, the requested deductible and the insurance type. Next it determines all the discounts and surcharges based on both properties of the car and of the customer. For example, a car with anti-lock brakes will get a 5% discount. When the base premium, discounts and surcharges are determined, the final quote is calculated and an $AutoQuoteResponse$ object detailing all this information, is returned. The pricing model represents a trade-off as well. On the one hand, the prices should be competitive with respect to other insurers in order for customers to accept the offer. On the other hand, the prices should cover the expected claims, business running costs and generate profit.

2.3 Decision Models

A DM describes both the business domain and the business logic, thus consists of two parts. The first part models the business domain of the decision. It describes the concepts that are subject to the decision and those which represent the decision outcome as well as the the concepts' properties, or attributes. The second part of a decision model is a program that, given an instance of the domain model as input, generates the output, i.e. decision. Both the input and output parameters of a DM are described by the domain model.

2.3.1 Domain model

A domain model defines the objects, their attributes and the attribute types of the objects that are subject to the decision. As such it serves as a type system for the business logic. It can be described using various techniques such as XML-Schema, relational database schema's or using an object oriented language such as Java. Chniti et al. presented the integration of semantic web technologies in DMSs [13]. An *ontology*, in information science, represents a set of concepts and their relations in order to support reasoning. We will use the term ontology to refer to the domain model and the concepts that it represents even though it might technically be implemented using different techniques.

In our scenario both DMs use the same ontology. This ontology describes concepts such as $AutoQuoteRequest$, $Driver$ and $Vehicle$. For each of these concepts it also describes the properties or attributes such as: $AutoQuoteRequest.driver$, $AutoQuote\ Request.maxPriceLimit$, $Driver.age$ and $Vehicle.hasAntilockBrakes$. ODM also allows the business analysts to specify verbalizations for each of the concepts in the ontologies, e.g.:

```
AutoQuoteRequest.driver: the driver of the request
Vehicle.hasAntilockBrakes: the vehicle has antilock brakes
```

This allows writing sentences in controlled natural language [14] for an ontology. Concepts of an ontology are structured in packages, similar to Java classes. Ontologies can therefore be considered to have a hierarchical structure following the natural grouping of packages, concepts or types and attributes. Additionally, concepts in the ontology can refer to each other. For example, the $AutoQuoteRequest.driver$ attribute is defined by the $Driver$ concept. Therefore, an ontology is better described using a compound tree structure.

Depending on the context, these ontologies can become large and therefore hard to maintain. One typical task in maintaining ontologies is to analyze the structural relationships and coupling between concepts and attributes. In the context of semantic web techniques, ontologies are described with the Web Ontology Language (OWL). A well-known platform for ontology modeling and knowledge acquisition is protégé [15, 16]. Protégé has a number of plugins that

visualize artifacts of an ontology [17]. Given the high similarity with traditional type systems of programming languages, a way to visualize ontologies is to use UML class diagrams. This might be applicable for relative small ontologies such as the one in our scenario, which consists of 22 objects with a total of 94 attributes. However, dataset sizes in industrial applications of DMSs are much larger, containing hundreds of concepts each having tens of attributes. A more successful approach could be the usage of the edge bundling technique by Holten [18]. This would support the analysis of coupling between concepts in the ontology.

2.3.2 Business logic

The business logic in a DM prescribes how, given the input parameters, the output parameters are determined. In all generality, the decision logic in a DM is a program. Because business policies can typically be expressed using $IF - THEN$ rules, they lend themselves very well to be modeled with expert systems. Overall, the decisions taken by these systems are complex and span multiple domains (e.g. law, finance or marketing). However, the logic used in these decisions is often simple. Even though the logic of individual policies might be simple, expert systems such as DMSs are Turing complete, therefore there is no loss of generality.

Business policies are typically expressed as logical conditions that are linked to an outcome when the conditions are met. This property makes business rules suited to express them using a forward chaining expert system. Forward chaining is a computational model that allows for declarative expression of the business logic. In a business context this gives the advantage that the focus can be primarily on the business policies without too much burden of technical details related to programming such as control flow. Because the lack of control flow in declarative languages, the order in which rules are executed is not described either but deduced based on the input, logical conditions and the actions.

Business logic is described by a set of production rules, which in ODM can be represented either with $IF - THEN$ statements or with decision tables. Rules have unique identifiers and are structured in packages. Production rules have a precondition (IF statement), and an action that is executed when the precondition is met (THEN statement), as in the following example:

```
IF
  the vehicle has antilock brakes
THEN
  add a 5% discount to the quote, reason: "Anti-lock␣Brakes␣Discount";
```

Both the precondition and the action can have an arbitrary amount of elements, allowing for the composition of more complex rules such as:

```
IF
  all of the following conditions are true :
  - the age of the driver is between 18 and 21
  - the number of accidents the driver has been involved is at least 1
  - the number of traffic tickets the driver has received is at least 1
THEN
  add a $8 surcharge to the Auto Quote Response,
  reason: "Young␣driver␣surcharge";
```

As stated before, production rules can be represented in ODM using decision tables, as shown in Fig. 2.2. The columns with a white background represent the preconditions and the column with the gray background the action, of which there can be more than one as well. Internally, each line of a decision table is represented as an individual production rule. For example the first line of the decision table in Fig. 2.2 could be written, using the natural language representation, as:

	Vehicle Value		Deductible	Base Premium
	Lower	Upper		
1	$ 0	$ 5,000	$250	$ 43
2			$500	$ 39
3			$1000	$ 27
4	$ 5,000	$ 10,000	$250	$ 47
5			$500	$ 41
6			$1000	$ 32
7	$ 10,000	$ 20,000	$250	$ 51
8			$500	$ 44
9			$1000	$ 35
10	$ 20,000	$ 30,000	$250	$ 52
11			$500	$ 49
12			$1000	$ 40
13	$ 30,000	$ 50,000	$250	$ 62
14			$500	$ 59
15			$1000	$ 48
16	$ 50,000	$ 100,000	$250	$ 67
17			$500	$ 59
18			$1000	$ 55

Figure 2.2: A decision table to determine the base premium of an insurance based on the car value and the deductible.

```
IF
  the value of 'the vehicle' is more than $0 and at most $5000
  AND the deductible of 'the coverage' is $250
THEN
  set base premium for 'the coverage quote' to $43
```

Decisions tables are basically a convenient way to specify decision trees without having to write a large number of rules manually, thus they structure the decision logic in a compact way. More importantly, they are a way to treat numerical input variables, such as the value of a vehicle in this particular case as ordinal variables. The decision table in Fig. 2.2 treats the numerical value attribute of the Vehicle concept as a nominal variable with six categories. The boundaries of these categories are determined by the business analyst based on experience and analysis of the business domain.

Although the forward chaining execution model fits business policies, for modeling a complex decision it is still sometimes convenient to have some control flow. It can be considered part of the business knowledge that a complex decision is being split up in several steps. For example, an insurance quoting decision could first determine the base premium, next determine the adjustments, followed by a calculation of the final quote. Modeling this order explicitly helps retaining insight in complex decisions that would have been hard to retain when the full quoting calculation was just modeled by a set of rules and decision tables. Control flow diagrams allow the user to make explicit how a decision is taken. These diagrams allow business users to articulate how multiple subsets of rules are orchestrated to make the decision. These subsets of rules typically deal with individual topics in the decision such as base quote, surcharges and discounts. An example is shown in Fig. 2.3, which depicts the control flow diagram for the eligibility DM. Each flow task represents either a single rule or a selection of rules. The High-Risk driver flow task for instance, is implemented by four decision rules and a decision table containing an additional three rules.

In this thesis, we work with a simplified domain model and decision logic. That is, the decision model we use does not represent a real decision model that is used in production. We reduced it in size, both in terms of domain concepts and decision rules. This simplified model is sufficient to present all structures that are part of a DMS. Yet, using a simplified model makes it easier to present all required concepts.

2.4 Decision execution

Rule execution engines are responsible for making an actual decision. These engines are typically wrapped in a service oriented architecture to provide a decision service to the other business applications in an enterprise's IT environment. Each time a decision service is called, the rule-

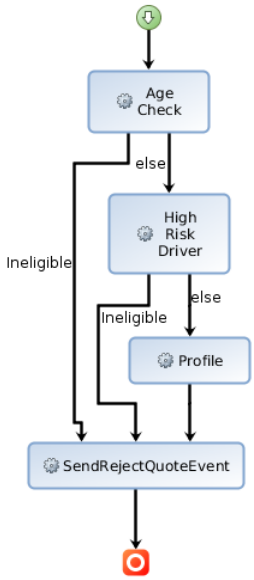


Figure 2.3: Flow chart for the eligibility decision model.

execution engine performs the decision logic for the given instances of the input parameters of the DM. For completeness, we note that such instances are also known under the name of individuals in artificial intelligence and observations in statistics.

Given an instance, the engine determines which rules need to be triggered, and in which order, to reach the final decision. To this end, rule execution engines use, like forward chaining expert systems, inference algorithms based on variants of RETE or TREAT [19, 20]. These engines have a working memory in which the input instances are injected. They evaluate the IF-statement of each rule for the objects in the working memory in order to determine if the rule must be triggered. When a rule is triggered, the action part results in the modification of objects in working memory or the creation of new objects. As a result rules that where until now discarded for execution might again become candidates for being triggered.

As noted in Sec. 2.3.2 the order in which rules are applied can be specified using control flow diagrams. When each rule in the DM is specified by a single flow task, the order of execution is fully defined by the control flow diagram. However, this is typically not the case, both due to the large number of rules in a typical DM and due to the existence of decision tables. A flow task typically is specified to apply a selection of rules which are either explicitly specified or by means of selectors (e.g. all rules in the $insurance.eligibility.HighRiskDriver$ package: $insurance.eligibility.HighRiskDriver.*$).

The exact selection and order of execution of rules for a particular input instance is therefore a side-effect of the rule execution engine. When testing a new or changed DM before deployment, this order can be of importance. In particular when a DM gives unexpected decisions with a test-dataset for which the expected decisions are known. However, from a business perspective this order does not reflect a particular business meaning as long as it results in the expected decision outcome.

After each execution of a decision, two kinds of data are logged in the operational data store of the DMS. First, the instance data for both the input parameters and the output parameters of the DM are stored. Second, the list of rules that were triggered to reach the actual decision are also stored.

2.5 Modeling business activity

Decision automation is a means to react in an automated way to events that relate to the business. A car insurance is requested, a credit card transaction is made, a package is received and so on. In order to react in a rational way to these events, information which is deemed relevant must be captured. Which information is being captured is specified by the domain model (Sec. 2.3.1). Clearly, the full complexity of reality cannot be modeled and this is not what business analysts strive for when they construct the domain model. Using their knowledge about the domain, business policies and regulations, they chose what gets modeled by the domain model and what is left out. Information might be left out of the domain model because it is deemed irrelevant, may not be captured due to regulations, is overlooked or not known about.

By writing control flow diagrams, decision tables and business rules the business analyst models the business logic (Sec. 2.3.2) which determines how is dealt with a particular event. This business logic is restricted by the domain model in what can be reasoned about. That is, a rule can only distinguish between males and females when the gender of a person is actually captured by the model. Rules deal with both constraints that are external to the enterprise and the policies of the enterprise itself.

External constraints are those that the enterprise must adhere to without being able to influence them in most cases. These constraints include for example regulation with respect to the domain of business and physical constraints. An example of such constraints is that the decision logic is not allowed to discriminate on race, gender or religion. Anoter example of such a constraint, is that a package can only be send by airmail from a certain city if there is actually an airport in this city. When reality changes (e.g. new regulations are put in place), these rules must be adapted accordingly. Like with the domain model, not all external constraints will be captured by the business logic.

For the rules that encode the business policies the situation is different. These policies reflect how the enterprise balances its risks and business goals. The policies are based on experience of business analysts, analysis of and assumptions about the business domain. Thus, production rules that implement business policies encode assumptions about reality, such as the following rules:

```
IF
  the driver has been caught for driving under influence
THEN
  set the status of the insurance request to ineligible

IF
  the driver has more than four speeding tickets
THEN
  set the status of the driver to High Risk Driver
```

The underlying assumption of the first rule is that it is never a good idea to sell an insurance to someone who has been caught for driving under influence. For the second rule the underlying assumption is that four speeding tickets is the limit to mark a driver as risky that results in the best balance between risk and business performance. Unlike rules that encode external constraints, these rules can be changed at any time. Changing these kind of rules is a way to reach a different balance between risk taking and business performance. This kind of adaptive control helps an enterprise to keep its strategic advance and to adapt to changes in the market.

With complex models, consisting of hundreds of concepts in the domains model and thousands of rules several problems arise:

1. It is very hard or even impossible to know if all these assumptions are correct.

2. The interplay between these assumptions and the external constraints as encoded in the business logic becomes more complex and consequently harder to fully understand.

3. It becomes harder to determine if information and logic that was *not* modeled should be included after all.

Over time, decisions are being taken with these models resulting in a gathering of facts that can be used to analyze and verify above mentioned assumptions and interplay. These decisions are taken out of human sight on a massive scale, millions of times a day sometimes. As a result, aggregated effects can have a huge impact on the business performance or can result in a huge risk. When for example a decision involving a financial transaction results in a one cent loss for each transaction, the result loss is enormous when this decision is taken hundreds of thousands times a day. Another example is when the risk limits in a loan application decision are almost but just not reached by a majority of the applications. As a result there will be more defaulters than expected in the end because many of them are so close to the risk limits. The problem we try to address with our approach generalizes the above observations, and can be summarized as follows: How can we support the exploration and understanding of how the reality (as captured by the decision model) *diverges* from the reality encoded by actual decisions taken over time?

2.6 Analytics requirements

In order to steer the solution for the above stated problem in the appropriate direction and evaluate the results we need to define the scope of the problem we address. As outlined in [21, Ch. 8], this scope is defined by the targeted users, the tasks that should be supported and the data available for analysis.

In itself, a DMS is just a software system that is not designed for one particular business area. Its goal is to enable enterprises to model their decision logic, whatever the domain is. Thus, one can find these systems in various business domains such as health care, finance and insurance with different goals such as improving efficiency, reducing fraud and increasing sales. Therefore a careful discussion of the above mentioned ingredients of the problem scope is required.

2.6.1 Users

When looking at DMSs we broadly distinguish three kinds of people working with these systems: software engineers and architects, integrators and business analysts.

Software engineers and architects work on the actual implementation of DMSs. They generally have only a very high-level understanding of the possible application domains. Their concerns relate to implementation details of the theoretical foundations of DMSs and software engineering related issues such as the performance and maintainability of the system.

Integrators guide the process of integrating a DMS in a enterprise's IT infrastructure. Integrators have a reasonable deep understanding of the technical details and limitations of a DMS. They understand the role of a DMS in an enterprise in more details. Additionally they have a good understanding of the particular application domain, though not in such detail that they can model all the business logic themselves. Their main concern is the connection of the DMS with the other critical components of the IT infrastructure. Eaxmples are, where does the DMS get its inputs from and which systems are notified once a decision is being taken such that it actually results in an action.

Business analysts have a deep understanding of the particular application domain, related regulations and business policies. They use their specific knowledge to model both the domain and business logic, a process also known under the name knowledge engineering. They are concerned with keeping the business adhere to local and national regulations, optimizing the business performance and at the same time reduce risk. Following the discussion earlier in this chapter, it should be clear that business analysts are the users targeted by our work. It should be kept in mind that, although we limited the target users to business analysts we still deal with a broad group of users. These analysts work, as mentioned before, in different domains and have different competences (marketing, finance, risk-management). As a result we should expect a broad range in the level of skills related to analytical methods and tooling used by these analysts.

2.6.2 Tasks

Introducing a DMS into an enterprise IT infrastructure brings a whole range of new tasks and concepts into the scope a business analysts. Some of these tasks will be straightforward and easy to understand while others might be more foreign to the business analyst.

Authoring and maintenance: The first task an analyst is confronted with is the authoring and maintenance of the decision models for the automated decisions. This consists of modeling the business domain and defining the business logic using control flow diagrams, decision tables and business rules. Depending on the scale of the enterprise, authoring might include access management as well. That is, global business policies are managed by different analysts than local or regional policies. Also different areas of expertise (legal, marketing, risk management) might managed by different analysts. In both cases analysts are only allowed to access parts of the DM that involve their responsibility and expertise. Finally, the DM will change over time as a result of changing regulations or business policies, requiring version management.

Monitoring: Once a DM is put in production its performance needs to be monitored. A business analyst is not mainly interested in the technical performance of a DMS but in the business performance resulting from the decisions being made by the DMS. This business performance is typically expressed in terms of Key Performance Indicators (KPIs). KPIs are metrics calculated over the decisions. These can be simple metrics such as the total number of eligible insurance requests. Sometimes they involve more complex calculations such as the expected revenue which is based on the quotes of sold insurances minus a risk factor times expected damage. Both the risk factor and expected damage come from statistics gathered by the enterprise. Based on experience and business policies, limits will be set for each KPI. These limits can represent both business goals that should be reached within a certain time-frame or border values that should not be exceeded. An important observation with respect to these limits is that natural common sense constraints have to be taken into account. For example, naively optimizing a decision for a particular KPI, will lead to decisions that are not optimal in the long term. When optimizing for revenue one could simply increase the price of the product. However, this will likely result in a lower number of sold products with the obvious negative impact on revenue. Weighing trade-offs with respect to these limits involves common sense and the expertise of domain experts.

Troubleshooting: From time to time it is necessary to investigate a particular decision. For example, a customer might call that his insurance application got rejected while according to the policies he should be eligible. Or the analyst has learned that the business performance is unexpected with respect to certain sub-population and wants to understand why the DMS behaves like it does with respect to this population. In both cases the analyst needs to find a particular or a specific selection of decisions and dive into the details of these decisions. This troubleshoot-

ing might go as far as stepping through a single decision and inspecting the internal state of the working memory in order to detect why the DMS produces the unexpected result.

Simulation and optimization: This can be considered one of the most important tasks of an analyst. Based on his experience and gathered data he will set up simulations that let him determine the best values for limits in decision tables and business rules. These simulations involve comparing the results of different versions of a DM with the same input data.

From these high-level tasks we can extract the following analytical tasks that we want to address with our approach:

T1 Gain insight in the static structure of a DM.

T2 Create selections of decisions for performance monitoring, validation, auditing and testing purposes.

T3 Understand correlations between input data and decisions.

T4 Verify expected and discover unexpected functioning of the decision system.

T5 Find KPI-related opportunities for improvement of existing decisions.

T6 Inject common sense and external knowledge in the discovery process.

2.6.3 Data

A DMS is a complex software system that comprises a multitude of components each of which produces a number of artifacts available for analysis.

User management: The amount of control a business analyst has over the DMs of an enterprise is depending on his specialties and responsibilities. Typically a DMS can be configured for multiple users and assign rights to these users. This allows for tight control who can read and modify which parts of a DM. The resulting artifacts include user details and access rights. Analysis of this data could result in various insights. It can be used to find the analyst who is responsible and has particular knowledge about a given topic. More interestingly, it can be used to create a map of how knowledge is spread over the analysts: If there is one analyst is responsible for most of the DMs, then he is a potential single point of failure. When on the other hand, access (and therefore knowledge) is equally spread over the analysts of the enterprise, this risk is mitigated.

Authoring: This component supports the main activity of users of a DMS: writing rules. It provides several kinds of editors to model the domain and enter business rules in the forms discussed in Sec. 2.3.2. At a lower level there are Application Programming Interfaces (APIs) available to parse these rules, used in the editors for features such as rule checking and auto-completion. The artifacts related to this components are the domain model and the rules specified in domain specific languages. Analysis of this data could help understand the impact of changes, i.e. when I change an attribute of a particular type in my domain model, which rules are affected?

Version Control: For various reasons changes to a DM need to be traceable. In some contexts audits on DMs are performed to verify conformance with regulations. When various versions of a DM are used during the period for which the audit is performed, a non-conforming decision

must be traced back to the actual DM that was used to make the decision. Also in cases of troubleshooting it might be required to analyze a decision with a specific version of a DM. In order to find the analyst that is accountable for a particular version of a rule, each change to a rule must be linked to the author. The artifacts related to this component are the typical version control system data: change set, author and time-stamp. Analysis of these artifacts can give insight in how a DM has evolved over time and who have been contributing to which parts of the DM.

Decision Execution Engine: The execution engine performs the actual decision. It loads the DM and processes the input parameters in order to set the output parameters to corresponding values. An Integrated Development Environment (IDE) tailored for DMSs provides step-debugging a decision execution. In production the engines will be optimized in order to be able to process large volumes of decisions. The artifacts resulting from a single execution are instances of the domain model for both the input and the output parameters of the DM and an object that contains a list of the triggered rules. For a single execution the artifacts do not provide much information. At best they can be used to troubleshoot a particular decision but to really understand why the execution gave the unexpected result the decision will have to be executed again in a debugging environment.

Logging: The decision execution engine is only responsible for performing the decision. In principle the software stack can be configured to emit an event containing the decision outcome and then remove the instance data from memory. For auditing, troubleshooting and analysis purposes this information needs to be stored though. The artifacts of this component therefore consist of an accumulation of decision execution data, i.e. instances of the input and output parameters and an execution trace. This data is serialized into a database.

Scale: Ontologies of a DM are complex, they consist of large amounts of concepts, which are a mix of categorical, numerical and temporal variables. Moreover, decision automation is usually applied in areas that have a high throughput. This leads to a large collection of facts with a high complexity that has a strong relation with the business logic. For instance, the above described scenario has the following metrics. The ontology for the Decision Models consists of 22 objects with a total of 94 attributes. The associated decision logic has 166 production rules. As input-set we used a sample of 100000 insurance requests, consequently resulting in 100000 decisions for $DM1$ and about 80000 decisions for $DM2$.

Dataset sizes in industrial applications of DMSs are much larger. For instance, IBM domain experts have been applying decision management in various commercial contexts such as the loan and credit card industries. A typical use case in the loan industry has a decision model which consists of 100 concepts with an average of 20 attributes per concept. The business logic in this model is described by 2600 production rules. The throughput for this decision service is in the range of hundreds of executions per second. The IBM credit card decision model has 1300 concepts. For this model, multiple rule-sets are used, each ranging between 10000 to 20000 rules. The throughput of the decision services in this case is of 2400 decisions per second on average, with peak days of 55 million decisions. With these large amounts of decisions that are taken out of human sight, aggregated effects can easily start happening and go undetected.

These data sources form a heterogeneous set containing large amounts of data related to the structure, evolution and functioning of a DM. In this thesis we are focusing on the structure and functioning of DMs and will therefore mainly involve the following three data sources:

DS1 **Decision model:** The ontology and production rules operating on this ontology that describe how decisions are made.

Users \Tasks	T1	T2	T3	T4	T5	T6
Software engineers & architects						
Integrators	Q1, Q2	Q5				
Business analysts		Q3, Q5 Q6, Q7	Q3, Q4 Q6, Q7	Q3, Q4 Q5	Q3, Q4, Q5 Q6, Q7	All

Table 2.1: Relationship between users, tasks and our research questions.

DS2 Decision instances: Instances of the ontology for which a decision has been made according to the decision model.

DS3 Decision execution traces: A trace of triggered production rules for each decision.

2.6.4 Relation to research questions

We now have identified possible users and also the tasks that we want to support. The number of challenges to solve is vast and we cannot address them all in within the scope of this thesis. In table Table 2.1 we lists in each cell the challenges that we outlined in Sec. 1.2, which are (partly) addressed when performing the corresponding task. From the table it is clear, that we focus our research mostly on tooling for a business analyst, who is interested in optimizing the business performance. However, some of our work is also useful for integrators who are, among other things, concerned with the run-time performance of DMs. The table does not sketch the full extent of the challenges related to DMS, but it marks the scope we try to address in this thesis and illustrates the fact that this scope part of a broader problem domain.

2.7 Similarities and differences with program comprehension

From the above discussion on the scope of our problem domain in terms of users, tasks and data we see that this problem domain has quite some similarities with the domain of software engineering in general and program comprehension in particular. By program comprehension, we understand here the entire set of activities performed by a software engineer for gaining knowledge about the structure, behavior, and evolution of a computer program [22]. Table 2.2 gives an overview of the most clear similarities.

Besides these similarities, there are also some important differences. First, tools such as version control systems, integrated development environments, debuggers and profilers belong to the standard toolset of most software engineers. Such tools provide well-delimited and well-specified functionality which is closely bound to the actual task. For most business users many of these tools and concepts are foreign. In addition some of these tools (such as version control) are seamlessly integrated in decision management systems and therefore likely not understood as clearly separate concepts.

Secondly, performance has different meaning in both contexts. From a software engineering side, performance is typically related to the actual runtime performance of a software system. For example, in case of a DMS a software engineer would be interested in the number of transactions that can be processed by the system. A business user is interested in the business performance, such as the number of ineligible applicants and the number of insurance offers. In addition to that, additional information relating to the business performance might become available later: An accepted loan resulted in payment issues after some time or the number of claims related to a sold insurance.

Software engineering	DMS
Tasks	
Software engineers write code, including classes that model the problem and control flow to get the expected functioning.	Business analysts write domain models and business logic to automate business decisions.
Software engineers use version control systems to keep track of the programs evolution.	DMSs provide version control for decision models for accountability and auditing purposes.
Software metrics are extracted and monitored over time for software quality purposes.	KPIs are calculated and monitored to maintain an acceptable service level.
Debuggers are used to troubleshoot malfunctioning of a software system.	Decision executions are debugged to understand why a faulty decision was taken.
Profiling tools are used to analyze run-time performance of software and to find areas in the software that should be improved.	There is a need to understand the resulting business performance related to automated decisions.
Data	
Access to software repositories is usually managed on a comitter base.	Access to decision models is managed based on responsibilities.
Source files structured in directories.	Ontology, business rule and rule flow representations stored in directories.
Detailed information of who changed what and when in VCS.	Detailed information of who changed what and when in VCS.
Compiled executables that can be tooled to log program execution traces.	Execution engine that can be tooled to log decision execution traces.

Table 2.2: Similarities between software engineering and decision modeling and automation.

Finally, even though software engineering problems can be complex and require advanced tooling, in the end many of the questions pertaining maintenance activities throughout its lifetime are well defined. When a software system is developed, the development is based on requirements and at the end of the day software engineers want to know if the requirements are met. Still, the requirements may be ill-defined but this is a problem that is not so much related to software engineering itself but to the requirement engineering process. As we have detailed in Sec. 2.5 a DM models both well understood constraints as business policies that are expected to give the right balance between the various business goals. Consequently, the process of understanding if the balance is met, if there is room for improvement and understanding deviations from the expected functioning is an exploratory process involving open-ended questions.

In the domain of software engineering, the field of software visualization evolved, which visualizes structure, behavior and evolution of software [22]. Various exploratory systems have been devised in order to address a broad range of software engineering related problems outlined above. In the context of DMSs we have, like in the software engineering domain, a heterogeneous set of data sources. The similarities with the software engineering domain, together with the

observation that, in the domain of DMS, questions are more open-ended and require common sense and external knowledge, seems to suggest that a Visual Analytics approach is appropriate for the problem at hand.

2.8 Summary

In this chapter we have introduced decision management systems. These systems are suitable for automating highly repetitive enterprise decisions. Throughout this thesis we use as an example of such a decision, eligibility for and quoting of car insurances. A decision model consists of a domain model, which describes the concepts, and the decision logic, which describes how a decision is taken for a given business case. We have argued that a decision model cannot model the full complexity of reality and that therefore, a divergence might take place between the expected functioning of a DM and the actual results of decisions taken over time. We also have pointed out the similarities of comprehension tasks in the context of DMSs with program comprehension tasks. In the next chapter we review relevant literature in the fields of decision support, program comprehension and visual analytics

Chapter 3

Related work

I go checking out the reports - digging up the dirt
You get to meet all sorts in this line of work

Private Investigations, MARK KNOPFLER

Information visualization and Visual Analytics (VA) have become a field of research with a solid body of knowledge [23, 24, 25, 26, 27]. However, as we have seen in CHAPTER 2, our problem is much wider than just the need for visualizing a certain dataset. Decision automation has close ties with artificial intelligence on the one hand and decision making on the other hand. We also showed in CHAPTER 2, that on a technical level there are many similarities with software comprehension tasks. In this chapter we will therefore review relevant literature on decision support, program comprehension and visual analytics tools and techniques involved in understanding the structure, behavior, and evolution of such systems.

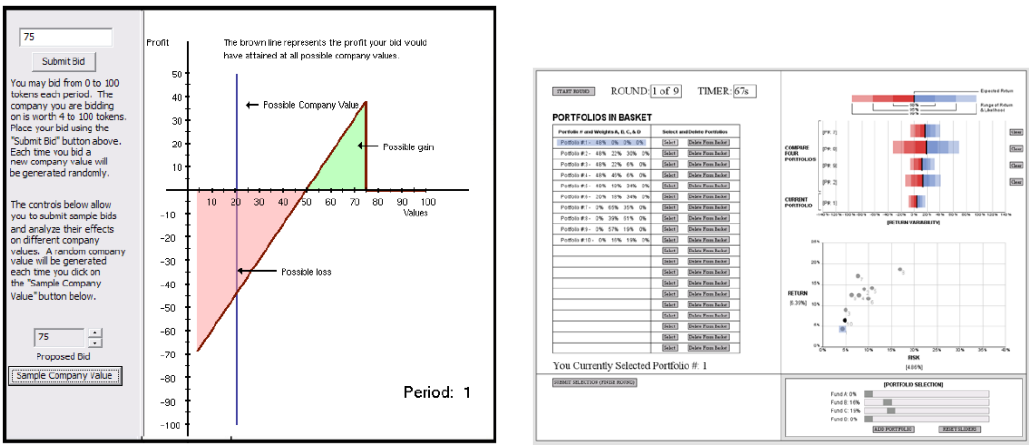
3.1 Decision Support

Decision automation has its roots in artificial intelligence. Knowledge modeling, representation, reasoning over knowledge, and consistency checking are topics from this field that closely relate to our problem domain. Another related topic from this field is machine learning, where predictive models are built from observations. This is different from what we try to do as we have an existing model expressed as IF-THEN production rules and want to use observations to let users discover how *suitable* this model is for a given set of observations. For a complete overview of the artificial intelligence field, we refer to Russell et al. [28].

Decision Management is embedded in a broader context called *operational intelligence*. Operational intelligence is the umbrella term for dynamic analysis of real-time data and business events and operations. A commonly known aspect of operational intelligence is business intelligence which involves data warehousing and data analytics in order to monitor and analyze data streams resulting from business activities. For analyzing data, data mining and predictive model building a wide range of tools is available such as SPSS [29], R [30], STATA [31] and Matlab [32]. The main focus of these tools is the analysis of tabular data, which is too limited for our purposes as we also have graph structured data that we want to take in account. More importantly, our analyses target questions located at a higher level than described by a single or a few data tables, so more sophisticated analysis tools are required.

Designing a new Visual Analytics tool or framework that supports decision making is a daunting task. Wang et al. present a methodology and work flow for designing Visual Analytics systems for decision support in organizations [33]. This methodology consists of a two-stage approach, the first stage being observation and characterizations and the second stage being user centric refinement. In the first stage essential analytical processes are identified. These identified processes are transformed into visualization and interaction specifications. In the second stage individual users' analytical processes are incorporated. The usage of a system by individual users is tracked by recording the actions of users. Based on analysis of these logs and qualitative

evaluations, visualizations and interactions are refined. Although the framework provides many helpful insights, it is a general framework that focuses on decision support. It does not fully provide for the particular challenges that come with a Decision Management System (DMS) and our concrete problem of giving insight in automated decisions that are otherwise taken out of human sight.



a. Interactive analysis of the resulting lose or gain in an auction

b. Interactive risk analysis for financial investment portfolios.

Figure 3.1: VA systems for economical problems by Savikhin et al. Images from [34, 35].

3.1.1 Human decision-making support

Several Visual Analytics systems have been developed to support humans in the understanding of complex data and decision making in various contexts. Savikhin et al. [34] address the problem of deciding the optimal bid for an item at an auction. Making an informed auction bid is important to overcome the winner's and loser's curses. These curses relate to naive bidding resulting in either paying too much for the item (assuming that the average bid represents the actual value) or bidding too low and therefore never win the auction. Most bidding subjects fail to take in account that the value of the item is known to the owner and that they bid the expected value of the item. In order to overcome this problem Savikhin et al. present an approach which help users to gain understanding about the relation between the bid, the value of the item and the expected loss or gain. A simple graph (Fig. 3.1a) shows the possible values of the item versus the expected profit or loss based on the current bid. In analysis mode the user can interactively change the bid and generate random item values to analyze the results of his bid.

Another decision problem from the financial domain is deciding the acceptable level of risk when selecting a suitable portfolio of investments. The two questions that arise are if the individual has a clear understanding of the risk level of each asset in the portfolio and if he is able to choose a portfolio that is appropriate for his risk tolerance. PortfolioCompare (Fig. 3.1b) is a visual analytics tool that supports financial planning decisions for financial investment portfolios [35]. In this tool, users can add multiple financial portfolios and analyze risk and return aspects of each portfolio. A portfolio consists of a number of selected funds for which can be allocated for a certain percentage. PortfolioCompare has a number of views that are updated when the active portfolio changes. The *Risk/Return* display, a 2D scatter plot, shows the risk and return aspect of the items in a portfolio. The *Return Variability* display shows the variability of the return for each item in the portfolio. By interactively updating the portfolio and monitoring the views, the user can compose a portfolio that matches his risk preferences.

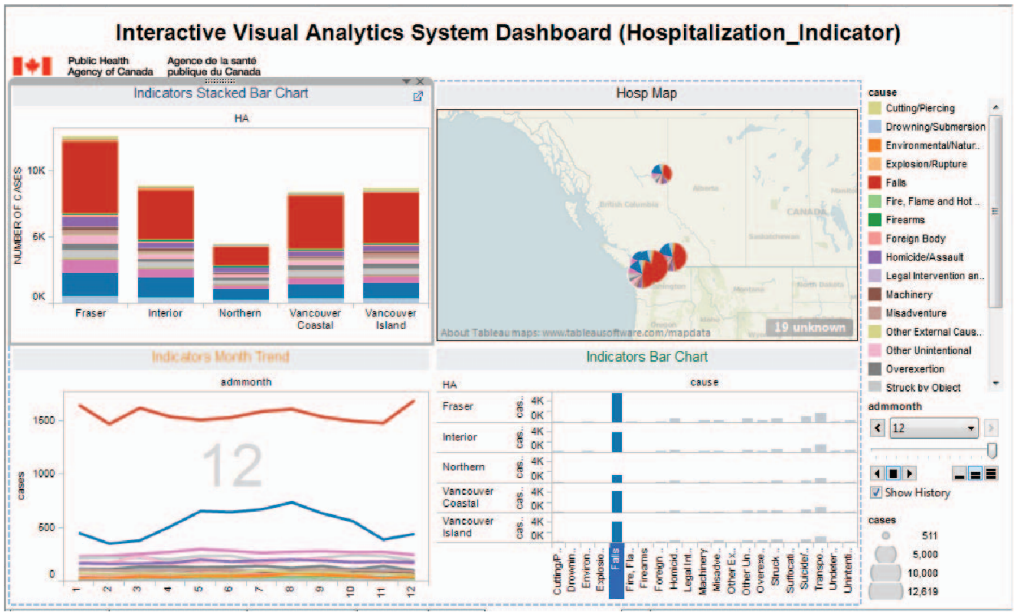


Figure 3.2: Tableau based dashboard for health data by Al-Hajj et al. Image from [36].

AlHajj et al. [36] address the problem domain of injuries based on a dataset containing mortality and morbidity injury data in British Columbia. This massive and complex data, which includes patient details, injury types and geographic location, contains a wealth of information that can be used to support health related decision-making. In order support exploration of this dataset a Tableau-based visualization (Fig. 3.2) was built containing time series visualization and a dashboard. The dashboard view consists of linked views which are updated to reflect changes in one view due to user interaction in the other views as well.

All above tools focus on improving the *human* decision-making process. The tools help domain experts to explore the dataset at hand and gain a deeper understanding of the phenomenon under study. This gained understanding combined with external knowledge is than used to write new policies or regulations. In contrast, we want to give a human insight in, and ways to improve, *automated* decisions. These decisions differ in two ways from the decisions involved in the above cases. In the first place they are automated, it is not a human who makes the decision but a DMS. Moreover, the decisions are in most cases made fully out of human sight. Secondly, these automated decisions are not rare or one-time decisions, but made thousands of times per day or even per hour.

3.1.2 Automatic decision-making support

Understanding Decision Models (DMs) becomes increasingly different when the number of conditions in individual rules and the number of decision rules grows. Wlodyka et al. [37] study the effectiveness of two decision rule visualizations in a medical context. The first representation is a decision tree (Fig. 3.3a), which is basically a node-link diagram based representation of the rules. Each rule is represented by a box that is connected to other rules with a link that represents the rule outcome. The second representation are so called rule-diagrams (Fig. 3.3b, showing one layer), which represent the conditional parts of the rules in a 3D matrix. In each layer of this matrix the rows and columns represent a condition value (or a combination of condition values) and each cell is colored by the decision outcome.

A problem very similar to understanding decision rules can be found in the field of data

mining, which extracts association rules from datasets. An association rule is an implication of the form $X \rightarrow Y$, where X is a set of antecedent items and Y is the consequent item [38]. More concrete, consider the following example of an association rule: a car that has anti-lock brakes and day-time running lights also has triple airbags. That is, $\{anti-lock\ brakes, day-time\ running\ lights\} \rightarrow \{tripleairbags\}$. Two values are calculated for each association rule to determine its usefulness: support and confidence (also known as predictability and prevalence). The support of a rule $X \rightarrow Y$ is defined as the percentage of items in S that satisfy the union of X and Y, where S is the overall set of terms. The confidence of a rule $X \rightarrow Y$, is the percentage of items in the dataset that satisfy both X and Y.

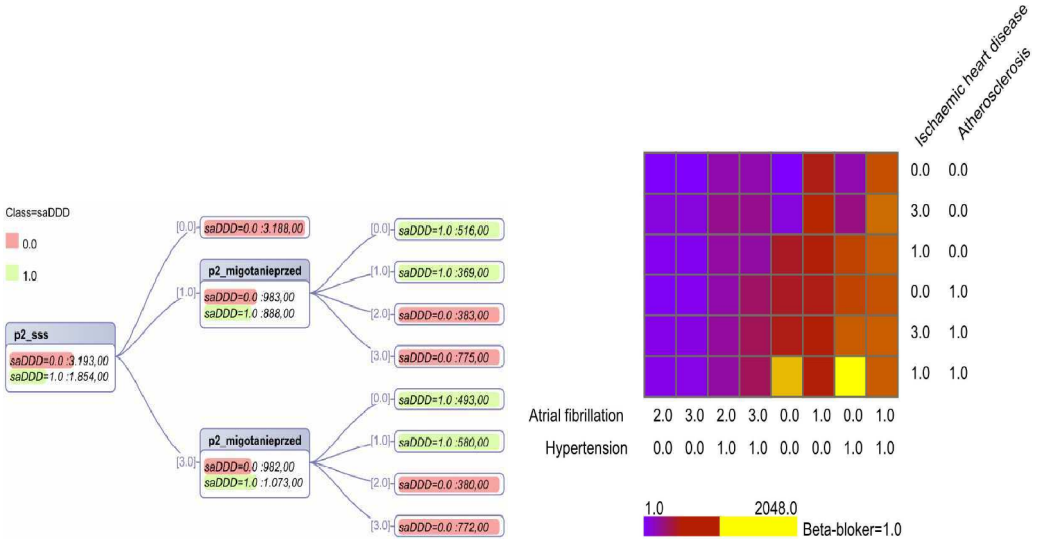
Although not exactly the same, association rules are very similar to decision rules. Using the support or confidence of an association rule, one could use association rules as a basis for decision rules:

```
IF the car has anti-lock brakes
 AND the car has day-time running lights
THEN set the probability of
  the car having triple airbags to 80%
```

Association rules are automatically mined from datasets. Often many rules are found by this process and it is up to the user to find the more interesting rules. Various visualization techniques have been proposed to help the user in this process.

Wong et al. [38] present a 3D visualization as shown in Fig. 3.4. In this visualization rules and items or topics are presented in a matrix. Each row represents a topic and each column represents a rule. The antecedents of a rule are marked blue in a column and the consequence is marked red. On top of the matrix a bar chart is showing both the confidence and the support for each rule.

Blanchard et al., [39], also present a 3D visualization for association rules as shown in Fig. 3.5, but use a completely different metaphor. They use an arena like view in which each rule is placed, represented with a sphere placed on top of a cone, like spectators in a real arena. The position of the object represents the implication intensity, which is a measure of the rules' statistical sur-



a. Decision tree. b. Rule diagram.

Figure 3.3: Decision rules visualizations by Wlodyka et al. Images from [37].

prisingness. Support is represented by the visible area of the sphere. Finally, the confidence is represented by the cone height.

Hahsler and Chelluboina [40], wrote an extension for the R statistical computing and graphing software to visualize association rules. Similarly to Wong et al. they use matrix views to display rules versus topics as shown in Fig. 3.6 They use 2D matrices though and replace the actual topic (or set of topics) with ids in order to be able to show a larger number of rules. Additionally, they apply grouping to rules (sets of topics) to reduce the number of rules to a manageable amount (Fig. 3.6b). These groups can be interactively analyzed, which result in a similar grouped view but only for the terms that are in the analyzed group.

All above presented methods are based on the assumption that rules, whether they are decision rules [37] or association rules [38, 39, 40], are extracted automatically. The main focus of these methods is on finding the interesting rules among a large corpus, which is typical for knowledge discovery. We, on the other hand are not in the first place interested in finding interesting rules, but in the aggregated effects of decisions resulting from these rules and the relationship between decisions and rules.

3.1.3 Decision outcome analysis

The systems in the previous section focused on supporting making decisions. In this section we discuss some systems that, more in the line of our work, focus on the decision outcome.

When there is an epidemic outbreak various actions (e.g. closing a school or applying antibiotics) are taken to stop the spread of the epidemic. Deciding which actions to take and simulating various chains of actions is important to understand how an epidemic is best fought. Afzal et al. [41] describe a system for interactive what-if analysis of the impact of decisions to fight epidemic spread. The system contains a spatial temporal view (Fig. 3.7, left), which is used to track the spread of an epidemic over time and space. This view is combined with various graphs that

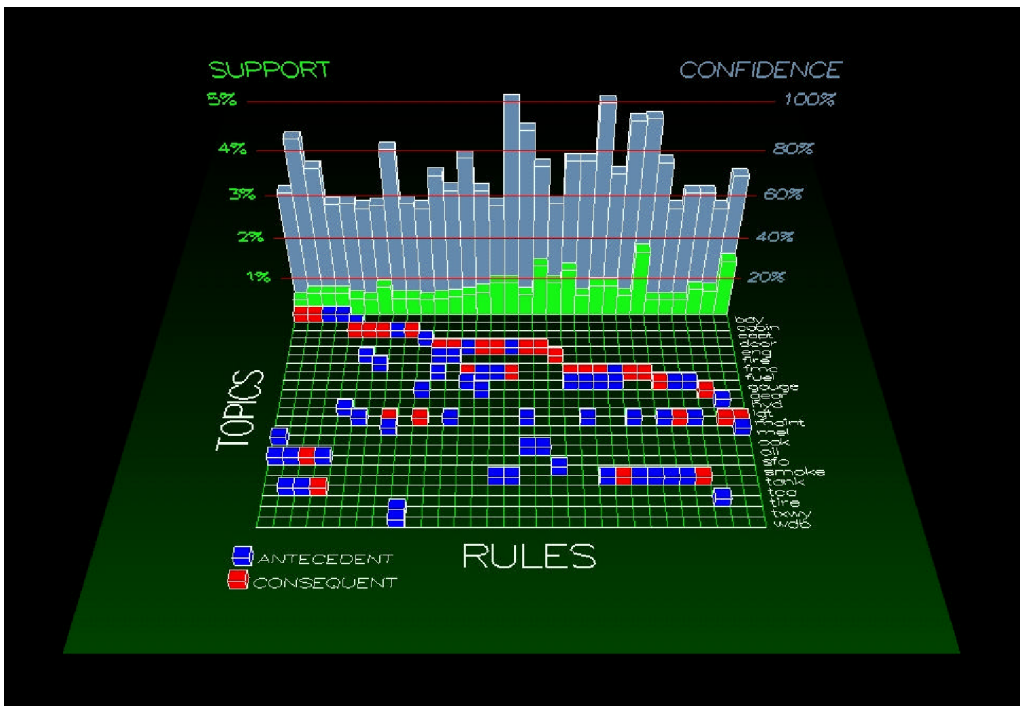


Figure 3.4: 3D visualization for association rules by Wong et al. Image from [38].

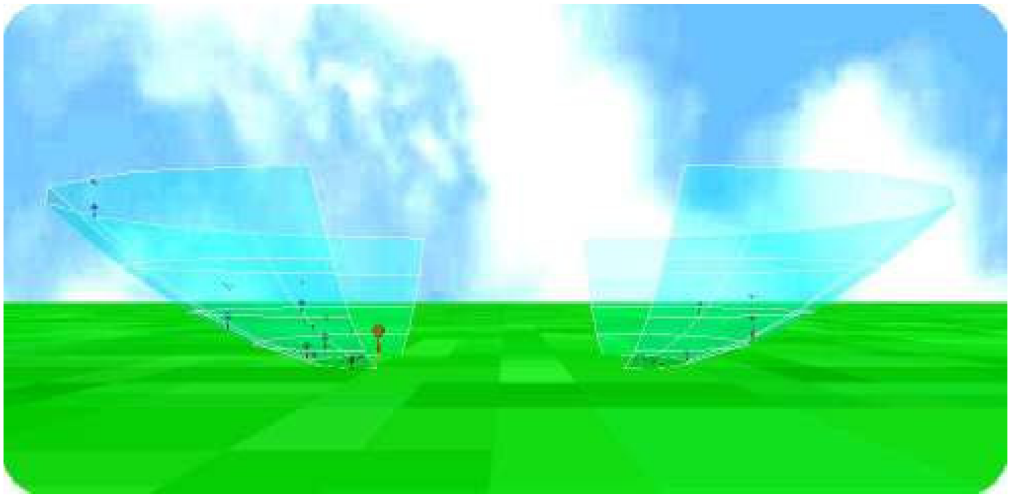


Figure 3.5: 3D visualization of association rules by Blanchard et al. Image from [39].

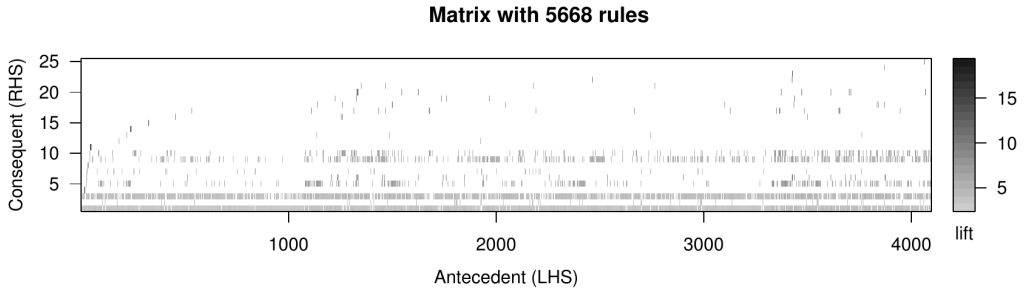
show the development of various variables (e.g. hospitalization case, death cases) at the selected point in time. A second view, called the history tree view (Fig. 3.7, right), shows the development of a variable (e.g. lives lost or number of hospitalizations) over time as a function of taken decisions. This helps analyzing the relation between a set of decisions and the outcome, where outcome is considered to be one variable.

Risk assessment is important in many areas such as medicine, security and forensics. Decision making is based on large amount of multi-dimensional data and foreseeing and weighing consequences is crucial for making balanced decisions. Migut and Worring present an interactive risk assessment framework [42]. Their approach is based on predictive models of the risk at hand (e.g. is a criminal offender going to offend again?). They want to give insight in the models in order to let the user obtain an intuitive understanding of it and thus support his decision making. To this end they focus on decision boundaries of classifiers and data elements that are responsible for a given classifier. The system they propose uses a scatter plot matrix to show all possible combinations of variables. In a scatter plot the data items are colored based on the classification by the predictive model and size shows if it was a misclassification. Additionally, they use Voronoi diagrams on top of these scatter plots in order to approximate the decision boundaries. Ordinal data is treated differently by visualizing it with mosaic plots [43].

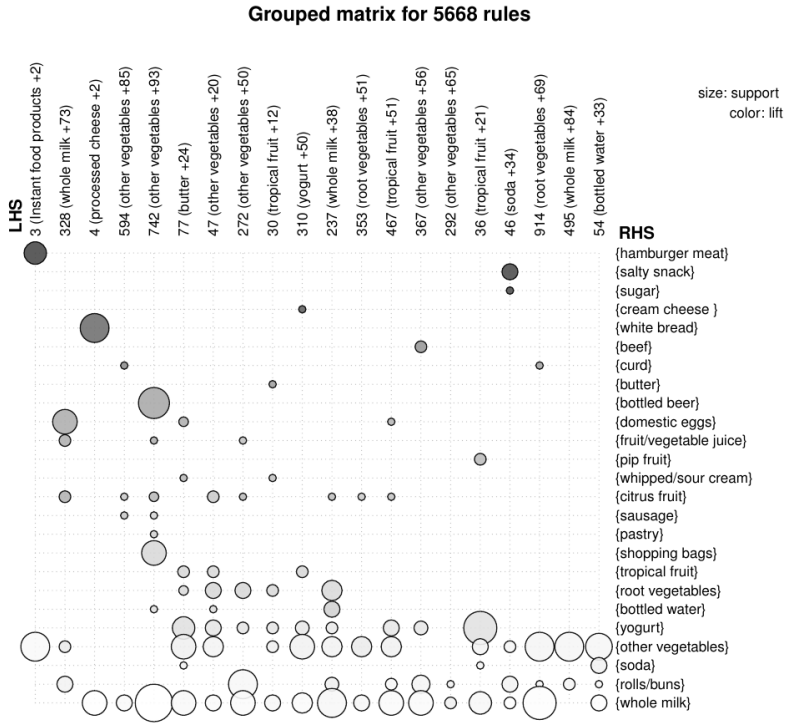
The problem that Migut and Worring [42] address comes close to our problem, but our focus is different: Explore the decisions made by a user-programmed rule-based system (as opposed to a trained classifier) and see how rules and decisions interact. We are interested to see the aggregated effects of the same decision made potentially millions of times. Thus, we do not focus on a sequence of different decisions but on one particular decision and consider that each decision has its own output variables. This is opposed to [41] where all decisions influence the same output variable.

3.2 Program Comprehension

Software visualization aims to visualize the structure, behavior and evolution of software [22] in order to support comprehension of the program at hand. As outlined in CHAPTER 2, there are several similarities between the life-cycle of software systems and decision models, and similarities between the involved data structures. Therefore we review various techniques from this



a. Ungrouped visualization for 5668 association rules.



b. Grouped visualization, using 20 groups, for 5668 association rules.

Figure 3.6: Matrix visualization for association rules by Hahsler and Chelluboina. Images from [40].

field and see if and how they apply to our particular problem domain.

3.2.1 Structure

Structure refers to the static parts and relations of a software system, parts that can be computed or inferred without running the program [22, p. 3]. In software systems this includes the source code, data structures, static call graph and module organization. Many of these parts can also be found in DMSs. Rules are represented in various domain specific languages, they are organized in packages and control flows have some similarities with call graphs.

Common techniques applied to textual representations of source code are pretty printing and typesetting. Pretty printing consists of formatting the source code to make its structure more clear to the reader. Most algorithms for pretty printing are based on Oppens algorithm [44]. Typesetting source code goes back to Donald Knuth, inventor of TEXthe document typesetting

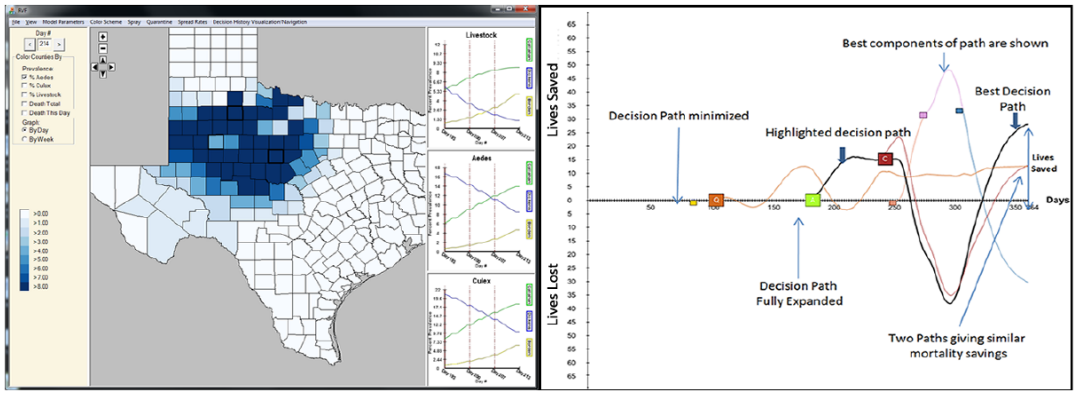


Figure 3.7: VA system to analyze the results of epidemic counter measures by Afzal et al. Image from [41].

system. He introduced term literate programming [45, 46], supporting his view that computer programs should be considered as works of literature. Although this idea did not led to a major change in how software is written, the importance of using typography to enhance the readability of source code was further investigated by Baecker and Marcus [47, 48]. Modern programming environments provide syntax highlighting and formatting, increasing readability of source code, for basically all modern programming and scripting languages. In the domain of DMSs we see a similar approach as shown in Fig. 3.8. Formatting is done manually, to our knowledge there are currently no editors that support automated formatting of controlled natural languages. Syntax highlighting is done automatically, variables are shown in light brown, keywords in dark blue and string values in green.

```
1 if
2     'high risk driver' is true
3     and the age of the driver is between 18 and 25
4     and the number of accidents the driver has been involved is at most 2
5     and the number of traffic tickets the driver has received is at most 3
6 then
7     make this application Eligible with reason:
8     "student risky driver within acceptable limits";
```

Figure 3.8: Formatting and syntax highlighting in IBM Operational Decision Manager.

Various diagrammatic approaches have been proposed to relations between program parts. Jackson diagrams [49] draw programs hierarchically using a node-link tree, with as basic elements actions which are decomposed in sub-actions. Goldstine and von Neumann introduced control flow graphs [50], in which rectangular nodes represent actions and diamond nodes contain branch conditions. Nassi and Schneidermann introduced structograms [51], better known as Nassi-Schneidermann diagrams, which are nested rectangular diagrams to model control flow. A common way to model software architecture and interaction of components is by the use of Unified Modeling Language (UML) diagrams, which is a combination of the methods proposed by Booch, Rumbaugh and Jacobsen [22, p. 58]. Irani and Ware proposed a 3D visualization of software architecture by means of geons, a set 24 primitive, view-point independent 3D objects [52, 53]. As we already saw in Fig. 2.3 control flow diagrams are used in the context of DMS as well. They are used to make the decision structure more explicit. Control flows in a DM therefore help a user, who is unknown with the decision model, to quickly form a mental model of what the decision is about and how it is structured.

Another common practice in software comprehension is the extraction of software metrics. These metrics are single values calculated from software data and can be interpreted as the degree

to which a software system possesses a certain quality [54]. Examples of such metrics are lines of code, number of bugs and number of classes. Display of these metrics is usually done by simple means such as bar charts or spider diagrams. In business intelligence we see a similar practice. Relevant business data is gathered in data warehouses and over this data functions are specified that calculate metrics that relate to the performance of the business. Such a metric is called a Key Performance Indicator (KPI) in a business context. These KPIs are monitored using dashboards and trigger alarms when they violate specified thresholds. Although KPIs somewhat relate to our problem, as they can indicate that a business decision does not perform as expected, they are indeed not more than indicators. These indicators, as is the case with software metrics, just represent a change in some aspect but do not explain *why* this change occurred.

3.2.2 Behavior

Another common task related to program comprehension is that of runtime analysis, i.e. given a certain input what is the behavior of the program. Probably the must used incarnation of this task is step-by-step debugging that is supported by most integrated development environments for the major programming languages such as Java and C++. To support debugging, software is usually compiled with debugging symbols which allow to extract human readable information from the executable at runtime. IBMs Operational Decision Manager, a software system for decision automation, also has a debug-mode which provides a similar step-through feature for automated decision debugging.

Other applications of runtime analysis require explicit instrumentation of the software. A common approach is to let users create a software architecture in a graphical way [55, 56, 57, 58]. From this architecture, source code is generated which is instrumented and serves as a base for the actual implementation of the software. When the software is run, runtime information is gathered and used to enrich the static representations or to generate sequence diagrams.

An interesting feature of Shimba [56] by Systä et al. is that, when drawing the sequence diagrams, it can make use of the abstractions in the static structure diagrams. Thus, it can group software artifacts into higher-level components. This is interesting in the context of a DMS because the execution of a single rule involves many function calls to all domain objects that are involved in the tests of the rules. For a business user, these low level calls are of no-interest because to him, these low-level calls are technical details which he probably does not understand. The rules that are executed to make a single decision is the level of granularity that makes sense to a business user. Therefore, aggregation of low-level artifacts (function calls) into higher-level artifacts (decision rules) is useful in the context of DMS.

De Pauw et al. [59] present various components to visualize dynamic aspects of software systems such as object allocation, method activation over time and object communication. Ducasse et al. [60] use so-called polymetric views in order to visualize runtime metrics. Two other approaches exploit the relation between execution traces and signals in time [61, 62]. To overcome scalability problems, Cornelissen and Holten et. al. present their tool EXTRAVIS [63, 64, 65], which combines a circular bundle view [18] with a massive sequence view.

In general, tools that visualize the dynamics of a software program, try to support low-level tasks such as the investigation of runtime performance, debugging a certain problem, detect late binding of software components and understanding of control flow in the software. When looking at a DMS from a user's perspective (as opposed to developers of DMSs) most of these problems are only of minor relevance. Runtime performance is expected to meet the requirements of the business environment but when it does not, it is up to the DMS provider to solve this issue. Debugging is sometimes required for complex DMs, however comparing the number of decisions that require debugging to the number of decisions being taken debugging can be considered of minor relevance. Understanding control is an interesting issue. Control flow is

made partly explicit in a DM by control flow diagrams that specify in which order sets of rules must be applied. However, the *concrete* control flow of decisions, i.e. the order in which rules are executed, is considered a side effect of the rule execution engine and therefore does not have a particular business meaning.

More important, the comprehension tools described above are designed to support a developer in understanding *how* a software systems works. What the software system is *supposed* to do is not considered to be a question, this is specified by the requirements. A rule execution engine for example, is designed to perform decisions, given an input and a DM. It will not generate an input nor generate a proper DM for the decision at hand, that would be a complete different system with other requirements. The problem we address, on the other hand, is an open problem. A DM models a decision that meets regulations and tries to optimize business goals. However, these goals are not always clear, they may change over time and optimization depends on the input which changes over time as well. As a result the questions that arise when addressing these problems are at a different level than the questions addressed by above described systems.

3.2.3 Evolution

Change plays a major role in the life cycle of successful software. If software is delivering value after its first release, the need to adapt the software to changing user requirements and environments will come up sooner or later. When software changes it is not only the source code that changes, but also the metrics (Sec. 3.2.1), the software archive, the structure of the software and the coupling.

SeeSoft [66] and SeeSys [67] are tools for visualizing statistics associated with source code. Both techniques use space-filling layouts in order to display statistics for large code bases. In SeeSoft, every file is represented by a box and in this box each line of code is depicted as a single line of pixels. The color of each line represents a metric such as last author, code age, number of bugs or profiling data. Evolution of these metrics is shown using animation, each frame visualizes the current metric at a different point in time. SeeSys uses treemaps because it captures well the hierarchical structure that is inherent to software. The treemap is used to reflect the component, module, submodule and file structure, while color is used to represent the value of a metric. Like in SeeSoft, evolution of these metrics is shown by creating an animation of visualizations at different points in time.

A software archive stores the history of a software project by means such as a Version Control System (VCS) and a bug database. These sources contain a wealth of information related to source files, such as who has been authoring the files or by which bugs it has been affected. VCSs are also used to keep track of released versions of the source code. Many clients exists for major VCSs such CVS, subversion and git, e.g. [68, 69, 70], showing the revision graph using various graphic representations. CVSScan [71] uses an approach that adapts the one of SeeSoft. In CSVScan, a source file is represented by a fixed width column in which each line of code is represented by a line of pixels, but these are of fixed length and do not use indentation. To visualize history of a source file, each version of the file is represented by such a column which are shown next to each other. Each line of code is than colored by a specific metric such as author, type of construct or line status.

A DM is subject to change as well. One of the common reasons for change is the need to adapt business to new regulations. For example, our car insurance DM stores rules that relate to state specific regulations in a separate package for each state. Now, when relevant regulations change, the responsible manager needs a way to verify that this is actually reflected in the DM. A tool similar to CVSScan [71], could be applied as an aid. Using such a tool, he can look at the history of the relevant package and discover that it has not been changed for the last months. Although

there is certainly use for such an approach in DMSs, the evolution of a DM is not the focus our research.

3.2.4 Conclusion

We have seen that various techniques that come from the field of software visualization indeed can or already do solve low-level problems in the domain of DMSs. However, as we pointed out, the questions for decision management systems (Sec. 2.6) are quite different from typical questions in software comprehension. Two main reasons can be identified which explain the different type of questions. Firstly, these differences stem from the different execution model (non-deterministic for rule-sets *vs* deterministic for software systems). Secondly and more importantly, our core question is different: How do the realities as captured by the decision model and the executed decisions diverge? This question is of a higher level than what typical software comprehension tools, such as dependency graphs and executions charts, directly address. Our core question reflects a deep difference between software comprehension and decision understanding. When trying to comprehend a piece of software, the question is not so much: does this software the right thing? Rather the questions are: *how* does the software what it does and does it do its task correctly? When automating a decision, the question of how is subordinate to the what question. This might seem counter intuitive because one could ask: what is unclear about the what of a legibility decision? And indeed, such a decision might behave as expected with the 10 or 20 example use cases that are available at the time of modeling. However, what does the decision *really* do when it is executed thousands or even millions of times. Does it bring the expected balance between risk, revenue and other interests that need to be balanced? This is a much harder question to answer and differs deeply from the typical questions answered by tools used for software comprehension.

3.3 Visual analytics techniques

Nowadays business data typically consists of many dimensions, both categorical and numerical. Studying, analyzing and visualizing such data is critical for understanding the underlying business processes. Recent work on both multivariate and categorical data [72, 73, 74] acknowledges the importance for tools that support understanding these kinds of datasets. In the context of this thesis we are in particularly interested in cause—effect relations. That is, what are the properties of the business cases that result in a certain decision outcome? In this section, we review Information Visualization (InfoVis) and VA techniqes, without making a distinction between the two, we review them collectively.

Metaphorically, we could compare a set of decisions with a statistical experiment, where each decision is an observation. In this experiment we have independent variables, that is all input variables, and dependent variables, that is all decision variables. Note that not all input variables are independent in themselves. Take for example the education level of persons, which serves as an input variable for a decision. The education level of a person is at least dependent on the age but likely also on other social and economical factors. However, these factors are not modeled in a DM, therefore these kind of variables are from a DM perspective independent. Unlike in the typical statistical experiment though, there is not a black box between independent and dependent variables. Decision variables are the result of processing business cases by means of the business rules in the DM. Therefore, decision variables are not only dependent but there is also information available on the structure of this dependence in terms of business rules and execution traces. In this section we take a look at techniques that can help the understanding of multivariate data consisting of both numerical and categorical variables.

Raw data has no value in itself, only the information that can be extracted from it has value. Therefore data must be processed and visualized properly in order to support reasoning for the task at hand [21, p. 1]. Visualizing data in order to extract information is a means to support the analytical process of someone who tries to understand a particular problem domain. Johansson et al. [75] use a broad classification of analysis tasks in the context of analyzing datasets that contain both quantitative and categorical attributes: similarity related tasks and frequency related tasks. Amar et al. [3] state that in this process, generally the person will formulate specific, low-level queries on the data. They argue that information visualization can benefit from understanding these tasks (summarized in Table 3.1) that the user tries to accomplish.

Task	**Description**
Retrieve Value	Given a set of specific cases, find attributes of those cases.
Filter	Given some concrete conditions on attribute values, find data cases satisfying those conditions.
Compute Derived Value	Given a set of data cases, compute an aggregate numeric representation of those data cases.
Find Extremum	Find data cases possessing an extreme value of an attribute over its range within the dataset.
Sort	Given a set of data cases, rank them according to some ordinal metric.
Determine Range	Given a set of data cases and an attribute of interest, find the span of values within the set.
Characterize Distribution	Given a set of data cases and a quantitative attribute of interest, characterize the distribution of that attributes values over the set.
Find Anomalies	Identify any anomalies within a given set of data cases with respect to a given relationship or expectation e.g. statistical outliers.
Cluster	Given a set of data cases, find clusters of similar attribute values.
Correlate	Given a set of data cases and two attributes, determine useful relationships between the values of those attributes.

Table 3.1: The analytical tasks as identified by Amar et al. [3].

Visualizing information on paper or computer screens is bound by the two-dimensional restriction of these media. To escape these flatlands [76], additional information can be encoded by using other visual means such as color, variable sizes of objects, contours, shading and texture, described in great detail in [25]. Choosing the right combination of visual cues is important to avoid confusing visualizations. When for example, perspective is used to visualize a three dimensional view, size of objects matters in determining their location: small objects are farther away, larger objects are close by. If at the same time size of objects is used to encode a variable of the dataset, it will become hard to compare two objects that are far away from each other with respect to this variable.

The visual analytics process tightly couples automatic and visual methods through human interaction to support knowledge extraction from data [21]. To efficiently support this knowledge extraction, a VA system has to provide the user with suitable interaction techniques to explore

the model settings and data at hand. Moreover, interaction support, transforms a one-time information extraction process into an iterative process which allows an analyst to change the focus of visualized data to those that match his reasoning. This importance of interaction has led to various taxonomies that describe the design space of interaction techniques. An extensive overview is given in [77], where they distinguish between four kinds of taxonomies: of low-level interaction techniques, taxonomical dimensions of interaction techniques, of interaction operations, and of user tasks. Their study of these taxonomies led to the specification of seven interaction categories based on user intents. Heer and Shneiderman [78] also use a categorization approach in their taxonomy. The taxonomy consists of three high-level categories: data and view specification, view manipulation, and process and provenance. Each of these categories contains four task types. The approach of Heer and Shneiderman is further refined by Kerren and Schreiber [79], who add the two additional task types reconfigure and adjust to the data and view specification category.

Two common methods to visualize univariate data are Tukey's box plots [80] for numerical data and histograms for categorical data. Datasets with more dimensions and which are a mix of both numerical and categorical data require more advanced visualization techniques. In the following sections we will evaluate techniques for high-dimensional data, techniques which are specifically designed for categorical data and at dimensionality reduction.

3.3.1 Visualization techniques for multivariate data

Various interactive visualizations for multivariate data have been proposed. One of the early approaches is the permutation matrix of Bertin [81]. Permutation matrices are a display and analysis strategy for multivariate data with a medium number of samples. In these matrices each row represents a variable and each column represents an observation. Each value is represented by a bar of which the height represents the actual numerical value. Finding patterns in a dataset using a permutation matrix is done by permuting (hence the name) rows and columns until visual patterns appear. Bertin even realized various mechanical devices, as shown in Fig. 3.9. Given the limited computer power at the time, these devices allowed him to actually make permutations of the data and look for patterns. Permutation matrices have led to various approaches, mainly in the context of graph visualization. Van Ham used it to visualize multilevel call hierarchies [82]. This approach was generalized in Matrix Zoom, by Abello and van Ham [83]. Henri et al. proposed hybrid approach for graph visualization, combining a node-link representation with matrix visualization [84] and MatLink [85], another hybrid approach where links are overlaid on the borders of a matrix. Chang et al. apply a matrix visualization to visualize time-varying data in WireVis [86]. Dinkla et al. present compressed adjacency matrices for visualizing gene regulatory networks [87]. Matrix based visualizations have been proven an efficient means, for graph related tasks in particular [88, 89].

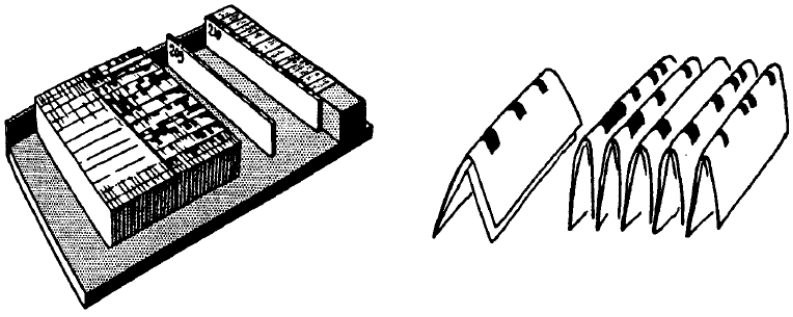


Figure 3.9: Mechanical devices implementing Bertins permutation matrices.

Rao et al. introduced table lenses [90, 91] a multivariate visualization that is in many ways very similar to Bertins permutation matrices. Table lenses represent variables by columns and observations by rows and values are represented by bar lengths. The table lens uses a single pixel line for each observation, allowing for a large number of observations to be shown on the screen. Correlations between variables are found by manual sorting the data on a particular variable and then visually scan the other variables. Categorical data items are displayed by colored patches that have a fixed width and a position in the cell that depends on the value (Fig. 3.10). Additionally, they provide a focus + context mechanism that is based on the fish-eye technique from Sarkar et al. [92]. This technique gives rows and columns around the focus point more height respectively width than the ones outside the focus area.

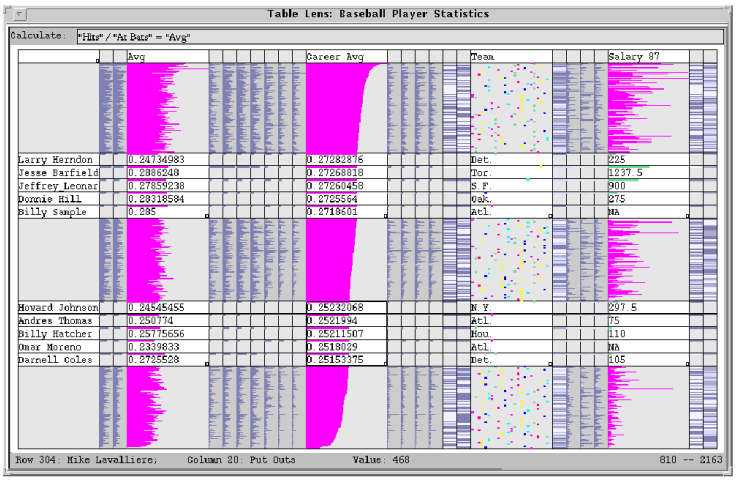


Figure 3.10: Table lens from Rao and card [90].

Table lenses have seen various improvements over time. Tenev and Rao propose a technique to have multiple focus levels and address the problem that arises when the number of data rows is larger than the number of pixel lines [93]. Telea extends the table lens view with the ability to sort on multiple columns and highlights clusters by drawing luminance cushions over the columns [94]. One of the main ideas of a table lens is to show a full tabular dataset in one view. This is also its weakness, because both horizontal and vertical available screen space determine the number of observations and variables that can be shown. When the number of records is larger than the number of vertical pixels either rendering artifacts occur or multiple records must be summarized by one pixel line. The dataset of our interest contains hundreds of thousands of observations or even millions, making table lenses an unsuitable technique.

Feiner and Besher present the worlds within worlds [95] interaction metaphor for n-dimensional data. Their n-Vision system implements this metaphor, which supports interaction with high-dimensional data spaces by nested heterogeneous coordinate systems. The user is responsible for building the world and deciding how to explore it, this is a major drawback as we want a solution that highlights the important information.

Inselberg proposed parallel coordinates [96] for visualizing multivariate data. In this technique, each variable is mapped as a vertical axes. All axes are aligned along the horizontal screen dimension. For each observation or data item a line is drawn through these axes, and crosses an axis at the value that the observation has for the particular variable. Fig. 3.11 shows a parallel coordinates visualization for the Iris dataset. The parallel coordinates approach, transforms the search for correlations in the data in a 2D pattern finding problem. The approach can be enhanced by interaction techniques such as automated or interactive ordering [97] of axes and highlighting

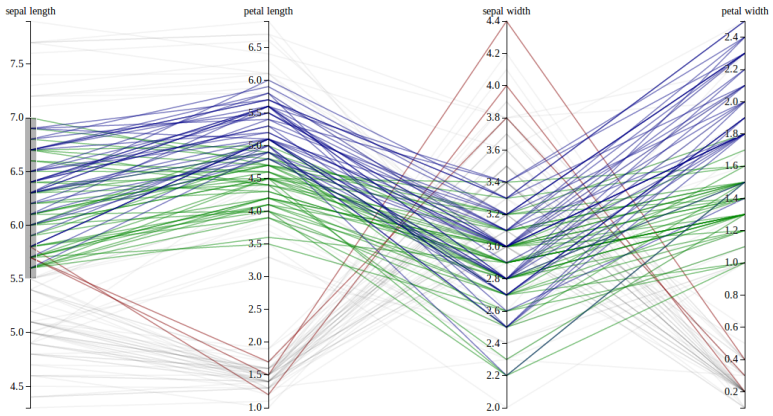


Figure 3.11: Parallel coordinates for the Iris dataset, created with d3.

selections of lines by brushing (e.g. [98]). The Parallel coordinates technique is known to only scale to about a dozen of variables and a limited amount of data points. Too many variables lead to axis that are placed to close to each other to identify any patterns in between. One approach to leverage the problem of too many dimensions is proposed by Yang et al. [97]. Their hierarchical dimension ordering method creates dimension hierarchies based on dimension similarity to reduce the number of displayed dimensions. Too many observations lead to an over plotting of lines, also hiding patterns. This, could be partly solved by using translucent lines, but over plotting will still occur with millions of observations as the case in our datasets.

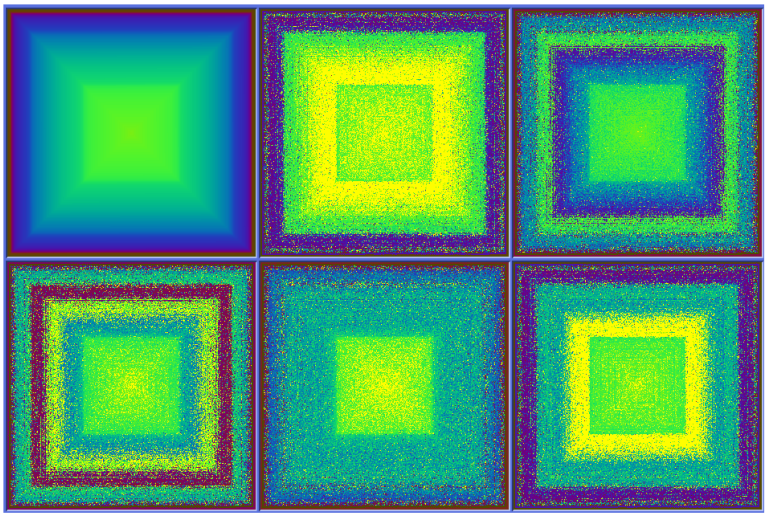


Figure 3.12: Dense pixel visualization in VisDB for a five dimensional dataset with 100 000 items.

Keim and Kriegel present VisDB [99], a dense pixel visualization where each pixel represents one data item. Pixels are arranged based on the query that is performed by the user. Data items that best match the query are laid out close to the center, gradually moving out in a spiral manner for data items less relevant to the query. Based on the visual feedback, users can adapt the parameters of the query in order to explore the dataset. Multiple dimensions are displayed by placing the views in a grid such as shown in Fig. 3.12. Although this method can visualize a large number of data items, finding interesting phenomena still largely depends on the user being able to provide an initial query that leads to insights that help him address the problem at hand.

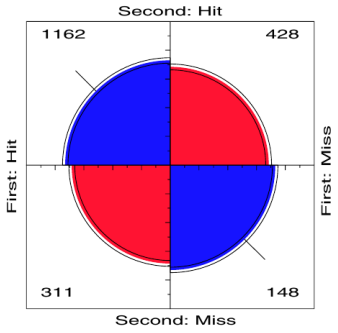


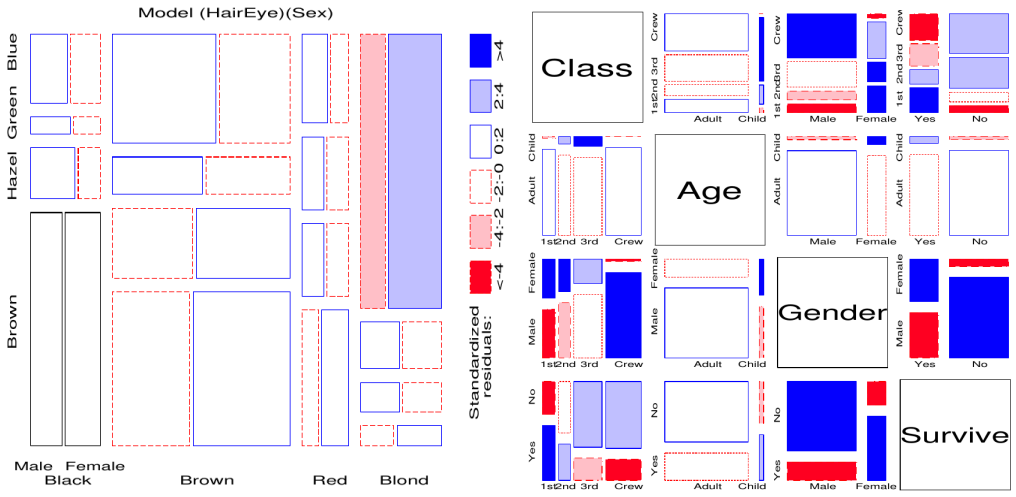
Figure 3.13: Fourfold display by Friendly of "hot-hand" data in basketball.

3.3.2 Visualization techniques for categorical data

Categorical dimensions are frequent in nowadays business data. Examples are gender, car brand, car security features and medication type. Studying, analyzing and visualizing those properties is of importance to understand the underlying business processes. Some of the techniques for multivariate data are less suitable for categorical data. All techniques that use an axis for a variable, such as parallel coordinates, force an ordering on the values of the categorical variable. Order of categorical values is meaningless, for example, when dealing with color, what does it mean that blue becomes before yellow? A particular order of values will create visual patterns. However, a different order, which is as valid, results in different visual patterns.

Many visualizations exist for categorical data, as reviewed by Friendly [100]. Fourfold displays [100] show two-by-two tables such as the "hot-hand" basketball data in Fig. 3.13. A fourfold display is used to visualize odds ratio, e.g. what are the odds that a player has a second hit after the first hit. A two-by-two table is a representation for two binary variables, which clearly is too limited for our use case.

Mosaic plots (Fig. 3.14a) and mosaic matrices (Fig. 3.14b) show multi-way tables [100, 43, 101]. In mosaic plots each tile represents variable value and has a size that is proportional to the frequency. The data is interpreted as hierarchical data. For example the hierarchy in Fig. 3.14a



a. Mosaic display

b. Mosaic matrix of Titanic data.

Figure 3.14: Mosaic displays for visualizing multiple categorical variables.

would be: hair color, eye color, gender. That is, at the first level tiles are split by hair color (vertical gaps that take full height). At the second level, each first level tile is horizontally for each eye color. And finally at the third level tiles are split vertically to represent males and females. This process can be repeated in order to show more than three variables.

CatTrees [102] use treemaps [103] to visualize categorical data. A hierarchical structure of tabular data is initially deduced by assuming that the index of the column represents the depth in the hierarchy. Afterwards the user can interactively remove variables from the hierarchy or place them at a different depth. When it comes to the layout CatTrees produce the same layouts as mosaic plots.

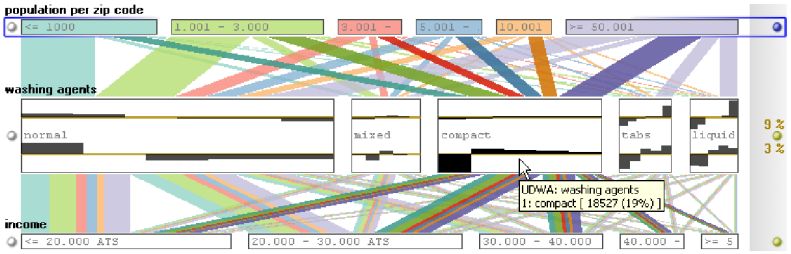


Figure 3.15: Parallel sets by Bendix et al. Showing a customer relationship management dataset containing 93.872 data items.

Parallel sets by Bendix et al. [104] is a technique that adapts parallel coordinates for categorical data. Each categorical value of a variable is represented by a box on the horizontal axis representing the variable as shown in Fig. 3.15. The single lines for each observation in parallel coordinates are replaced with bands that represent the count of items that have selected the values for the two variables that are connected by the band. Recognizing that order of values has no meaning, parallel sets supports interactively reordering of values in order to reduce crossing of bands. However, like parallel sets, the number of variables that can be shown at the same time in a meaningful way is low.

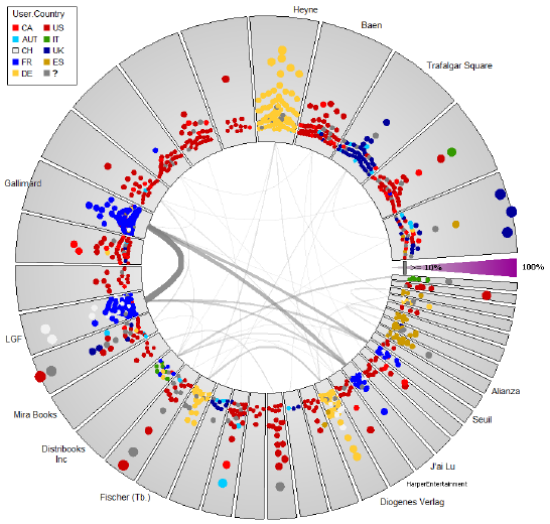


Figure 3.16: The contingency wheel by Alsallakh et al. showing a book rating dataset.

The contingency wheel [73] by Alsallakh et al. shows categories as sectors in a ring chart as shown in Fig. 3.16. The size of each sector maps the marginal frequency of the category. Rows

that have a count for a particular value are drawn as nodes in the corresponding sector. Like the four-fold display, the contingency wheel can only be used to display two variables at a time.

Another approach to visualize multi-dimensional categorical data, proposed by LeBlanc et al. [1], is dimensional stacking. This approach starts by taking the two dimensions with the lowest cardinality and divide the two axes of the 2D visual plane by the dimension that is assigned to the axis. Next, each resulting sub-area is further divided by a third and forth dimension of the data. This way, multiple categorical dimensions can be mapped to the same spatial axis as shown in Fig. 3.17. Dimensional stacking is used and adapted in various techniques and tools, such as in [105, 106, 107]. One of the problems with dimensional stacking, as pointed out in [2], is the combinatorial explosion that quickly occurs even if the cardinality of dimensions is relatively low. As a consequence, in most cases not all dimensions can be visualized at once, leaving it to the user to determine which are the relevant dimensions to visualize.

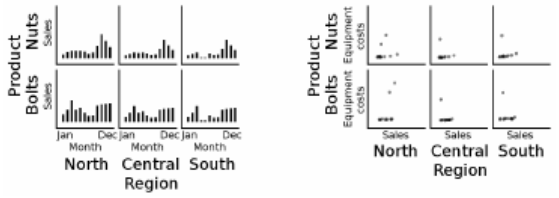


Figure 3.17: Dimensional stacking by LeBlanc et al. [1]. Image from [2]

Emerson et al. [108] observed that scatterplot matrices [23], are hiding features of the data when it contains categorical dimensions. Therefore they proposed the Generalized pair plots, which shows a different plot in each matrix cell based on the data types of the two dimensions visualized in the cell. This technique was further refined and adapted by Im et al. [2] in their Generalized plot matrix, which adds various interactive features and visual refinements to the generalized pairs plot.

Although these methods are specifically designed for categorical data, they are also more focused on, and effective for, *frequency-related* tasks. In contrast, we want to enable exploration of correlations and, in the same time, classification of the data at a granularity that suits the user.

3.3.3 Dimensionality reduction

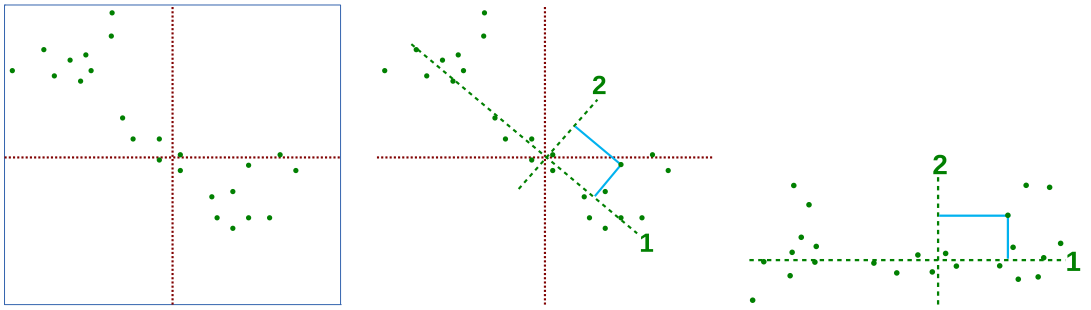
Dimensionality reduction is a way to summarize the structure of high-dimensional data in a lower number of dimensions, usually two or three. These new dimensions give the best representation of the similarity structure of both the observations and the variables. This lower dimensional data space can then be visualized using classical techniques such as scatter plots. Dimensionality reduction techniques have been used in a wide variety of contexts such as analysis of text documents [109, 110, 111], multimedia [112] and biomedical data [113, 114]. In this section we detail the principles of dimensionality reduction followed by a review of techniques.

Dimensionality reduction principles

A widely used technique to study multivariate data is Principal Component Analysis (PCA) [115, 116, 117]. The goal of PCA is to summarize a data table by representing the important information as a set of new, orthogonal variables. These variables are called principal components or factors. Both variables and observations can be plotted in a new map, where proximity of two points reflects similarity.

In PCA, the principal components are defined as the axes in the original data space which represent the largest variance in the data. As an example of PCA we take Fig. 3.18, which shows

an artificial 2D dataset. We start with Fig. 3.18a, which shows the data points in a regular scatter plot, the red axis representing the original scales of the two variables. Next, in Fig. 3.18b, the principal components, i.e. the axes along which the spread of the data is largest, are determined. The principal components are depicted as the green axis and are also known as eigenvectors. The first axis explains the largest spread direction and the second axis, which is orthogonal to the first, explains second largest spread direction of the data. Each data point is now projected on both axes as shown by the blue lines in Fig. 3.18b. The projected values for each point on each axis are called the factor scores. Finally, the factor scores are used to plot the original data points in the new coordinate system of the principal components.



a. An artificial dataset in the original space.

b. Principal components or factors are determined and observations are projected onto them.

c. Principal components serve as the new coordinate system in the reduced space.

Figure 3.18: Dimensionality reduction by principal component analysis.

From this simple 2D example we can make some observations. First, it is easy to see that the PCA of a 2D dataset basically results in the rotation of this dataset such that the horizontal axis shows the largest spread. We can also see that that there is no transformation of the data other than the rotation. That is, the inter-point distance in the new plot is exactly the same as the inter-point distance in the original dataset. Applying, PCA to a data table results in as many principal components as there are variables in the original dataset. Furthermore, each principal component explains part of the variance in the data, and all principal components explain the full variance. Finally, each of the principal components is a linear combination of the original axes. The converse is also true: each observation coordinate is a linear combination of the eigenvectors.

When applying PCA to a data table with more than two dimensions, with the goal to reduce to two dimensions, one of these observations is no longer valid. Since we keep only two eigenvectors, it means we need to ignore part of the data variance. Still, the resulting projection on the first two principal components, can be interpreted as a rotation of the original dataset. However, we are now projecting on two axes while we have N, $N > 2$, variables. As a result, some information is lost: The projected-space distances will no longer be proportional to the original-space Euclidean distances. When using PCA as a dimensionality reduction technique, the amount of information lost when retaining only the two main components can be computed using all the eigenvalues: it is the sum of the $N - 2$ other eigenvalues divided by the sum of all the N eigenvalues. For other (non-linear) approaches, the resulting deformation, or difference of the high-dimensional *versus* the low-dimensional Euclidean distances, is often measured by a so-called stress function. Also, with a growing number of variables it becomes harder to understand what the axes of the projected space mean.

PCA is applied to data tables that consists of numerical variables, while in our case we are also dealing with categorical variables. For categorical variables extensions of PCA are developed, namely Correspondence Analysis (CA) and Multiple Correspondence Analysis (MCA) [118, 119].

CA is a generalization of PCA, for analysis of two categorical variables. MCA is an extension of CA to deal with more than two categorical variables. They both operate on data tables that represent frequencies. The resulting factor scores are usually visualized using scatter plots or biplots [120, 121, 122].

Dimensionality reduction techniques

Techniques such as PCA, CA and MCA are all based on Singular Value Decomposition (SVD). The problem with the SVD is that it is computational intensive, making it less suitable for large datasets. As such, research in various domains, led to a multitude of dimensionality reduction techniques which are faster at the cost of allowing for a larger deformation in the 2D or 3D projection space. Paulovich et al. distinguish between three types of projection techniques: spectral decomposition, nonlinear optimization, and force based schemes [113].

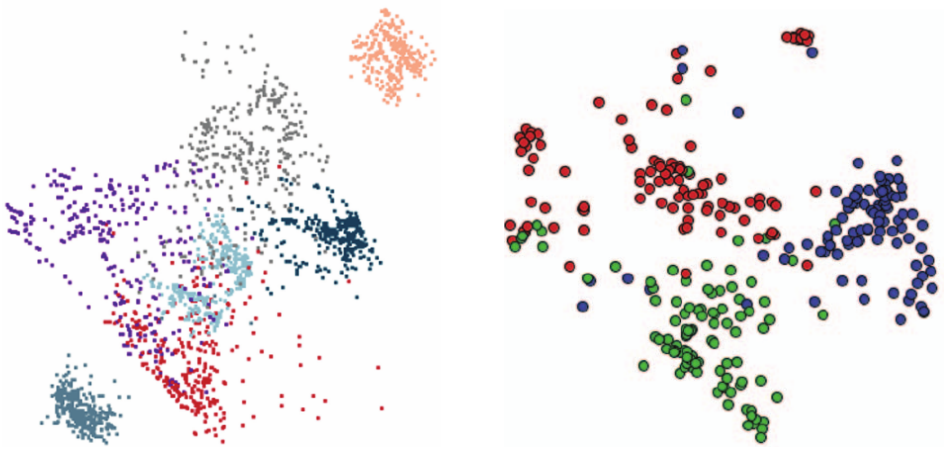
Spectral decomposition: Spectral techniques are also known as classical scaling and they project data points along the eigen vectors that have the largest eigen values of the point-wise distance matrix. Some approaches reduce the calculation time by using numerical methods which are specifically devised to solve sparse eigen problems, such as LLE [123] and Isomap [124, 125]. Even more speed increase can be won by applying classical Multi Dimensional Scaling (MDS) to a small subset of representative points such as done in Landmarks MDS [126] and Pivot MDS [127]. The remaining points are projected by using local interpolation. Fastmap achieves linear complexity in the input point count [112], at the cost of a less well minimized stress cost.

Nonlinear optimization: Nonlinear optimization methods iteratively search the projection parameter space to find a minimum for the stress cost [128, 129]. To speed searching, as opposed to naive gradient descent, multigrid numerical solvers can be used [130]. Pekalska et al. propose a speed-up based on projecting a representative subset (using gradient descent) and fitting the remaining points using local interpolation [131].

Force based schemes: Force-based methods are a special class of nonlinear optimization DR methods, and have been used mainly in the graph drawing area [132]. Chalmers speeds this up by using the small representative subset idea outlined earlier [133]. Further speed-ups are achieved by using multilevel solvers and GPU implementations [134, 135], and by recursively selecting representatives using a multilevel approach [136]. Tejada et al. use a heuristic to embed instances by a force-based relaxation mechanism [137]. LSP positions a representative subset with a force-based scheme and fits the remaining points using Laplacian smoothing [110]. LAMP also uses a representative subset to locally construct affine projections, and allows users to interactively place these points to optimize for the desired overall projection layout [109].

All dimensionality reduction techniques, from the simple PCA to the more complex nonlinear techniques, share acommon problem: Interpreting the results of an otherwise valuable analysis can be hard. The result of these techniques is a 2D or 3D point cloud. Without additional visualization and interaction techniques it is hard to understand such a point cloud. Even more, most of these techniques require many parameters to be configured. These parameters require a deep knowledge of the technique at hand to understand their impact of a parameter setting on both the performance and the resulting projection. Finally, it is not clear how accurately a low-dimensional projection reflects the point proximities in high dimensions, due to inherent projection errors implied by dimensionality reduction. Above mentioned problems have challenged the usage of such techniques in business contexts where end users need fast, easy to interpret,

results and may not have the time or background needed to map the more abstract multivariate data analysis results to their concrete problems.



a. Projection of the segmentation dataset from [138] using the LAMP [109] technique.

b. Projection of a document set using the LSP [110] technique.

Figure 3.19: Dimensionality reduction.

3.3.4 Conclusion

There are various visualization techniques for multivariate data containing both categorical and numerical dimensions. However, none of these techniques can directly and fully address a solution to all our challenges. Indeed, what we need is not just to show similarity relations between data points, but also cause-effect and correlation-effect relations – for instance, which production rules cause a certain decision and how does a correlation of values influence the triggering of production rules.

For multivariate data we identify a number of computational techniques to reduce the number of dimensions, while preserving the similarity structure. The resulting projection is often visualized using either 2D or 3D scatter plots. Points are colored based on an attribute of the original variables, in order to help users assign meaning to clusters, i.e. explain which properties neighbor points share. Fig. 3.19 shows two examples of results presented in dimensionality reduction technique papers: a colored point cloud. Although, [110] mentions automatic generation of labels, the actual coloring is still after manual classification. Even with labeling, there is however, no explicit explanation of what we see and interaction and manual work is required to understand what the point cloud means. This makes it hard to discover classes, to understand their meaning and their relations. Additionally, for end-users the results of all these different techniques are hard and/or not intuitive to learn and use. Thus, even though there are many MDS techniques, the interpretation of their results poses difficulties to end-users.

Sec. 4.1 was published as: Capturing the Design Space of Sequential Space-Filling Layouts
Baudel, T. and Broeksema, B.
in IEEE Transactions on Visualization and Computer Graphics, 18(12).

Chapter 4

Information Visualization for Decision Management Systems

> Meanwhile, feedback control systems where creeping into factory assembly lines, because a mechanical system, too, can modify its own behavior. Feedback is the governor, the steersman.
>
> *The Information*, JAMES GLEICK

The artifacts of a Decision Management System (DMS) provide a wealth of information that can be visualized to support understanding of the Decision Model (DM). Our overall goal is to develop tools which extract information from these artifacts and provide insights. This insight serves as feedback to improve the understanding that a business analyst has on the functioning of the decision. Based on the newly gained insights and his expertise, he should be able to infer if and how a DM should be changed to improve the business performance. Due to similarities with software comprehension, as pointed out in CHAPTER 2, we next apply and refine techniques that have been successfully used in the context of program comprehension. In particular, we apply the following techniques to various artifacts of a DMS: treemaps, graph drawing and bundled edge views. The application of these methods lead to the support of comprehension of the structure of a DM, finding rule execution patterns and understanding the impact of changes to the DM. The chapter concludes with a discussion on why these methods, even though providing valuable insights, do not fully addres the the core question of this thesis: How can we support a business analyst in supervising automated decisions, such that he gains a better understanding of the operating and effectiveness of these decisions?

4.1 Treemaps

In an early attempt to visualize the various data spaces mentioned in Sec. 2.6.3, Baudel applied his discovery framework [139]. Recognizing that these spaces contain tree-structured data, he used treemaps as a visualization tool. Using the parsing and information extraction Application Programming Interface (API)s from ODM he exported the available data in a format suitable for visualization. This enabled him to visualize the ontology and the decision logic (**DS1**) [140].

Fig. 4.1 shows the ontology visualized with a treemap. From top to bottom, the tree levels consist of packages (one or more levels), concepts (one level) and attributes (leaves). Each non-leaf level has a label stating the name of the package or concept and the number of children. Attributes (i.e. leaf nodes) are colored as follows: functions are colored light blue, attributes that have a concept as type are dark blue and primitive typed attributes are colored black. This compact representation allows for the display of large ontologies. The user can interact with the treemap by clicking nodes to get more details about an attribute and use zoom to dive into regions of interest.

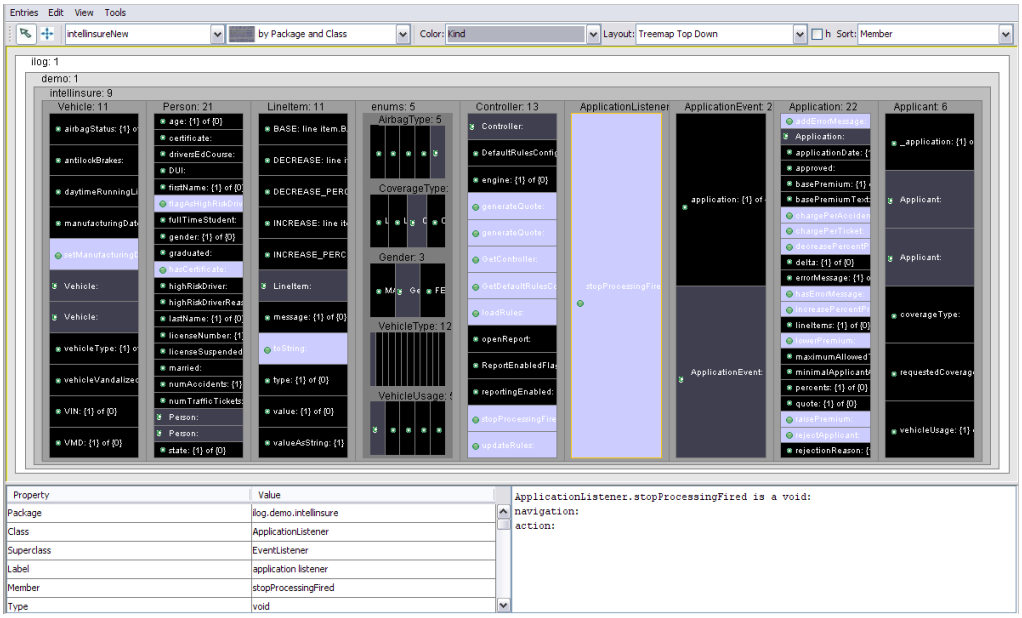


Figure 4.1: Visualizing an ontology with a treemap

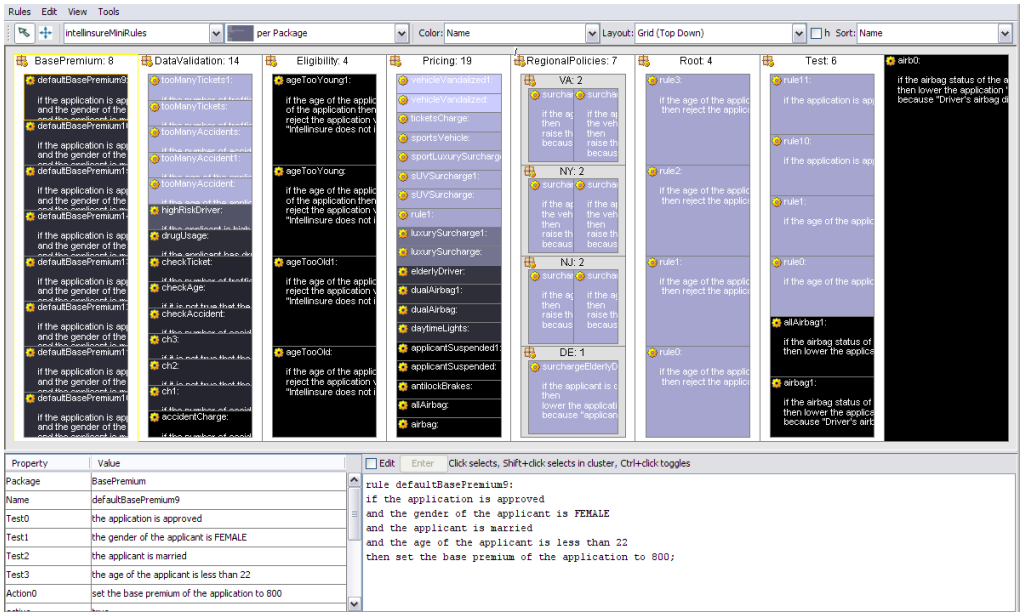
As with ontologies, the logic of a DM can become very complex, in terms of both the number of rules as well as the complexity of individual rules themselves. To support the understanding of the rules of a DM, Baudel used the treemap to display the business logic from two different angles. Fig. 4.2a shows the first angle, which is a straightforward representation of the rules as they are structured in packages. Each non-leaf node represents a level of the package path and each leaf node represents a business rule from the domain logic. This view allows a business analyst to gain a quick overview of the organizational structure of the business logic and dive into a region of interest.

In Fig. 4.2b the decision logic is displayed from a different angle. As outlined in Sec. 2.3.2 the decision logic is expressed as production rules having one or more conditions. Each condition consists of an operator and one or two operands, the conditions themselves are combined by logical operators. Baudel extracted all conditions from a DM neglecting how they are combined with logical operators. Next, he created a tree structure consisting of three levels:

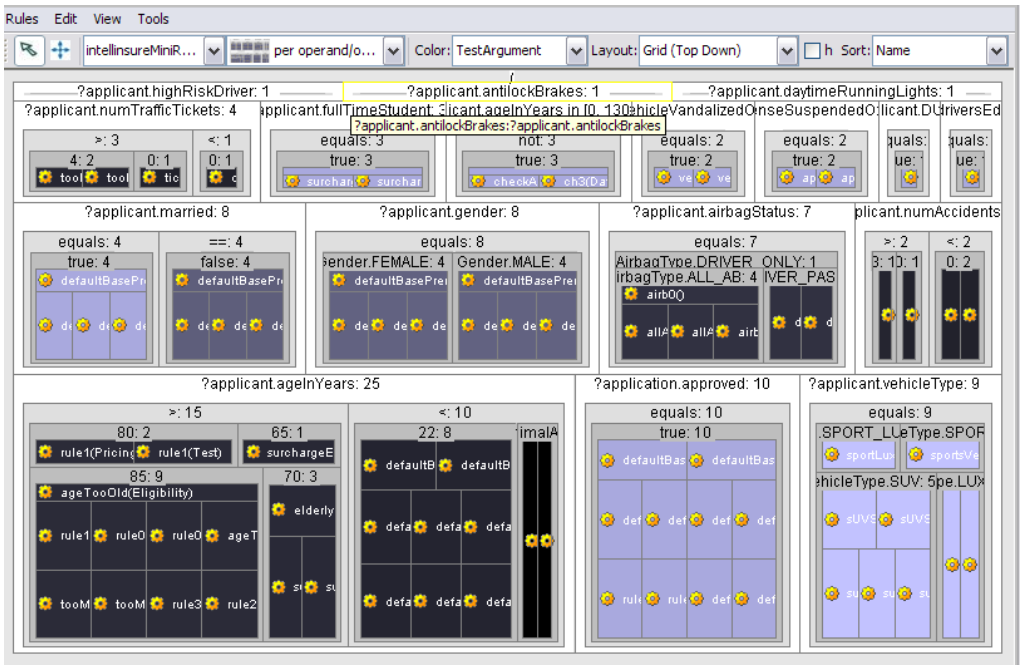
1. the attributes of the DM that are tested;
2. the operators that are used to test the attributes;
3. the values against which the attributes are tested.

This creates a view on the decision logic that helps identifying how tests for captured aspects of the input attributes of a DM are spread over the decision logic. For example, in the center of Fig. 4.2b we see the test for the gender of an applicant. The decision logic contains four tests for both females and males, which seems to suggest that there is no explicit gender-based discrimination in this decision logic. On the other hand, bottom-right we can see that there are five tests for SUV vehicles, while there are only one or two tests for the other vehicle types.

The above presented methods successfully visualize various aspects of a DM. This approach is comparable to the static program visualization approach from the software visualization domain, which consists of visualizing software programs or program analysis results. The goal of this activity is to gain understanding about the structure of a program. Analogously, the above introduced visualizations support gaining insight in the structure of a DM.



a.



b.

Figure 4.2: Visualizing the decision logic of DMS from different angles with a treemap. (a) Logic is represented using the package structure. (b) The logic is parsed to extract the tests from the rules. The tree levels are: attribute that is tested, operator and the value against which is tested.

4.1.1 Motivation

The successfulness of a treemap visualization is depending on quite some factors. First of all, there are many layout algorithms such as the original slice and dice algorithm by Schneidermann, squarified layouts and pivot layouts. Besides, various drawing aspects such as labels, margins and coloring contribute to the ease of interpretation as well. All these options present the users

with a combinatorial problem each time they want to visualize data with a treemap: how to chose which values for all of these variables? These consideration lead to the idea that a useful approach to this problem is to provide sane defaults based on the data (e.g. depth of the hierarchy and data types at each level) and possibly some guidelines from the user with respect to the kind of questions he wants to answer. Before such an approach can be developed though, more fundamental work is needed to unify the many layout algorithms. Below we present an unifying algorithm and discuss the design space it spans.

Treemaps are now over twenty years old [103] and have generated much enthusiasm in the information visualization community. Treemaps even have become a small research area of their own [141]. Therefore, the topic of treemaps or more general rectangular space-filling layout is an interesting research direction in itself. In the general public, treemaps have had their moments of fame with the map of the market [142]. They are also making their way as a standard device in the toolkit of graphic designers [143, 144]. Success stories for treemaps are the result of talented graphic design work, where the visualization designer has crafted the layout and visualization parameters to match a specific context and narrative.

We attribute the need for careful crafting to the lack of a good understanding of their design space. Choosing the right layout parameters to suit a particular dataset and features to be highlighted requires a solid experience and a dedicated presentation effort. In analysis contexts, this presentation effort is a distraction from the research task and therefore often sub-optimal layouts are used. This lack of presentation automation, or easier customizability, creates a barrier to more widespread adoption in the contexts where rectangular space-filling layouts, such as treemaps, could really bring insight.

Our goal here is to define more precisely the design space of a particular class of layout algorithms that lie at the root of the treemap concept: rectangular space-filling layouts, i.e. the layouts that tile a unit square with rectangles in a space-filling manner. We describe how input data is transformed into a set of rectangles that tile the unit square through a process that is constrained by the dimensions that span this design space. In addition we present a universal algorithm for a given class of rectangular space-filling layouts. This universal algorithm is parametrized by functors that represent the described dimensions. We present the algorithm with various examples of useful values for these dimensions, which allow creating well known as well as novel rectangular, space-filling layouts. We believe that a solid understanding of this design space can serve as a basis to develop methods and heuristics that determine the most appropriate layout given a particular dataset. Finally, it has been suggested that the design-space of space-filling rectangular layouts is very large, and that the generic problem of creating such layouts falls in the category of NP-hard problems [145]. To the contrary, we show here that:

1. Useful rectangular space-filling layouts belong to a class of limited complexity, which we call sequential space-filling layouts. They have an average complexity of $O(n)$ or $O(n\,log\,n)$, with worst case at $O(n^2)$.

2. Only five dimensions suffice to characterize the task of sequentially laying out a dataset in space-filling rectangles: order, size, chunk, recurse and phrase (the chunk and phrase dimensions echo Buxton's perspective on the structure of input [146]).

3. Functional representations (functors) of these five dimensions are the parameters of a universal algorithm for the class of sequential, rectangular, space-filling layouts.

To support those assertions, we first state the problem more formally by specifying the class of problems we address and place our work in context. Next we describe the five dimensions of the design space spanning the introduced problem class. This is followed by the construction of an universal algorithm which covers this space. Next, we show how these dimensions serve

as parameters of the algorithm and demonstrate how various values for each dimension result in different well-known or new layouts. Finally, we extend the algorithm to allow handling hierarchical data structures for the realization of a generalized treemap layout framework. This lets us conclude on the potential of our algorithm to help mastering better the design space of treemaps and rectangular space-filling layouts, as well as address more general visualization techniques with similar algorithmic space characterizations.

4.1.2 Problem statement

There are various ways to define the concept of rectangular space-filling layouts which underlie the design space of treemaps. We choose to state the problem in a more general, yet simple to formalize way. The problem we address can be stated formally as follows. A *rectangular space-filling layout* is an algorithm which

- takes as input an ordered list of N positive integers: $\{a, b, c, ...\}$, whose sum is equal to S;
- outputs a tiling of the unit square $[0, 1] \times [0, 1]$ with N non-overlapping, rectangles of surfaces $\{a/S, b/S, c/S, ...\}$. The sides of these rectangles are only allowed to be aligned with the sides of the unit square, i.e. no rotation is allowed. Colloquially, we call these "orthogonal" or "Manhattan" rectangles;
- maximizes a given objective function.

The objective function will define, for a large part, the algorithm to apply. This function is generally a weighted sum of objectives [147] involving criteria such as:

1. Aspect ratio of rectangles should be close to 1 or some chosen value.
2. Preservation of the input order.
3. Stability to resizing or adding or removing a small number of items.

Other criteria could be interesting to investigate, such as including the input surfaces as part of the objective function instead of posing them as hard constraints.

While the layout phase is the center of our attention, treemaps are often perceived as a method to visualize hierarchies. To address this perception, our generalized approach applies the solution for the above layout problem to each level of the hierarchy. This definition discards some interesting work from our focus: Ellimaps [148] or Voronoi Treemaps [149] for instance, are moved out of our scope. The former is not space-filling and both methods use non-rectangular shapes to pave the surface. Still, our proposal covers most of the central area of treemap research: slice and dice, squarified, pivot layouts. We stress, however, that our approach is not limited to treemaps but also covers related rectangular space-filling visualizations such as mosaic displays, icicle plots and 100% stacked bar charts.

A Generic Algorithm

We propose a generic algorithm, whose parameters allow specifying points (i.e. concrete rectangular, space-filling layouts) in a specific portion of the problem space described above. For convenience, in the remainder of this section we will use the following conventions. Our algorithm takes as input a collection of N tuples [150], which we will denote T. We refer to a single element in T as $t_i \in T$ with $0 <= i < N$. Each tuple t_i has a finite number of attributes, which we denote a_j for the jth attribute.

The output of our algorithm is a representation of the input data presented as a list of graphic instructions in the form $drawRectangle(t_i, x_i, y_i, w_i, h_i)$ for each i. This list is constrained: output

rectangles do not overlap and together they fill the unit square. In our algorithm description, we consider that the output is produced through a succession of calls to a rendering function: $render : T \times Rectangle \rightarrow Graphical\ Output$. Hence, our algorithm can be characterized as a function that takes a collection of tuples T and a renderer R: $draw : T \times R \rightarrow Graphical\ Output$.

Sequential methods

Considering that our target problem is defined as a combinatorial optimization problem, it would seem, at first sight, that reaching optimal solutions is a hard problem. Some have assumed that the disjunctive nature of the problem makes it NP-hard [145]. We do not adhere to this view. We will show that the constraints for tiling the full surface impose some severe restrictions on the allowed choices, and that dynamic programming-based solutions work satisfactorily.

However, it remains clear that describing the full algorithmic design space of rectangular space-filling layouts, of unbounded complexity, is a challenging task. Rather, we decide to focus on a class of greedy methods, which we call *sequential layouts*. These methods are not allowed to use backtracking techniques of unbounded depth. They can perform a fixed number of passes on the input set and partition the input set to apply to each partition element a further layout method (divide and conquer/dynamic programming approach). This class of algorithms corresponds to the class of input-linear visualizations [151, 139], augmented with partial recursion capability. As defined in [151], algorithms are input-linear (or data-linear) when there is a constant K such that, for any input set T of length N, for all $i < N$, t_i is accessed at most K times. Augmented with local recursion, the algorithms we consider are contained in the class that we call $quasi - input - linear$ algorithms, defined by the proposition that t_i is accessed $log(N) * K$ times on average and $N * K$ at most, which gives algorithms in this class a worst-case complexity of $O(N^2)$.

There are several reasons to restrict oneself to exploring this class:

- All widely known algorithms that tile rectangles belong to this class.
- If further improvements are needed, local optimization techniques provide a range of simple techniques to move down to local minima.
- A design space of this class of algorithms can be fully characterized in a simple way.

Algorithms in this class are conveniently written as a sequence of ordered passes (sequences) over the dataset, which is why we call this class "sequential algorithms".

4.1.3 Related work on rectangular layouts

In the previous section, we have restricted our ambitions to provide for a simpler model. Still the space-filling model we describe covers a large number of commonplace views which are used in various contexts.

Tables, grids, file browsers, but also simple hierarchical trees are representable in our model. For the later case, it suffice to replace the rectangle renderer with a node-link renderer to obtain commonplace trees out of a space-filling layout algorithm.

Rectangular space-filling layouts also cover a large portion of common statistical graphics. Equal height and 100% stacked bar charts are two of the simplest space-filing layout to implement. More interestingly, treemaps are predated by mosaic plots [43, 152], which are hierarchical, space-filling layouts too. Dimensional stacking [1] and pixel bar charts [153] are also covered by our design space.

Starting from the seminal paper of Johnson and Shneiderman [103], treemaps have been a widely explored area. The issue of the poor aspect ratios produced by the slice and dice technique have been addressed quite rapidly after: M. Hascoet-Zizi proposed, as soon as 1992, the

squarified layout algorithm [154]. Bruls et al. [145] rediscovered this technique and popularized it. Around the same time, Wattenberg proposed a variety of pivot-based layouts, while Bederson introduced strip and quantum layouts [155]. A recent technique worthy of discussion is spatially ordered treemaps [156]. We refer to Shneiderman [141] for a more exhaustive listing. All the techniques presented above are particular instances of our generalized layout technique.

Of interest to our work are some recent efforts to model the treemap design space more formally. Onak et al. introduced "Fat Polygonal Partitions" [157] and "Circular partitions" [158] which are interesting variants of the problem we address, splitting the unit square with quadrilaterals instead of rectangles. The extra degree of freedom gained lets them improve the optimization results, and in particular provide better overall aspect ratios. Interestingly enough, both techniques can be recreated with a generalization of our universal layout algorithm wherein the output is changed to general quadrilaterals. Schulz et al. [159] survey the design space of implicit hierarchical visualization. They define layout as one of the axes of this space. However, they only distinguish between subdivision and packing, which leaves layout still as an ill-defined black box. We show that layout can be defined more precisely and covers a design space on its own. Vliegen et. al. [160] discuss several of the dimensions we consider for our space: direction, size, nesting as a equivalent to our partitioning operator, but not our recurse functor. They also discuss other dimensions: transformation, uniform density. However, those dimensions are treated as parameters of finite range (or black-box parameters) and the algorithm features that are the foundation of the problem space are not discussed.

Our work inscribes itself in a variety of foundational work on describing the structure of graphical representations. Our approach is inspired by some seminal work in formalizing the design space of information visualization such as Chi et al. [161]. Bertin's seminal Semiology of Graphics [162] provides numerous insights, including a treemap layout (p. 270). Wilkinson's Grammar of Graphics [163] went much further by including data projection and statistical concerns in his framework. This work was extended by Wickham and Hofmann [164] who propose 1D and 2D primitives for mapping data to various space-filling and non-space-filling plots. We are not aware of related work regarding decomposing and rationalizing the layout problem. Slingsby et al. [165] proposed a general technique to configuring hierarchical space-filling layouts. Our design space is structured quite similarly, but has been inspired by the earlier work on Discovery [139]. Our contribution over the work of Slingsby et al. is that, instead of providing a preset number of layout functions, we request the specification of five functional parameters: order, size, chunk, recurse and phrase. In short, our model does not rely on a preset number of "black boxes" predefined layouts. All possible layouts in the class we consider are described through parameter settings only. Order and Size are similar to Slingsby's $sSize$ and $sOrder$ operators. Slingsby's model can probably replace its $oLayout$ operator with $oChunk$, $oRecurse$ and $oPhrase$ operators and make all those operators functional to reach the full expressiveness of our model. Recently, Schultz et al. [166] proposed a generative approach for rooted tree drawings. Their approach is similar to our in that they provide functional building blocks which are strung together in a pipeline that captures the layout process. As opposed to our model, they do not consider their building blocks as a design space which allows them to have dependencies between the building blocks. This gives a greater flexibility in the kinds of trees that can be drawn. However, it trades of the insight a design space yields for a "getting the job" done approach. Our contribution is aimed at the understanding of the design space to support further research in automatic parametrization of the layout algorithm.

4.1.4 Design Space

The design space for sequential, rectangular space-filling layouts is determined by five independent dimensions: $order$, $size$, $chunk$, $recurse$ and $phrase$. Those dimensions are functional: they

Dimension	Signature	Description
Order	$T \to T$	Order in which the data items are laid out.
Size	$T \to \mathbb{R}^+$	Determine how individual data items are sized.
Chunk	$C \times \mathbb{R}^+ \to \mathbb{B}$	Determine how data items are chunked together.
Recurse	$C \to \mathbb{B}$	Determine how a chunk lays out its content.
Phrase	$C \to (Side, Direction)$	Determine how chunks are assembled in the available space.

Table 4.1: Overview of the five dimensions that span the design space of sequential, rectangular space-filling layouts. All functors are stateful, meaning their actual signature is $X \times state \to Y \times state$.

are functors that define choice points to be taken based either on the input data or on the current state of the layout process. For example, a particular instance of the $chunk$ functor can make the decision to start a new chunk every tenth item. All functors are stateful, meaning that their actual signature is $X \times State \to Y \times State$ A brief overview of those functors is given in Table 4.1.

Order

The data is handled sequentially. Therefore, the first dimension that determines the layout is the order in which to process input. The functor that defines this dimension takes as input the original data items to be laid out and return the items in a particular order. Basic ordering functions include sorting data items ascending or descending based on a particular input attribute a_i. More elaborate ordering functions can be used: for instance, returning all items at even indices followed by items at odd indices. Functionally this dimension is represented as a permutation, or:

$$order : T \to T$$

We must take in account an important restriction: because we cover *only* the class of sequential algorithms, only ordering functions that are sequential can be used in our model. This includes linear time sort algorithms such as bucket sort and divide and conquer sort algorithms such as quick-sort.

Size

We have defined the layout problem as the tiling of a unit square, given a list of sizes. The size determines the relative space a given data item will take in the final layout. Retrieving these sizes from the original data can be expressed as a function on the input data. The most trivial functions return either a fixed size or use a particular numeric data attribute. Other possibilities include calculations on attributes of the data items or transforming categorical data attributes to a numeric value that serves as size based. Functionally this dimension is represented as:

$$size : T \to \mathbb{R}^+$$

Here, $size(T[i])$ returns the size of the ith tuple (i.e. L[i], from the initial problem statement).

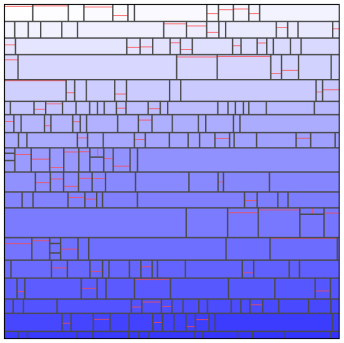


Figure 4.3: A strip treemap with recursion enabled. Elongated items that would have been placed horizontal are now placed vertically.

Chunk

A $chunk$ is a rectangle in the unit square that fully contains one or more of the output items. In the remainder, C denotes the list of chunks that pave the unit square and c_i the ith chunk, with $0 < i <= N$. When $i \equiv N$, all items are laid out in separate chunks.

A consequence of the limitation that the algorithm must be sequential is that items must be placed sequentially in successive chunks. How big these chunks are, i.e. how many items are laid out at once, can be determined in various ways. Therefore, $Chunking$ defines yet another dimension of our design space. The decision of adding an item to the current chunk or to the next can be based on various variables such as the number of items in a block with respect to the total number of items to layout or the aspect ratio of items in a chunk. Functionally this dimension is represented as:

$$chunk : C \times \mathbb{R}^+ \rightarrow \mathbb{B}$$

Here, $chunk(c[j], L[i])$ returns false when $L[i]$ should be added to the jth chunk and true when a new chunk should be formed. Recall that C denotes the list of chunks that pave unit square, thus when this method returns true, a new chunk is added to this list.

Recurse

The recurse dimension indicates whether, after having isolated a chunk, the algorithm should recurse into the chunk, reapplying itself to further improve the aspect ratio or other optimization goals. This is a partial recursion and it is bounded by the number of data items to be laid out. This functor therefore causes the algorithm's worst case complexity to grow into $O(N^2)$, just under the same kind of scenario as the quicksort algorithm: the worst case occurs when $N-1$ items are laid out in a chunk and in the recursion proceeds $N-2$ times, until finally no items are left to process. This results in $N(N+1)/2 = O(N^2)$ steps.

This dimension allows implementing the various types of pivot layouts. Recursion can be used for instance to improve the aspect ratio of small items in a strip treemap without compromising the ordering (Fig. 4.3). Functionally this dimension is represented as:

$$recurse : C \rightarrow \mathbb{B}$$

Where, $recurse(C[j])$ returns true when the algorithm must recurse into the jth chunk and apply itself recursively to the items in this chunk and false otherwise.

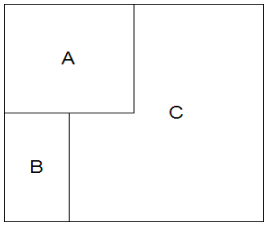


Figure 4.4: When it is decided to place chunk A of size 1 in this square of surface 4, and if there are only two items left to fit in the square which sizes are 0.5 (B) and 2.5 (C), then there is no way to produce a rectangular space-filling tiling of the square.

4.1.5 Phrase

When a chunk is finished it must be laid out in the available space. Once a chunk is laid out, no changes to the aspect ratio or to the location of the chunk can be made (because of fixed depth backtracking). Consequently, the way a chunk can be located in the available space rectangle is strongly constrained, as explained in Fig. 4.4: if a chunk was allowed to be placed anywhere in the available space, using any aspect ratio, it would be easy to create input sequences that defeat the space filling constraint and therefore the sequential constraint. Hence, there are only four possible locations for a new chunk: the four sides of the containing rectangle. Besides newly created chunks *must* take either full height (resp. width) and grow horizontally (resp. vertically) as items are added. Finally, items can be stacked in various directions inside a chunk. In a horizontal chunk, items can be either stacked from left to right or vice versa. In a vertical chunk, items can be stacked from top to bottom or vice versa. This brings the total number of possible block configurations to eight. For completeness we note that items could be stacked vertical in horizontal chunks and stacked horizontal in vertical chunks, doubling the number of configurations. Functionally this dimension is thus represented as:

$$phrase : C \rightarrow (Side, Direction)$$

Here, $Side \in \{North, South, East, West\}$ and $Direction \in \{Up, Down, Left, Right\}$. The function call $phrase(C[j])$ returns the layout configuration for the jth chunk.

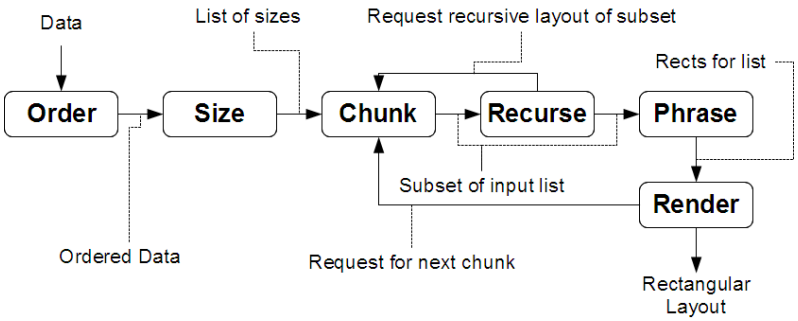


Figure 4.5: Functional view of the algorithm, showing dependencies among components.

4.1.6 Algorithm

Now that we have specified the dimensions that span the design space of rectangular, sequential space-filling layouts, we can stitch them together in order to construct a universal algorithm that

creates rectangular space-filling layouts in a sequential manner. Fig. 4.5 summarizes the steps required for creating rectangular space-filling layouts. It gives a functional view of the algorithm and shows the various functors that transform data into rectangles.

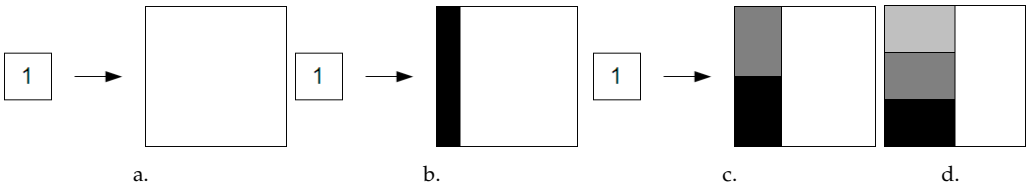


Figure 4.6: Stacking items in a chunk.

Chunks

Our algorithm stacks items in a chunk as shown in Fig. 4.6. The chunk is placed on the left side of the unit square, and three out of six equally sized items are added to the chunk. Items are stacked from bottom to top. The order of stacking is depicted by color, going from black for the first to light gray for the last item. In Fig. 4.6a, no item is added, hence the chunk height is equal to the available space plane and its width is zero. Next, in Fig. 4.6b one item is added and the chunk now has a width proportional to the size of the item that was added previously over the total size to be laid out. In Fig. 4.6c a second item is added. The width of the chunk again grows proportionally, but the heights of the items are reduced as they are stacked in the chunk. Finally, Fig. 4.6d shows the result of adding yet another item to the chunk.

A chunk has six properties: *side*, *direction*, *fromX*, *toX*, *fromY* and *toY*. These are initialized using an initial chunk configuration (combination of side and direction), and the available space rectangle, but with a flat rectangle along the progress direction. Once no more items are to be added to a chunk, it can reduce the available space rectangle by moving one of its corresponding borders proportionally to the chunk's surface. Assuming that Fig. 4.6d shows the final state of the chunk, the available space would be reduced to $[x : 0.5, y : 0, w : 0.5, h : 1]$. This progressive reduction of the available space is materialized by the $reduce$ function in our algorithm.

```
function chunk(Array T)
  // size and score are global function pointers
  Chunk currentChunk = new Chunk()
  Chunk chunks[] = [currentChunk]
  float prevScore = -inf
  for (var i = 0; i < N; ++i)
    float curScore = score(currentChunk, size(T[i]))
    if (curScore < prevScore)
      currentChunk = new Chunk()
      chunks.append(currentChunk)
      prevScore = score(currentChunk, size(T[i]))
    else
      prevScore = curScore

    currentChunk.add(T[i])

  return chunks;
```

Listing 4.1: The chunking algorithm

Chunking

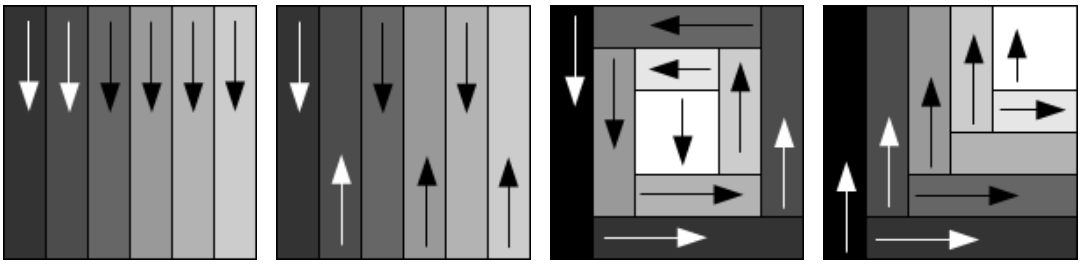
Chunking is the process of deciding whether t_i gets added to c_j or c_{j+1}. This decision is based on a simple scoring function which takes a chunk c_j, the size of tuple t_i and returns a score s. Or more formally:

$$score : C \times \mathbb{R}^+ \rightarrow \mathbb{R}$$

Here, $score(c_j,\ t_i) >= score(c_j,\ t_{i+1})$ when adding item t_{i+1} is considered to be an improvement of the layout. Like the main functors of our algorithm, $score$ is stateful as well. The function describes an optimization function whose successive local maxima are used as chunk delimiters as the algorithm progresses through the input. This chunking process is described as pseudo code in Listing 4.1.

Phrasing

When a new chunk is started, it must be decided how the chunk will be laid out in the available space. Recall that a chunk must be placed along one of the four sides of the available space, and that a stack direction must be given as well. Those two characteristics, taken together, form a chunk configuration. $Phrasing$ is the process of picking a configuration for each successive chunk. Phrase is a functor that takes the previous chunk c_{i-1} and returns the chunk configuration for chunk c_i. That is, $phrase(chunk_j) \rightarrow (side, direction)$. The first chunk is phrased according to the initial configuration set given to the algorithm, which is one of the above mentioned 16 possibilities. Depending on the strategy, the next configuration can be determined in a number of ways. There are four simple and useful strategies, which we call data independent. An overview of these strategies is shown in Fig. 4.7. Color depicts order of placement from first (dark) to last (light) and the arrow depicts the direction in which the items are placed in the chunk. These strategies work as follows:



a. Strip b. Zigzag c. Spiral d. Spikes

Figure 4.7: Four data independent phrasing strategies to create a space filling layout.

- Strip (Fig. 4.7a) - Using this strategy, each chunk is put the same way in the available space. This results in each first item of c_i being placed next to the first item of c_{i-1} for $i > 0$, i.e. a discontinuous placement of items.
- Zigzag (more accurately, Boustrophedon[1]) (Fig. 4.7b) - Is similar to strip except that the stack order of the items is reversed for every new chunk. Each first item of c_i is placed next to the last item of c_{i-1} for $i > 0$, i.e a continuous placement of items.

[1]This term comes from Greek boustrophēdon. It means ox-turning, referring to the turning of oxes when plowing a field.

- Spiral (Fig. 4.7c) - In this strategy, chunks are laid out against the next border of the available space based on the border and layout direction of the previous chunk. Each first item of c_i is placed next to the last item of c_{i-1} for $i > 0$, i.e a continuous placement of items.

- Spikes (Fig. 4.7d) - Chunks are laid out perpendicularly to the previous chunk so that the first item of c_i is close to the first item of c_{i-1} for $i > 0$, i.e. a discontinuous placement of items.

Layout

All the basic components of our layout algorithm have now been introduced. The actual layout algorithm can be described by extending the earlier presented *chunk* function as follows. The first chunk is given an initial configuration. Next we start chunking as detailed before. However, before a new chunk is started, we first test if recursive layout is required and apply the algorithm recursively if so. Next, the phrase functor is used to determine the configuration for the new chunk. The resulting pseudo code is listed in Listing 4.2.

```
function layout(Array T, int from, int to)
  Rect availableSpace = new Rect(0,0,1,1)
  Chunk currentChunk = new Chunk(phrase(null), availableSpace)
  int currentFrom = from
  Chunk result[] = [currentChunk]
  T = order(T, from, to)
  float prevScore = -inf

  for (i = from; i < to; ++i)
    float itemSize = size(T[i])
    float curScore = score(currentChunk, itemSize)
    if (curScore < prevScore)
      currentChunk.reduce(availableSpace)
      if (recurse(currentChunk))
        Chunk recursiveChunks[] = layout(T, currentFrom, to)
        if (!recursiveChunks.isEmpty())
          result.pop()
          result.append(recursiveChunks)
      currentChunk = new Chunk(phrase(currentChunk), availableSpace)
      result.append(currentChunk)
      currentFrom = i
      prevScore = score(currentChunk, itemSize)
    else
      prevScore = curScore
    currentChunk.addItem(itemSize)

  if (currentFrom != from && recurse(currentChunk))
    Chunk recursiveChunks[] = layout(T, currentFrom, to)
    if (!recursiveChunks.isEmpty())
      result.pop() // replace last chunk with subdivision result.
      result.append(recursiveChunks)

  return result
```

Listing 4.2: Basic layout algorithm

Finally, we can implement the draw function mentioned in the input and output section as in Listing 4.3. The *chunk.rectangle* method simply assigns iteratively a rectangle to each item in the chunk based on the chunk's rectangle and its *direction* attribute and increments a local counter to assign the border of the next item to be drawn.

```
function draw(T, R)
  Chunk chunks[] = layout(T, 0, T.length())
  foreach (chunk : chunks)
    R.drawRect(chunk.rectangle())
    foreach (t : chunk) { // item in chunk
      R.drawRect(t, chunk.rectangle(t))
```

Listing 4.3: Draw function implementation

This fully describes our universal algorithm that solves the problem stated in Sec. 4.1.2 with sequential methods. Note that some parts of the algorithm are optimized in practice. For example, the *ordering* and *sum* functors are memoized, instead of being recalculated for each recursive iteration. These kind of optimizations are left out for clarity.

Completeness

As we have mentioned in the introduction, the dimensions we describe are independent, and each covers a functional, recursively enumerable, domain. We can show in a nonconstructive way that our generic algorithm allows the depiction of *any* sequential, rectangular, space-filling layout as defined in Sec. 4.1.2: The *order* functor is instantiated with a general function that can perform *any* calculation and store its result in the state variable. The *chunk*, *phrase* and *recurse* functors are then free to reuse those results as they see fit. Hence, it is always possible to "pack" a knowingly sequential algorithm into the *order* functor, which has the adequate signature, and leave the other functors just retrieve precomputed values to perform the layout. Finally, as long as the *order* function stays sequential in the sense of Sec. 4.1.2 and the other functions stay in constant time, all possible parametrizations of our generic algorithm stay sequential.

While this sketched analysis somewhat downplays the use of separate *phrase* and *chunk* functors, these two dimensions greatly simplify the expression of a large variety of rectangular space-filling layouts. *Phrase* and *chunk* allow expressing in a terse way the base ingredients of a divide and conquer approach to layout. The divide and conquer approach is a sensible technique to encompass efficiently the various objective functions of the problem stated in Sec. 4.1.2.

Indeed, the combination of *chunk* and *phrase* functors capture the fact that the chunking process needs to proceed by growing chunks along the borders of the area to fill. If a sequential layout algorithm decided to attribute to a chunk a rectangle that is *not* on the border of the available area, and does *not* extend to the full height or width of this area, then it suffices to input the algorithm two (or more) additional items which cannot be made to fit the remaining space: the algorithm fails to find a proper tiling. Fig. 4.4 illustrates this fact. In other words, an algorithm that wishes to chunk some items together into a distinctive block to further layout this block independently *must* proceed by placing the distinctive block along one of the edges of the area to layout, *or* be allowed to perform backtracks of arbitrary depth to decide how to organize those blocks together in the alloted space. These backtracks of arbitrary depth push the complexity of the method beyond the realm of input-linear and quasi-input-linear methods.

These considerations let us claim that our algorithm is universal for the class of algorithms considered, i.e. sequential methods for tiling the unit square with rectangles of varying surfaces. We acknowledge that a detailed proof for this claim is desirable. However, such a proof goes beyond the scope of this thesis as it would require further formalization and introduction to the

class of input-linear algorithms. Instead, we now show how in practice those five dimensions can be combined to provide a wide variety of layouts.

4.1.7 Layout parameters

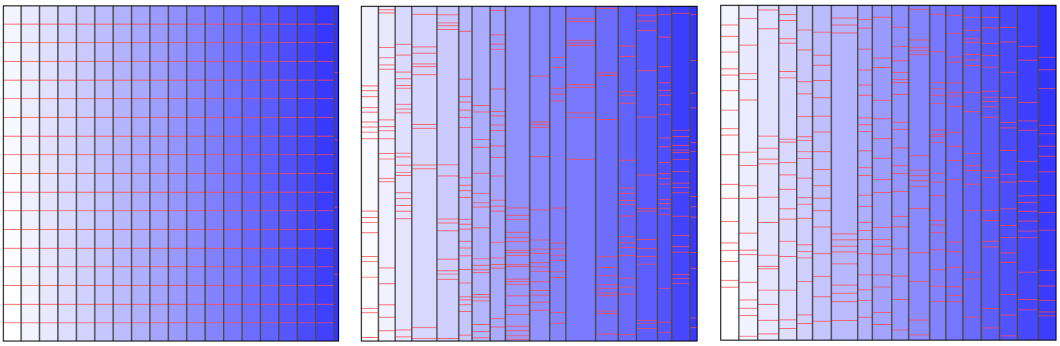
This section presents various examples of functions for each of the considered dimension and demonstrates how different configurations of the algorithm create instances of well known or new layouts. First, we briefly discuss the $order$ and $size$ functions as these are rather trivial. Next, we illustrate how the $score$ and $phrase$ functions interplay with each other to allow producing a variety of known or novel layouts. Finally, we discuss the recursion functor which adds the ability to reenter the layout algorithm inside chunks. In the following we use a dataset containing US cities, the states they belong to and measures for various properties such as population, climate, education and health care. Individual items are colored by item index from light (low index) to dark (high index) and separated by red lines. Chunks are separated by black lines.

Order and Size

The $order$ functor orders (a subset of) the original input dataset. When this functor is not specified the original order of the data is kept. A trivial example of an ordering function is one which sorts the tuples based on a particular attribute of the dataset, e.g. by population. However, we explicitly named this dimension $order$ because more complex ordering methods, which fall outside the range of sorting, are allowed. For example, we chose to access first the tuples that have an even index, followed by the tuples that have an odd index.

The $size$ functor lets one specify which attribute or function to use to compute the size of each tuple. There are only a limited number of useful $size$ functions:

- Constant (Fig. 4.8a). One for tuples and nodes alike. This is useful for similar usages in grid layouts.
- A function that returns a tuple attribute (Fig. 4.8b) which should be a ratio attribute for best result. For nodes in a hierarchy, the value of this function is the sum of all values of the tuples it contains. An efficient implementation will of course cache this sum for each node to avoid recomputing it needlessly.
- The result of a computation on tuple attributes (Fig. 4.8c). This is merely a special case of the former where a value is calculated based on one or more fields of the available data.



a. Constant b. Population c. Education over HealthCare

Figure 4.8: Various settings for the size functor.

Chunk scoring

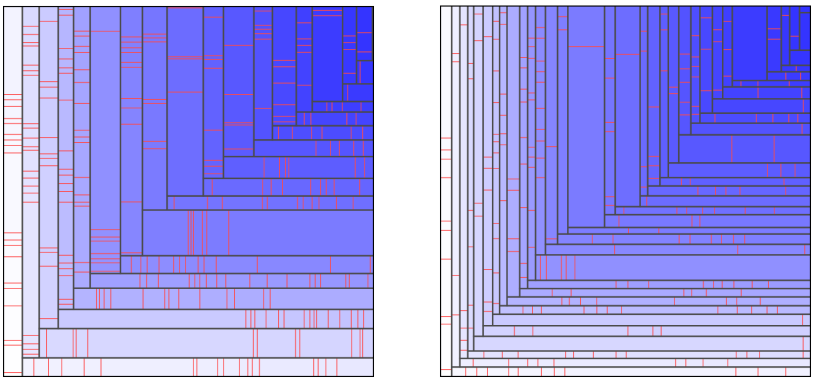
The Chunk scoring function is the most powerful and complex parameter. The *score* functor evaluates a score for a chunk composed of a current chunk and one extra item. So, this *score* functor is evaluated *before* adding an item to the chunk. In the remainder of this section we assume a global variable state is available for bookkeeping, which is updated by the algorithm as required.

The simplest instances of this functor are *Slice* and *Dice*, which return the maximal score when the chunk has only one item, resp. when its cardinality is the full node. When these methods are alternatively applied at each level of a hierarchical data structure, this produces the regular slice and dice layout. Equally simple is the *Grid* function. This function returns its maximum when the number of items is equal to the closest approximation of the square root of the number of children in the node. This results (approximately) in a number of chunks with equal cardinality. To produce a perfect grid, the *size* functor needs to be adjusted to return the smallest square integer above the cardinal of the input size. As its name indicates, this chunking method produces grids (Fig. 4.8a) if the size attribute is set to constant, or some variant of a strip layout if not.

```
function Slice(Chunk c, double itemSize, State state)
  return c.itemCount == state.itemCount ? 1 : 0

function Dice(Chunk c, double itemSize, State state)
  return 1 - c.itemCount

function Grid(Chunk c, double itemSize, State state)
  return square(c.itemCount) <= state.itemCount ? 1 : 0
```

Listing 4.4: Elementary chunking functions

More interesting, yet complex, chunking functions can be used, allowing for instance to implement a variety of squarified and strip layouts. The *bestAverageAspectRatio* (Listing 4.5, Fig. 4.9a) and *bestMinAspectRatio* (Listing 4.5, Fig. 4.9b) functions return as a score respectively the average aspect ratio of all the items in the chunk and the aspect ratio of the smallest item in the chunk. The closer this value is to one, the more "fit" the current chunk is to be laid out. To



a. Best Average b. Best Minimum

Figure 4.9: Two chunk scoring functions based on the aspect ratio of items: (a) best average aspect ratio. (b) best aspect ratio for smallest item. The *order* function is not specified, resulting in a (intentional) poor squarification of the layout.

compute these values, the width (resp. height) of the chunk has to be computed by evaluating the ratio of the sum of the current chunk over the total sum of the square to be laid out. The height (resp. width) is equal to that of the enclosing space. Having computed the width and height of the chunk makes computing the average and minimal aspect ratios of its content elementary.

```
function BestAverageAspectRatio(Chunk c, double itemSize, State state)
  Rect rect = state.availableSpace
  float expandingDim = c.isVertical ? rect.height : rect.width
  float fixedDim = c.isVertical ? rect.width : rect.height
  int newChunkSize = c.sum() + itemSize
  float aspectRatio = expandingDim / (c.itemCount + 1)
    / (fixedDim * newChunkSize / state.overallSize);
  return aspectRatio > 1 ? 1 / aspectRatio : aspectRatio

function BestMinAspectRatio(Chunk c, double itemSize, State state)
  Rect rect = state.availableSpace
  float expandingDim = c.isVertical ? rect.height : rect.width
  float fixedDim = c.isVertical ? rect.width : rect.height
  int newChunkSize = c.sum() + itemSize
  int minItemSize = (c.minItemSize() < itemSize)
    ? c.minItemSize() : itemSize
  float aspectRatio = expandingDim * (minItemSize / newChunkSize)
    / (fixedDim * newChunkSize / state.overallSum)
  return aspectRatio > 1 ? 1 / aspectRatio : aspectRatio
```

Listing 4.5: Calculating score based on average and minimum aspect ratio

Finally, to implement Pivot layout [155], other *score* functions can be used that split the input data in two approximately equal parts (Listing 4.6). Pivot by middle has its maximum around the middle of the item count, Pivot by size has its maximum at the index of the biggest item, and Pivot by split size has its maximum where the sum of the sizes is closest to half the total sum.

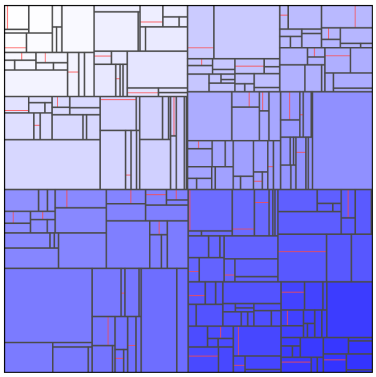


Figure 4.10: A pivot treemap created using a pivot chunk scoring function and recursion.

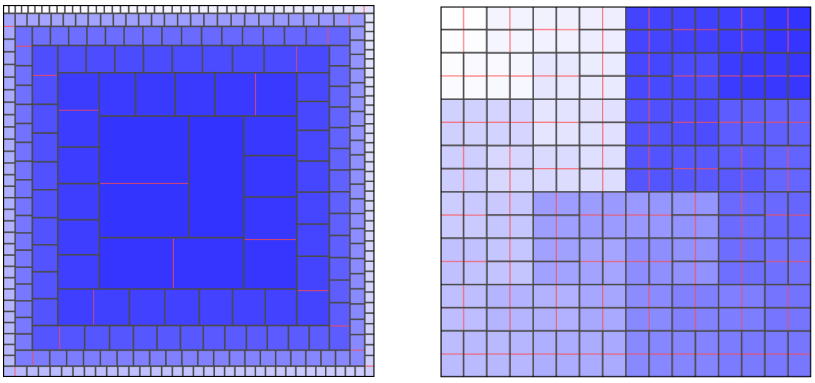
```
function PivotByMiddle(Chunk c, double itemSize, State state)
  return -square(c.itemCount + 1
                 - (state.to - state.from) / 2)

function PivotBySplitSize(Chunk c, double itemSize, State state)
  return -square(state.currentChunk.size() + itemSize
                 - state.remainingSum / 2)

function PivotBySize(Chunk c, double itemSize, State state)
  return state.currentIndex == state.indexOfBiggestItem ? 1 : 0
```

Listing 4.6: Pivot layout functions: The first two are parabolas which have their maxima at the desired item index (Fig. 4.10).

Phrase

In Sec. 4.1.6 we introduced four simple, data independent phrasing strategies. The implementation of these strategies is done by means of functions which basically consist of a switch statement over the current chunk phrase configuration to pick the next one. A novel layout is shown in Fig. 4.11a, where we apply spiral phrasing to our US population dataset. More complex phrasing functions, combined with recursion, can use some state variables and a stack to produce sophisticated layouts such as the Peano-Hilbert curve of Fig. 4.11b.

In addition, phrasing can be made dependent on the input data, to produce data dependent strategies. Those can echo the data independent strategies described earlier. We call the data dependent alternative of Strip phrasing *Worst Discontinuous* phrasing. In this strategy, each chunk is placed so as to degrade the aspect ratio of the available space. Items in new chunks are stacked in the same direction, resulting in a discontinuity at each chunking step. The data dependent variant of zigzag is what we call the *Worst Continuous* strategy which places the next chunk so as to degrade the aspect ratio and reverse the stack direction. The data dependent variant of Spiral is what we call *Best Continuous*. In this strategy the chunk is placed so as to improve the aspect ratio of the available space and orders the items so that the first item of the next chunk is next to the last item of the previous chunk. Finally, we call the data dependent variant of Spikes (Fig. 4.12a) *Best Discontinuous* (Fig. 4.12b). It places each chunk so as to improve the aspect ratio of the available space and so that the first item of the new chunk is next to the first item of the



a. Spiral phrasing b. Hilbert curve phrasing

Figure 4.11: Two different ways to place chunks consecutively in a plane: following a spiral and following a hilbert curve.

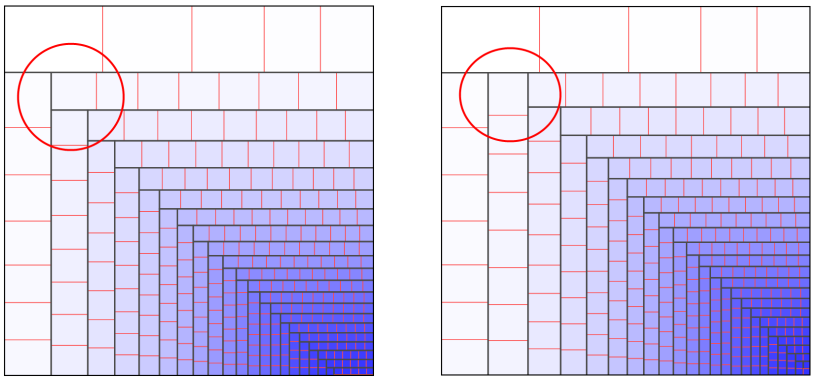
previous chunk. This layout is a novel and interesting improvement over the regular squarified layout as it results in significantly squarer items, as shown in Fig. 4.12b.

```
function hilbertPhrasing(state)
  state.sequence += 1
  var direction = state.parentConfig
  if (state.depth % 2 === 0)
    // at even depth, we split in 2 blocks of 2 and reverse
    // direction of the 2nd chunk
    if (state.sequence % 2 === 0)
      return direction
    else
      return direction.reverse()
  else
    // at odd depths we toggle the orientation of each quarter
    if (state.parentSequence % 2 === 0) // 1st half
      if (state.sequence % 2 === 0)
        return direction.alternate() // q1
      else
        return direction // q2
    else // 2nd half
      if (state.sequence % 2 === 0)
        return direction.alternate() // q3
      else
        return direction.alternate().reverse() // q4
```

Listing 4.7: .Hilbert phrasing (see Fig. 4.11b). direction, sequence and depth are state variables maintained at the scope of each chunk.

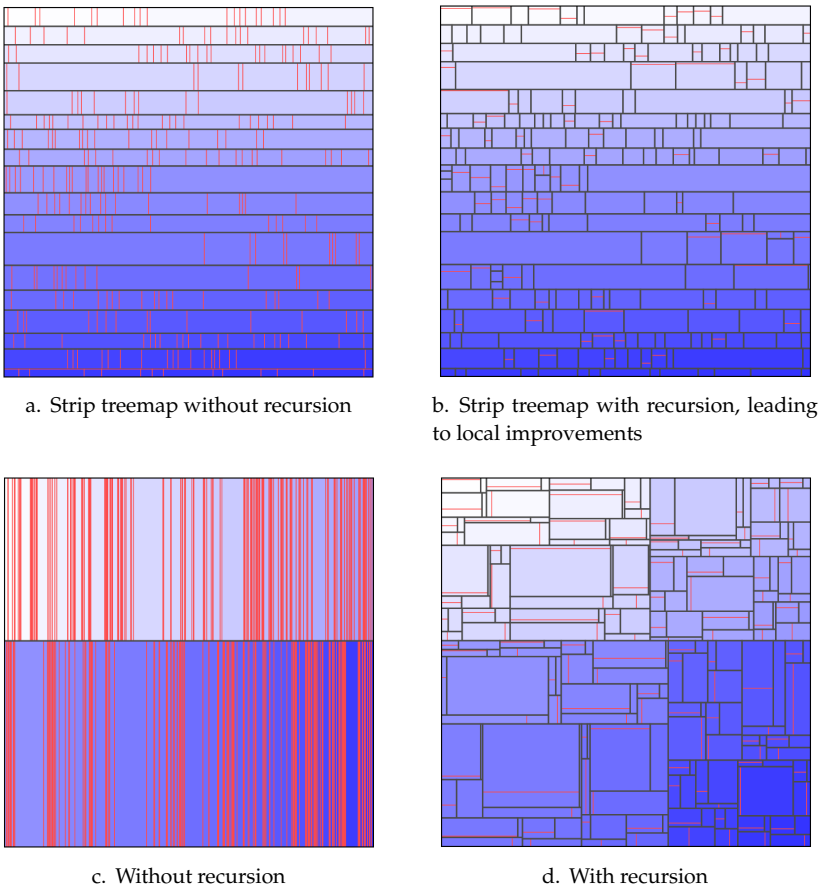
Recurse

As with the other dimensions, $recurse$ is a functor in our algorithm that returns a boolean. When true is returned the layout algorithm is applied to the items of the current chunk. If the recursion results in one or more chunks, the current chunk is removed from the result and replaced by the chunks that resulted from the recursion step. Obviously, there must be a stop criterion for the



a. Spikes b. Best Discontinuous

Figure 4.12: Data independent and dependent spike phrasing strategies. Notice the two vertical chunks at the left in (b).



a. Strip treemap without recursion

b. Strip treemap with recursion, leading to local improvements

c. Without recursion

d. With recursion

Figure 4.13: Applying recursion for layout improvements and pivot layouts.

recursion. Therefore, the simplest value of the recurse functor should not just be a function that returns true but limit itself based on the number of remaining items. In our implementation it stops when only two or less items are left to be laid out.

One reason to use recursion is that it might improve aspect ratio of small items. Fig. 4.13a and Fig. 4.13b show that after recursion many of the small items got a better aspect ratio. Another reason for recursion is to implement various variants of the well known pivot layouts. Fig. 4.13c and Fig. 4.13d show how recursion complements the pivot by middle scoring function.

4.1.8 Structuring

So far, we have focused purely on the layout aspect of space-filling visualizations. However, as stated before, treemaps are a central concern, and those are most often perceived as a method to visualize hierarchies. To accommodate this perception, we extend the presented algorithm by introducing a structuring phase. This phase consists of partitioning and ordering which reflect the $sHier$ and $sOrder$ states from Slingby et al. [165]. First we define a new structure Node, used for representing hierarchies. This allows the algorithm to take as input either a tabular input dataset and turn it into an ordered hierarchy according to some set criteria, or a preexisting hierarchy of ordered nodes. Each node has a unique identifier and keeps a list of children. The children can be tuples, nodes or a mix of both.

We adapt the previously introduced $layout$ and $draw$ functions to take this new context into account. First, the layout function now takes a node n as input and additionally does not assume

the unit square as available space any longer but takes a bounding rectangle BR. Furthermore, the children of a node can be a mix of tuples and nodes. Hence, the size function must be adapted to return the appropriate size for the ith child of n. The required changes for the layout function are listed in Listing 4.8.

```
function layout(Node n, Rect BR, int from, int to)
  Rect availableSpace = BR
  // Same as in listing 2
  int N = n.childCount()
  int S = sum(n, from, to).
  order(n, from, to)
  float prevScore = -inf
  for (int i = from; i < to; ++i)
    int itemSize = size(n.children[i])
    float curScore = score(currentChunk, itemSize)
    if (curScore < prevScore)
      // Same as in listing 2
    else
      prevScore = curScore;
    currentChunk.add(n.children[i])
  // Same as in listing 2
```

Listing 4.8: Layout function for hierarchical structures

Like the layout function, the draw function now takes a node n and a bounding rectangle BR instead of assuming unit square. The items in b are now elements e, which are either tuples or nodes. In the former case, we call draw on the Renderer as before. In the latter case, we recursively call draw, but now we set the bounding rectangle to the rectangle that was laid out for this node. All children of the node will therefore be laid out not in the unit square but in the input bounding rectangle. The adapted version of the draw function is listed in Listing 4.9.

```
function draw(Node n, Renderer R, Rect BR)
  Chunk chunks[] = layout(n, BR, 0, n.childCount())
  foreach(chunk : chunks)
    foreach(element : chunk)
      if (e instanceof Tuple)
        R.drawRect(e, b.rectangle(e))
      else // e is a Node
        // Recursively draw the next level in the tree.
        draw(n, R, chunk.rectangle(e))
```

Listing 4.9: Recursive draw function for hierarchical structures

At this stage our space-filling layout algorithm is able to deal with both flat and hierarchical data. In both cases the initial step is to convert input T into a root node containing T as children. Now an additional structuring function is added before calling draw, which converts the flat root node into an ordered hierarchy. The structure function executes in two phases, partitioning and ordering. These two phases are detailed now.

Partition

Partitioning aggregates tuples in nodes according to a user provided expression. This expression is expected to return a unique sub node identifier for each tuple, or null if the tuple is to remain

at the current level. Calling this functor on each child tuple results in creating one additional level to the hierarchy. Listing 4.10 shows how a partitioner can be used to create a hierarchical structure out of flat data.

```
function structuring(Node n, int depth)
  Array result = []
  Map partitionMap = {}
  foreach (e : n.children)
    Object id = partitionExpression(e, node, depth)
    if(id != null)
      child = partitions.get(id)
      if(child == null)
        child = new Node(id)
        structure(child, depth + 1)
        partitions.put(id, child)
        result.add(child)
      child.add(e) // Add element to node
    else
      result.add(e) // Add element as leaf
  order(result)
  n.children = result
```

Listing 4.10: Structuring

The simplest partitioning expression, $Enumeration$, returns the value of a nominal tuple attribute: $returnget(t, myPartitioningColumn)$. This partitioner allows, for instance, to group a set of US cities by states. Likewise, date and numeric partitioners can be defined to split according to values and fields held in a date or numeric attribute. A hierarchical partitioner can be defined as a list of single column partitioners, resulting in mosaic plot layouts. Hierarchical partitioners need to keep track of the depth of the current node, to invoke the proper level of partitioning. Finally, a path partitioner takes an attribute column that describes a path (directory + file name, or URL, or date in the form y/M/d h:m:s, for instance). At each depth level n, this partitioner will return the corresponding nth substring in the attribute value. This partitioner is used for instance to display a file hierarchy in a treemap.

Order

Once partitioning is done, the resulting list can be ordered. Because the children can be either nodes or tuples, the function used to determine order needs to take into account the possibility of having to compare a node against a tuple. Past this hurdle, defining a comparator is fairly simple. The most common comparator, like the $Enumeration$ partitioner, uses a column as its sort criteria. To handle nodes, this comparator defines an aggregation value for nodes, which can be, for instance, the sum of all the tuple values, or their average, maximum or minimum. Alternatively, the comparator can decide to place all nodes before or after all tuples, and sort them according to their node id, their immediate child count or the number of tuples they contain. Each of those possibilities will determine different placement of the nodes at a given level, but will not affect the layout algorithm per se. Finally, when ordering is done at the structuring phase, it should be obvious that ordering can be removed from the layout phase.

4.1.9 Conclusion

We have defined the design space of sequential, rectangular, space-filling layouts. This space is covered by five independent, functional, dimensions, namely: order, size, chunk, recurse and

phrase. In addition we presented a universal algorithm for sequential, rectangular, space-filling layout. Our method leverages the observation that optimization techniques for sequential layout methods are essentially of the divide and conquer type, requiring chunking the input into blocks that are further laid out separately. The sequential nature of the process imposes the chunking process to proceed from a full side of the available space to fill, and then proceed along one of four possible sides, along one of four possible directions. Giving the user the ability to specify the value of each of those dimensions at each step of the algorithm results in a universal method to describe this class of algorithm. Additionally we discussed an extension of the algorithm in order to make it suitable for hierarchical data as well.

The Discovery [139] framework implements the majority of the presented algorithms and techniques, even though it was not formalized at the time. To provide broader access, we have implemented the generalized space-filling layout algorithm as a small, independent component. This component consists of only a few hundreds of lines of JavaScript code and follows closely the presented algorithms.

A future research direction is to extend the presented algorithms to other layout problems involving only data-linear or quasi-data-linear visualizations. We have mentioned the possibility of describing circular partitions [158] and [157] with the same techniques. Many tree and graph drawing algorithms that do not require global heuristics (where the position of a node depends possibly on all the other node positions) are also representable with our technique, simply replacing the drawing of rectangles with drawing of arcs joining the centers of the rectangles.

Yet, let us return here to the original motivation of our work. Baudel used treemaps to visualize various static artifacts extracted from **DS1** [140] (See Fig. 4.1 and Fig. 4.2). However, he found that for a usable representation of these structures a careful selection of configuration of the treemap layout and rendering settings is required. In a business context we cannot expect users to have the required experience to find the most optimal representation. In order to leverage this problem, automatic configuration based on the tree-structure, data types and user questions would be desirable. To this extend, we defined a design space for space-filling, rectangular layouts and presented a generic algorithm which can generate all possible layouts within this design-space. Hence, our goal in defining this design space is to provide a more solid grounding to support analysts in their use of space-filling displays. We envision using this design space to develop a systematic method and a set of heuristics to (semi-) automatically choose a proper layout given some datasets, appropriate meta-data and contextual information.

Finally, our work presents some aspects that seem to be new in the area of computational geometry. As said before, we are not aware of related work regarding decomposing and rationalizing layout design spaces relying on algorithmic properties of the problem specification. Similar under-specified problems (whose objective function is a weighted sum of objectives, the weights being personal decisions) are abundant in the literature. By leveraging the cases where these problems are satisfactorily addressed by algorithms of low complexity (sequential, dynamic-programming...), we have found a way to characterize such algorithmic design spaces elegantly, through a limited set of functors that represent elementary decision functions that are to be made by the algorithm: in our context, ”how to group items together? (chunk)”, ”in what order to fill the plane? (phrase)”, ”do we refine the groups? (recurse)”. The study of input-linear and quasi-input linear algorithms in such contexts could bring new insights to these classes of problems.

4.2 Rule Execution Visualization

When decisions are executed, the DMS will among other things store the rules that where triggered for each decision (**DS3**). This rule triggering information is hierarchical in that it also contains the control flow tasks to which rules belong. Control flow tasks can be control flows them-

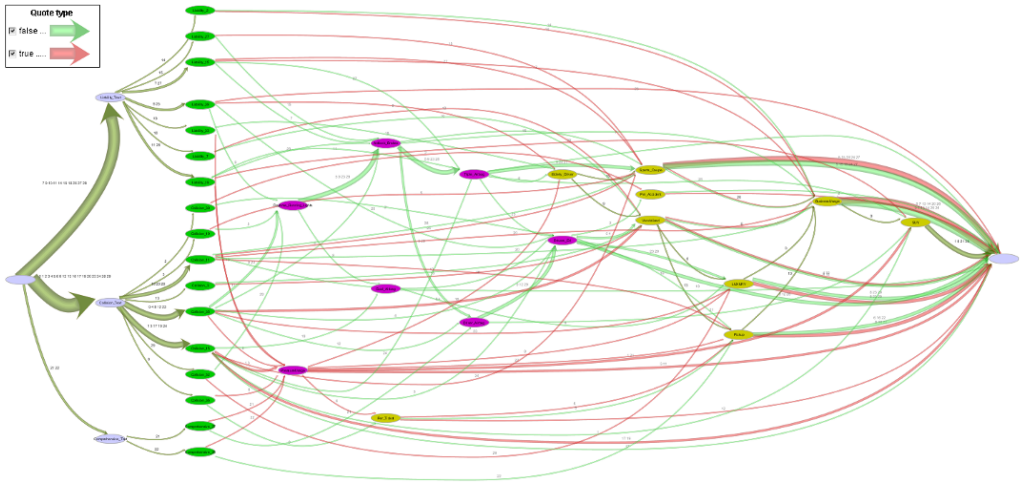


Figure 4.14: A rule trigger graph for two version of the car insurance DM.

selves, thus resulting in a hierarchical structure. Combined with the decision instances (**DS2**), detailed information can be extracted with respect to the functioning of the DM.

4.2.1 Visualizing rule execution graphs

Rule execution traces for individual decisions are typically small due to the nature of the business logic. Typically, a decision will trigger five to twenty rules. The actual patterns depend on the on the application domain. For example, a for quoting application each decision triggers this amount of rules, while in a fraud detection application most decisions trigger no rules at all while some decisions (the fraudulent) result in a cascade of rules being triggered. Two common tasks performed by the business logic are segmentation and dealing with exceptions. For segmentation there may be tens or even hundreds of rules, typically structured in a decision table, to divide inputs in several segments. However, each input will be put into only one segment, hence only one of these many rules will be triggered for a given input. Dealing with exceptions means that those cases are not expected to happen often, thus the majority of times this kind of rules will not be triggered.

Unlike in program comprehension, in the context of DMS a single trace is only of interest if the resulting decision is problematic. More interesting are the aggregated results of many decisions. To this extent, Baudel constructed a structure that is comparable to a call graph [140]. Each rule becomes a node in this graph and two rules A and B are connected with a directed link when B is triggered after A. The thickness of an edge represents the number of decisions that triggered the two rules connected by the edge. Note, that there is a semantic difference here with a call graph, where the same link would represent method A calling method B.

Fig. 4.14 shows the result of an early prototype by Baudel [140]. After extracting the rule trigger graph from the execution traces he colored the nodes based on their package names:

- light blue nodes left: separation based on insurance type
- green nodes: base quote decision table
- purple nodes: discounts
- yellow nodes: surcharges
- light blue node right: final quote calculation

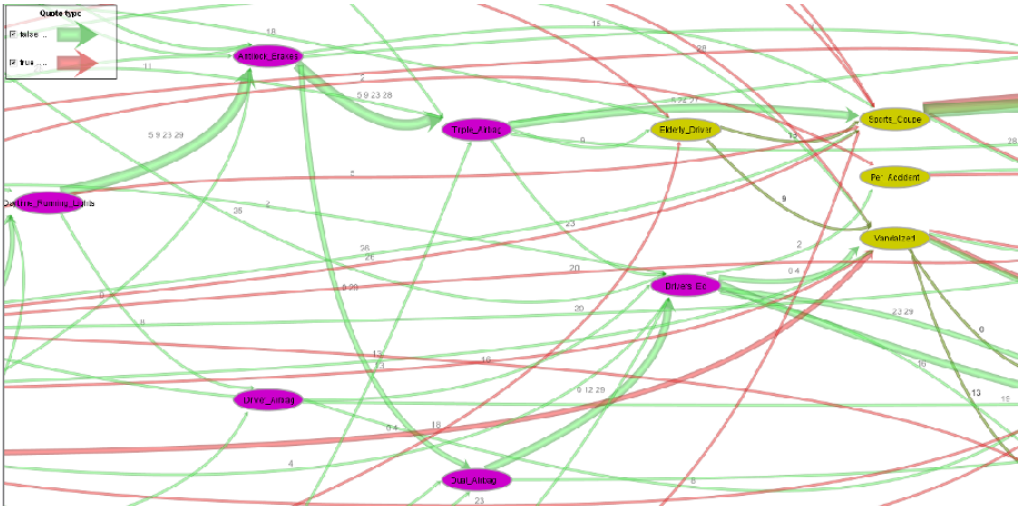


Figure 4.15: Detail of the rule trigger graph for the car insurance DM, showing how various discounts are triggered together.

This graph has two purposes: the first is to find trigger patterns that look suspicious or give hints for improvements, the second is to compare scenarios. The green edges in Fig. 4.15 reveal how the daytime running lights discount and the anti-lock brakes discount almost always come with a triple airbag or dual airbag discount. Even though all these option individually might contribute to a lower chance of getting involved in accidents, business wise it is questionable if it justifies the summed discount.

As a result of this insight a business analyst can decide to adapt the DM to adjust its functioning. For example, he decided to introduce one discount for cars in the price range that likely have one or more of the above mentioned security features. After this change he would like to verify the new functioning. This is shown in Fig. 4.14, where the green edges represent the functioning of the first version of the DM and the red edges the new version. It is clear that with the version of the DM, represented by the red edges, this combined triggering of discounts does not happen any more.

However, as pointed out by various authors [167, 168, 169], scalability issues inevitably arise. Execution traces of software systems can easily require hundreds of megabytes or even several gigabytes of storage space. In the context of DMSs a similar problem arises. With a growing number of decisions it becomes more likely that exception rules are being triggered. For example, the graph shown in Fig. 4.16 is the result of 1000 executions. In this graph we can still see some coarse grained patterns such as the three types of decision (basic, basic + personal, full coverage) and the rejections at the left side. However, detecting more fine grained patterns in this graph is already very hard let alone comparing different scenarios. Yet, in practice the number of decisions can easily go into the millions making this approach unusable. Additionally, this graph is a hierarchical graph. Laying out such a graph is a NP-complete problem and consequently takes a long time to render, making it unsuitable for interactive analysis.

4.2.2 Conclusion

The graph survey by von Landesberger et al. [170] suggests that there are more suitable representations for the rule trigger graph. However, there is a deeper reason that should be taken in consideration to determine if this is a fruitful direction or not. In Sec. 2.4 we explained that the order in which rules are triggered is largely a side effect of the execution engine. Therefore, pre-

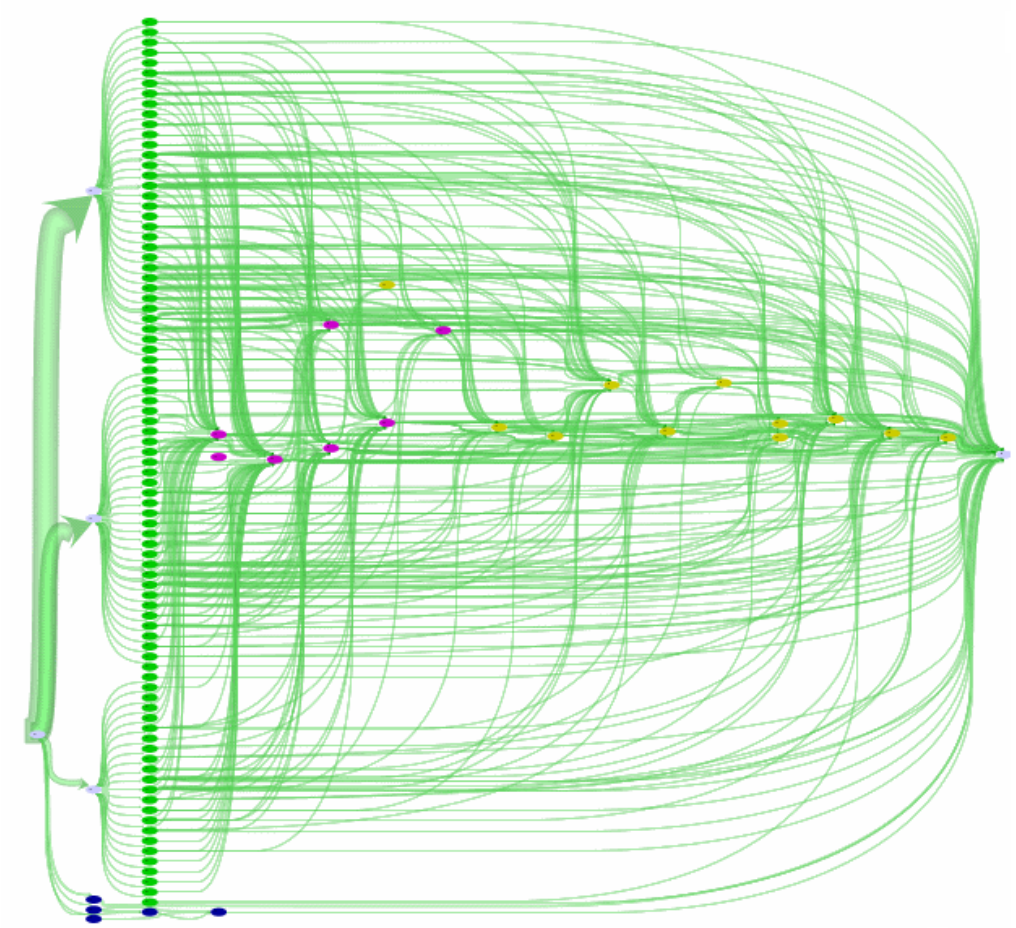


Figure 4.16: With 1000 decisions being taken, more different rules are triggered resulting in a hard to read rule trigger graph.

senting the rule trigger graph to a business user could become the source of subtle interpretation problems.

First of all the resulting layout can present the decision flow in a way that does not match the mental model that the analyst has. His mental model of a DM is reflected in representations such as decision tables and rule control flows (see Fig. 2.2 and Fig. 2.4). Those artifacts are actually constructed by the business analysts themselves. Additionally, while decision flow charts have a notion of rule execution order in them, it is not said that the actual rule execution as determined by the execution engine will exactly match this order. The explicit rule control flows might therefore be a better starting point than actual rule execution graphs for showing aggregated effects of input instances flowing through the decision model.

A second reason is that this representation forces the analyst to segment the input by groups of triggered rules as opposed to segmenting inputs by properties of these inputs. That is, the typical reasoning that is aimed at is: by observing that rule X, Y and Z are triggered together the analyst deduces that the inputs flowing through these rules have certain characteristics. In the above example, the consecutive triggering of the anti-lock brakes, daytime running lights and airbag discount rules led to the conclusions that these deal with expensive cars. Clearly, this kind of reasoning only works if the rules are simple enough, which should not be assumed.

4.3 Change Impact

The DMs of decisions which are automated with DMSs evolve over time. To adapt to changing markets, law and business policies, DMs are modified. These changes can involve both the rules and the domain model. In the case of rules, new rules are added and existing rules are modified or removed. The domain model is extended with new concepts and attributes, while existing concepts and attributes are modified or removed.

4.3.1 Visualizing domain model change impact

Chniti et al., present an approach to model the changes that can be made to the domain model, such as structural changes, conceptual changes or entity definition changes [171]. They focus on how each particular kind of change will lead to potential inconsistencies in the decision logic authored over the changing domain model. For example, when an entity or attribute is renamed, all rules that refer to the old names have become invalid.

The approach of Chniti et al. also contains repair rules for problems or inconsistencies that arise as consequence of a proposed change. It is not clear though, if all detected inconsistencies can be repaired automatically. Additionally, the examples discussed in their work are typically small, i.e. the approach is only tested with a couple of changes on small DMs. In reality, both the number of changes and the size of the DM are much larger. Therefore, prior to performing the changes, it might be useful to visually inspect the impact of the modeled changes.

We extended the prototype of Chniti et al. [171] as follows. Their prototype already allowed for modeling changes and it parses the decision logic in order to find the impacted business rules. We added export fucntionality to the prototype in order to export both the hierarchical structure of the domain model (packages, concepts, attributes) and the hierarchical structure of the decision logic (packages, business rules). Additionally, for each changed artifact of the domain model a link was exported to the impacted rule. For quick prototyping purposes we exported this compound tree structure in an XML format suitable for SolidSX [172], a visual analysis tool for code structure, dependencies and metrics.

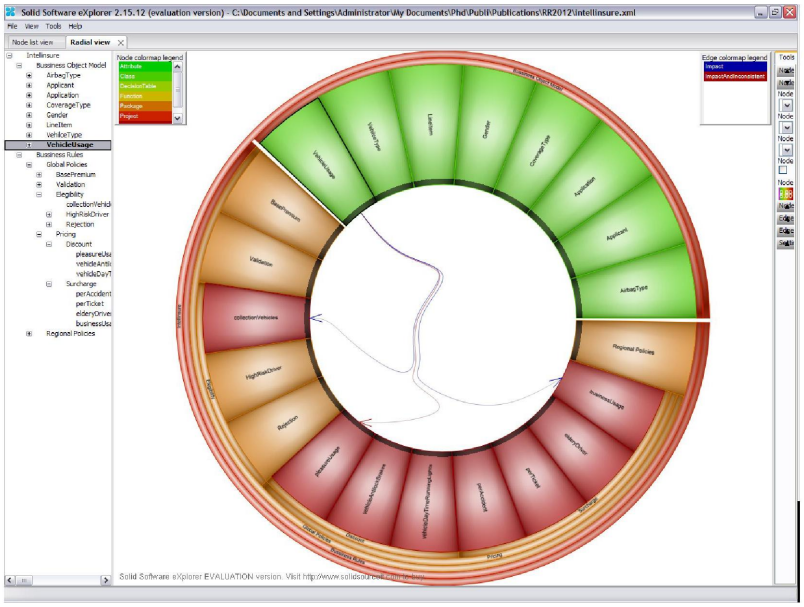


Figure 4.17: Visualizing the impact on the decision logic of changing the $VehicleUsage$ concept.

Fig. 4.17 shows how the impact on the business logic can be visualized using this well-known approach from the software comprehension domain. The green nodes in the circular view represent the concepts of the domain model. Orange nodes represent packages and the red nodes represent business rules. Business domain concept nodes and rule nodes are connected with a blue link when the rule is impacted by a change to the concept.

4.3.2 Related applications

We addressed a similar problem in the domain of software maintenance. In [173] we describe a system to estimate porting efforts for projects that need to port a C++ code base from one version of a dependency to a newer version of that dependency. This system allowed a developer to perform queries on a code base to find the usage of an API. Based on these queries we visualized porting dependencies between subsystems as shown in Fig. 4.18. Visualizing porting dependencies is conceptually very similar to the problem of changing ontology. The metaphor here is as follows. A software library which is a dependency of the software system under analysis, is like the domain model in the DMS case. The API of the depended library is like the concepts and attributes of the domain model. A change in the API of the depended library, requiring changes in the source code using the API is like a change in the domain model requiring a change in the decision logic.

As we see, there is a conceptual similarity between change impact in the domain of DMSs and in the domain of software maintenance. We also see that this conceptual similarity allows for a similar practical solution. Hierarchical edge bundling has been applied in the context of software maintenance and comprehension where it proved to be scalable to large software systems. Therefore it is reasonable to expect that it works similarly well in the context of DMSs, where the artifacts (domain model, business logic) do not exceed the size of earlier mentioned software systems. However, this method does contribute only to a very limited extent to the problem we are trying to solve in this thesis. As such, we do not pursue refining this visualization approach further.

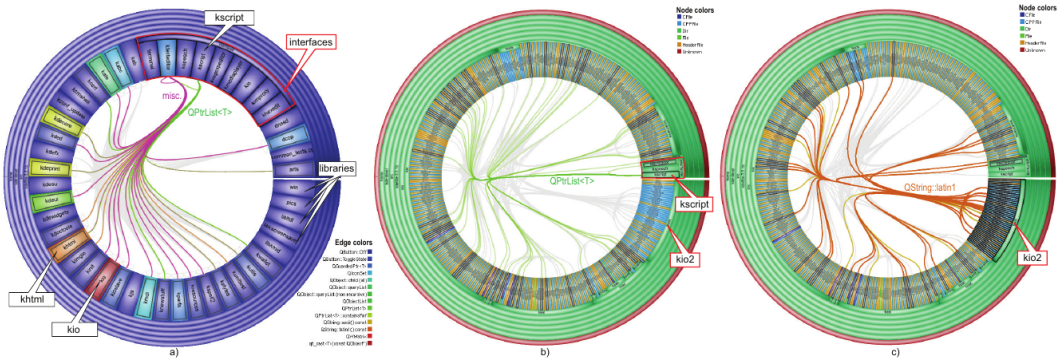


Figure 4.18: Understanding porting dependencies by visualizing query hit relations between subsystems. Image from [173]

4.4 Conclusion

In this chapter we presented various visualization approaches for artifacts that come with a DMS. Baudel used treemaps to visualize both the domain model and the decision logic [140]. We have set a first step to further refine his approach by presenting a generic algorithm for rectangular

space-filling layouts. Using treemaps is one possible solution for our research question **Q1**, gaining insight in the structure of a DM. We disccused the graph visualizations approach by Baudel [140], which visualizes the rule trigger flow and compares flows for different versions of the same DM. While this is a possible starting point for a solution for research question **Q5**, we argued that this approach does not scale to the number of decisions we are dealing with in practice. Finally, we applied the well known circular edge bundling technique of Holten [18] in the context of DMS. We used it to visualize the impact of domain model changes to the decision logic which addresses our research question on change impact (**Q2**).

In this chapter we have shown how to visualize the structure of the DM, the rule execution graph and change impact. We have contributed an improvement for existing techniques in the form of a generic algorithm for rectangular layouts. Although all these artifacts have some relationship to understanding the aggregated effects, none of these techniques gives full insight in these effects by themselves. Being able to understand the structure of a DM is important when it comes to changing the model. Similarly, seeing the impact of a change helps understanding to what extent a planned change to a DM affects the model and might help to detect possible side effects of the change. Both techniques are static techniques though, thus by definition they do not relate runtime data to the structure of a DM. These techniques do not tell if a change is useful. Moreover, these techniques do not gives us additional insight into the statistical relationships between concepts in the model and give no hints about the relationship of a rule to a decision outcome.

We also note that the techniques presented here are basically just visualizations of various data aspects. What we actually need, beyond the mere display of data, are problem-solving techniques which allow a business analyst to incorporate his experience. That is, we do not just want to visualize low level artifact data of a DMS but we want to enable the finding of high-level answers. Finding these high-level answers require a broader approach. First, the answers are not to be found in the individual data spaces, but in the interplay between them. To find answers we need to find ways to combine structure of the DM, structure of the decision logic, statistical properties of the business cases and rule trigger patterns. Secondly, the raw data as we have been looking at right now might not what we want. We want to highlight interesting facts from the modeled knowledge which triggers a business analyst to bring in external context and common sense that has not been modeled. Therefore, in the next chapters we follow the extended Information Visualization (InfoVis) mantra by Keim et al.: “Analyze first, show the important, zoom/filter, analyze further, details on demand” [21]. We apply analytical methods to the data at hand and provide work flows that allow for a more exploratory approach in order to support the high-level tasks we outlined in Sec. 2.6.2.

Published as: Visual Analysis of Multi-Dimensional Categorical Data Sets
Broeksema, B., Telea, A. C. and Baudel, T.
in Computer Graphics Forum, 32

Chapter 5
Visual Analytics for Decision Management Systems

> An approximate answer to the right problem
> is worth a good deal more than an exact answer
> to an approximate problem.
>
> *Super Freakonomics*, JOHN TUKEY

In the previous chapter we have shown that information visualization is an important part of a solution to the problem of gaining insight in the structure and functioning of a Decision Model (DM). However, it also became clear that just visualizing information extracted from the various artifacts of a Decision Management System (DMS) does not provide a full solution. The approaches in CHAPTER 4 focus on the visualization of raw data from individual data spaces of a DM. We concluded the previous chapter with the remark that a better approach would be to combine information from multiple data sources using automated analysis, to discover interesting and relevant bits of information for a business analyst. In this chapter, we take a new look at the data and questions that we introduced in CHAPTER 2. We present a different set of visualization and visual analytics techniques that address these questions. The techniques we present in this chapter where inspired by two early approaches involving analysis of DMS artifacts data. We first discuss these early approaches to identify what is interesting in more detail and to find shortcomings in these approaches that have to be overcome. This discussion is followed by a presentation of our techniques for high-dimensional data and rule triggering analysis.

5.1 Early analytic approaches

When providing analytics in the context of a DMS, first one has to decide what to analyze, second how to present it to the user. Baudel and van Ham took a serendipity approach in two methods for analyzing business rules [174, 175]. The first method determines input attributes that are important for given rule, while the second determines interesting rules for a given rule based on co-occurrence. The idea is to present the user with information while he is editing the business logic that might trigger interesting insight. To this extend these methods analyze input attributes that are relevant for the edited rule and rule trigger patterns that relate to input attributes. We next briefly summarize both methods to provide some background and understanding in the kind of issues that we are interested in.

5.1.1 Input attributes important for rule

One aspect in understanding the function of a DMS is the correlation between the firing of a rule and certain input/output properties. For example a certain rule may be triggered only for people over forty, even though this was not explicitly tested for by the rule itself. By providing means

that bring up this kind of correlations, the business analyst might get a better understanding of the DM at hand.

In [174] Baudel and van Ham propose a method to identify correlations of potential interest. The observation they did is that a given rule only is triggered for a subset of the overall set of decisions. This allows for comparison between attributes. One could for example compare the age distribution of the persons that triggered a certain rule with the overall age distribution, using a Chi-square test to find significant differences. When a rule tests for certain age ranges, the distribution of the subset is very likely to differ and this is therefore not a suprising difference. Therefore additional processing is done to leave out the attributes that are tested in the rule itself.

Baudel and van Ham propose to integrate this method in a visual way in the rule editing environment. When a rule is opened for editing the distributions of the differing attributes can be showed together with the overall distribution.

5.1.2 Rule co-occurrence

In [175] Baudel and van Ham present a related method, focusing on the rules themselves in stead of the attributes. As discussed in Sec. 4.2, rule trigger patterns are interesting, but their exact execution order is not. Following this observation, the presented method keeps track of how often each rule is triggered together with each of the other rules. Based on this book-keeping a similarity score is calculated for each pair of rules. Next, upon editing of a rule, rules with a similarity score above a certain threshold are shown in the editor, marked as potential interesting rules. This prompts the business analyst to verify if the related rules should be changed as well or if external factors may occur which require further modification of the DM.

5.1.3 Discussion

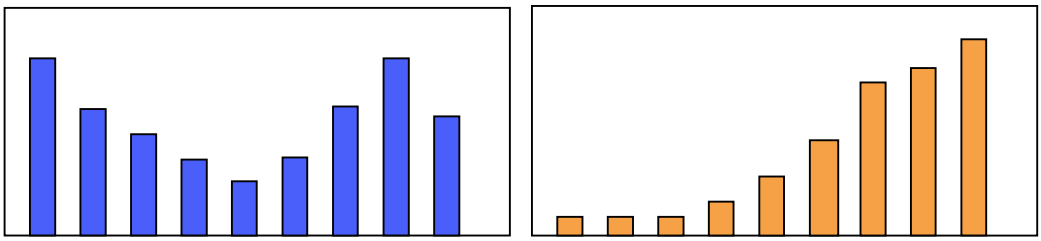
Both methods, discussed in Sec. 5.1.1 and Sec. 5.1.2, perform automated analysis on the extracted artifacts in order to determine what is considered to be important information. The important input attributes method [174] combines statistical analysis of the domain model attribute values with static analysis of the rules from the domain logic. The rule co-occurrence methods [175] method aggregates rule co-occurrence statistics and use a threshold to determine interesting co-occurring rules. Note that, unlike the execution graph visualization, this rule co-occurrence method actually performs some analysis before presenting information to the user.

Interestingly, both methods use aggregated effects to identify important information with respect to the function of a DM. Using these aggregated effects, the methods try to identify patterns that might be unexpected for the user. But what does unexpected mean in this context? Recall from CHAPTER 2 that a DM models human activity. That is, the explicit purpose of rules forming the business logic of a DM is to express in a tangible way how an enterprise wants to treat a certain kind of population or deal with particular events relevant to the business. Whether this is about persons being students, cars having a value in a certain price range or loan applications having a risk score above a certain threshold, looking at the conditions in the rules will tell what kind of population is triggering the rule. Individual rules will not have test conditions for each possible attribute of the domain model, each rule checks a set of conditions that have a semantic connection that makes sense in the particular business context. Thus, there are now two hints that make an attribute "special" *in the context of this rule*: First, the distribution of the attribute values significantly differs from the overall distribution. Second, the values itself are not tested for in the rule. Similarly, rules that co-occur together while having conditions that test for distinct sets of domain attributes, could be said to be unexpected in the sense that the explicit knowledge expressed by the rules do not give hints for the reason of co-occurrence.

Let us assume that the analyst is looking at a rule that tests if a person has an age above forty (we are not interested in the action at this point):

```
IF the age of the person is above 40
THEN ....
```

Using the important input attributes method [174] he might be presented with two distributions of an attribute that differ significant statistically such as the persons income. Fig. 5.1 show how this might look like. Fig. 5.1a shows the income (vertical) distribution versus the age (horizontal) for all persons that passed through the decision service. Fig. 5.1b on the other hand, shows the income distribution for the persons that triggered above rule, i.e. persons with an age above forty.



a. b.

Figure 5.1: Income distributions for the overall population (a) and for people above 40 (b).

The two distributions differ significantly and indeed the explicit knowledge expressed by the rule does explain the deviation or even hint at it. However, it can hardly be argued that this would be an unexpected piece of information for a business analyst. They will not only use the knowledge that is explicitly captured by the DM but will also bring in the contextual knowledge they have themselves about the business domain. Using either common sense or knowledge acquired in different ways, the analyst is likely to expect that older people in general will have higher salaries. In a similar way, it could be argued that it is not unexpected if one of the most co-occurring rules is the one that tests for an income above 50K:

```
IF the income of the person is above 50K
THEN...
```

We can conclude this discussion, by stating that these methods by Baudel and van Ham [174, 175] might present unexpected information to the business analysts but do not necessarily do so. As we have seen, the unexpectedness of these pieces of information depends on the contextual knowledge and assumptions of the business analysts. Both methods are meant to be integrated with DM authoring environment in order to provide contextual serendipity for the authoring task. As a consequence, a business analyst must open a particular rule for editing in order to be presented with this information.

This is a major weakness of a serendipity approach for the following reasons. Both approaches rely on historical data being available, and therefore a first version of the DM is put into production already. When at this point a rule is opened for modification, this means that there already has been some trigger for this modification. A problematic decision could have been identified or the resulting business performance related to the decision service is not meeting the expected service levels. That a particular rule is opened suggests that the business analyst has at least some idea what needs to happen to fix the identified problem or improve the business performance. The information he gets presented by means of the above methods at this stage could indeed help him to further identify issues with the DM related to the problem he tries to solve. However, at the same time it will leave other potential interesting information hidden because there

is no correlation with the rule he is editing at the moment. Thus, for a broader understanding of the functioning of a DM where contextual knowledge can put to use as well a more exploratory approach is required. In the remainder of this chapter we will present two exploratory methods for analyzing the input/output data of a DM and the rule trigger patterns.

5.2 Analyzing business case and decision data

Domain attributes of both the inputs and the resulting decisions are both numerical (e.g. a persons age or the value of a car) and categorical (e.g. the brand of a car or the salary scale of a person). When analyzing this data we would like to combine those attributes to get a complete insight of the correlations in the data. As we already detailed in Sec. 2.3.2, numerical attributes are typically treated as categorical (ordinal) attributes by a DM. This suggests that the appropriate way to analyze the input/output data is by treating it as a multidimensional categorical dataset where numerical attributes are quantized. Categorical data is typically analyzed using Multiple Correspondence Analysis (MCA), an analysis technique related to Principal Component Analysis (PCA) but tailored for categorical data (see also Sec. 3.3.3 and Sec. 5.2.2). Our goal is to expose the often complex correlations in categorical data, and answer questions such as:

1. How do values of one attribute (or variable) relate to values of the same, or other, attributes?
2. How to find clusters of similar observations?
3. How do such clusters relate to a certain value of an attribute?

To meet above goal, we propose several linked views which blend existing and new visualization techniques. While the complexity of MCA analysis is introduced gently to end users, we still allow refining MCA results to extract additional insight.

The main contributions of this chapter are as follows:

- A visualization for the analysis of relationships between the inherent dimensions of categorical data;
- An interactive legend which helps explaining the meaning of dimensions extracted by MCA in terms of dataset attributes;
- An enhanced treeview which integrates raw-data information with the MCA analysis results;
- Interaction techniques that reduce the amount of information shown in the above views and help finding salient data point groups and inherent data dimensions.

5.2.1 High dimensional categorical data

Representation and understanding of high-dimensional datasets is an important issue in many contexts. The dimensionality K of objects under study is a measure of the attributes that describe the structure of these objects. All K dimensions that describe individual objects, together span a K-dimensional data space in which individual objects are defined. These objects are named differently in different contexts: points in mathematics, observations in statistics, instances in computer science and individuals in artificial intelligence. We will use the term *observation* in this thesis.

Each of the dimensions is an attribute a_k, with $0 \leqslant k < K$, of an observation whose structure is defined by these dimensions. The location of an observation in the data space is defined by the

Table 5.1: Basic data types and their properties. Types lower in the table also have the properties of above type(s).

Attribute type	Supported operations
Nominal	values are $=$ or $\neq$ to other values
Ordinal	obeys a $<$ relation
Quantitative	can do arithmetic on values

value that each attribute takes. Each dimension and therefore each attribute of an object has a particular type, i.e. a classification of the kind of values that are considered valid for the attribute. We distinguish three basic types: nominal, ordinal and quantitative [24, p. 20]. Additionally, we talk about categorical attributes, by which we mean the attributes that are either ordinal or nominal. There is an important difference between the two: order of values has a meaning for ordinal attributes but is meaningless for nominal attributes. However, they both lack a general measure of similarity. The properties of each type are shown in Table 5.1. Besides a data type, each attribute has a domain which specifies the range of valid values for the attribute. Depending on the context the data type can be the same for each dimension or a mix of different types.

When observations have only one attribute or dimension, we speak about one-dimensional or univariate data. In the case where observations have exactly two attributes we speak about two-dimensional or bivariate data. Finally, with three or more dimensions we speak about high-dimensional data. A collection of I observations, each having K attributes, forms a dataset D_{IK}. The symbol o_i, with $0 \leqslant i < I$, denotes the ith observation in D_{IK}. The collection of values of attribute a_k for each observation o_i defines the variable V_k.

Let us consider an example dataset, where the objects under study are persons and the questions of interest relate to gender inequality with respect to education and salary. For each person the following attributes are recorded: age, gender, education and salary, resulting in a 4-dimensional dataset (i.e. $K = 4$). The *data space*, i.e. the space which spans all possible combinations of values for all attributes, of this dataset is shown in Table 5.2.

Table 5.2: An example data space with $K = 4$ for a dataset: four dimensions with their type and domain

k	Attribute	Attribute type	Domain
0	Age	Quantitative	$[0..120] \in \mathbb{N}$
1	Gender	Nominal	$Female, Male$
2	Education	Ordinal	$0 - none, 1 - primary\ school,$ $2 - high\ school, 3 - bachelor,$ $4 - master, 5 - doctorate$
3	Salary	Quantitative	$[0..5M] \in \mathbb{N}$

In this example we assume that 100 persons are interviewed, resulting in a dataset $D_{I=100,K=4}$ with 100 observations. These observations o_i can be listed in tabular form as shown in Table 5.3. Each observation o_i is represented as a row, marked with a red border. The value of each attribute a_k of an observation is shown in column k. For example, the green marked cell in Table 5.3 represents the value 28000 of attribute $a_3 = Salary$ for observation o_{100}. Variables are represented by the columns in Table 5.3, such as the blue marked $Gender$ variable.

Table 5.3: An example dataset $D_{I_{=100}K_{=4}}$ represented in tabular form, with an observation in red, a variable in blue and an attribute value in green.

	Age	**Gender**	**Education**	**Salary**
o_1	25	Female	3-Bachelor	25000
o_2	31	Male	4-Master	32000
o_3	51	Male	2-Highschool	12000
o_i	...	...	...	...
o_{100}	32	Female	4-Master	28000

5.2.2 Analyzing categorical data: Multiple Correspondence Analysis

Correspondence Analysis (CA) is an analytical technique for analyzing two categorical variables. The basis of the analysis is a cross tabulation of the two variables under consideration. For example, Table 5.4(a), which shows car price category versus car type. The rows are a classification of the cars based on their price, three categories in this example. The columns on the other hand, classify the cars based on their type. For example, there are 580 cars that fall in the price category $< 2.5K$, 420 of them being a Sedan, 130 a SUV and 30 are luxury cars.

Table 5.4: Cross tabulation for the car type and car value attributes. With (a) the actual frequencies and (b) row normalized values.

(a)

	Sedan	SUV	Lux.	Sum
$< 2.5K$	420	130	30	580
$2.5K - 5K$	150	160	50	360
$> 5K$	20	90	120	230
sum	590	380	120	1180

(b)

	Sedan	SUV	Lux.	Sum
$< 2.5K$	0.724	0.224	0.052	1
$2.5K - 5K$	0.417	0.444	0.139	1
$> 5K$	0.087	0.391	0.522	1

To perform CA, each row is expressed relative to its total as shown in Table 5.4(b). Each row gives the position of a price category in a three-dimensional space. However, because each row adds up to one, each of the rows is projected on a 2D-triangular plane. This is shown in Fig. 5.2. In this figure each dimension, the three car types in this case, is represented by a black axis. A point, for example, the row for cars in the price category $< 2.5K$, represented by the red dot, is projected using its normalized values (blue lines). These points always fall on the triangular plane, represented by the green lines. More generally, when there are N dimensions, with $N > 3$, each row is still projected on a 2D-plane in the N-Dimensional space.

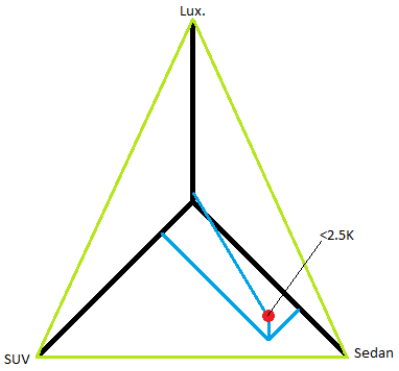


Figure 5.2: Income distributions for the overall population (a) and for people above 40 (b).

CA, calculates the principal axis of the data points in the triangular plane (in the case of the above example). Each axis explains a certain amount of the variance in the data, the first axis explains most. This calculation is very similar to principal component analysis for numerical data.

MCA extends CA to handle multiple (three or more) *categorical* variables [176]. Several variants of MCA exist, all leading to the same equations as pointed out by [177]. In sociology, MCA has been popularized by Pierre Bourdieu [178] as a key tool to find hidden relationships between various sociological factors.

We briefly overview the MCA technique, to form a basis for understanding our visualization, and to show the MCA interpretation problems that our visualization addresses next. For a thorough understanding of (M)CA, we refer to [179, 176] on which our implementation is based. MCA operates by first converting data from categorical to numerical form. Naively assigning a numerical value to each possible categorical value of an attribute can create artificial, arbitrary, distances between two values, which would cause misinterpretations. In contrast, MCA encodes each categorical attribute with a bitmask, one bit for each possible category value. For example, if the domain of an attribute *Car.Type* is $[Sedan, SUV, Luxury]$, the value $Sedan$ is encoded as $[100]$ and the value $Luxury$ as $[001]$. This effectively replaces a single attribute in the original dataset with several new (binary) attributes. These binary attributes are stored in a so-called indicator matrix of which we give an example in Table 5.5. Note how for each attribute only one of the values will be set to one for a given observation o_i.

Table 5.5: Indicator matrix encoding three attributes each having three distinct values.

	Car.Type			Car.Value			Car.Airbag		
	Sedan	SUV	Lux.	$< 2.5K$	$2.5K - 5K$	$> 5K$	None	D	D + P
o_1	1	0	0	1	0	0	0	1	0
o_2	0	0	1	0	0	1	0	0	1
...	...	...	...	...	...	...	...	...	...
o_I	0	1	0	0	1	0	0	1	0

The indicator matrix is next processed using standard CA techniques. For a table with I observations, each with K attributes which in turn have J_k levels or distinct values $\{v_k^1, \ldots, v_k^{J_k}\}$, let $\mathbf{X}$ be the $I \times J$ indicator matrix, where $J = \sum_1^K J_k$. Performing CA on $\mathbf{X}$ gives a row factor score and a column factor score. These factor scores are the projections of observations (rows) and attribute values (columns) on the eigenvectors of $\mathbf{X}$. Let N denote the grand total of matrix $\mathbf{X}$. The first step of MCA is to compute the probability matrix

$$\mathbf{Z} = N^{-1}\mathbf{X} \tag{5.1}$$

Next, we define the vectors $\mathbf{r}$ and $\mathbf{c}$ which contain the row, respectively column, totals of $\mathbf{Z}$. Let $\mathbf{D_c} = diag\{\mathbf{c}\}$ and $\mathbf{D_r} = diag\{\mathbf{r}\}$ be matrices whose diagonals are $\mathbf{c}$ and $\mathbf{r}$ respectively. We compute the factor scores by solving the following Singular Value Decomposition (SVD) [180]. Let

$$\mathbf{A} = \mathbf{D_r^{-1/2}} \left(\mathbf{Z} - \mathbf{r}\mathbf{c}^\top\right) \mathbf{D_c^{-1/2}} \tag{5.2}$$

then the SVD to solve is

$$\mathbf{A} = \mathbf{P}\Delta\mathbf{Q}^\top \tag{5.3}$$

with

$$\mathbf{P}^\top\mathbf{P} = \mathbf{Q}^\top\mathbf{Q} = \mathbf{I} \tag{5.4}$$

Here, Δ is the diagonal matrix of the eigenvalues of $\mathbf{A}\mathbf{A}^\top$, $\mathbf{P}$ are the eigenvectors of $\mathbf{A}\mathbf{A}^\top$ and $\mathbf{I}$ is the identity matrix. $\mathbf{Q}$ are the eigenvectors of $\mathbf{A}^\top\mathbf{A}$. From the SVD we compute the row factor scores

$$\mathbf{F} = \mathbf{D_r^{-1/2}}\mathbf{P}\ \Delta \tag{5.5}$$

and the column factor scores

$$\mathbf{G} = \mathbf{D_c^{-1/2}}\mathbf{Q}\ \Delta \tag{5.6}$$

by projecting the attributes on the respective eigenvectors. In addition to the factor scores two additional pieces of information can be calculated that help the interpretation of the analysis result: weights and contributions.

The weight of each column in $\mathbf{X}$ reflects its importance for discriminating between observations (rows). Recalling that each column in $\mathbf{X}$ represents an attribute value, the weight thus reflects if a certain attribute value helps separating observations selecting this value from the other observations. By aggregating the values per attribute, we can also use the weight to reflect the importance of each attribute for discriminating between observations. The weights are calculated by first determining the barycenter $c^\top$ of X, i.e. the average observation

$$\mathbf{c}^\top = \left(\underset{1\times I}{\mathbf{1}} \times \mathbf{X} \times \underset{J\times 1}{\mathbf{1}}\right)^{-1} \times \underset{1\times I}{\mathbf{1}}\mathbf{X} \tag{5.7}$$

where $\underset{1\times I}{\mathbf{1}}$ is a one by I vector of ones and $\underset{J\times 1}{\mathbf{1}}$ is a J by one vector of ones. The weight of each column of $\mathbf{X}$ is defined as the inverse of the column component in the barycenter

$$\mathbf{w} = [w_j] = [c_j^{-1}] \tag{5.8}$$

Contributions help locating the columns (attribute values), important for a given factor. The contribution b of column j to factor l can be calculated as

$$b_{j,l} = \frac{c_j g_{j,l}^2}{\lambda_l} \tag{5.9}$$

where c_j is the weight of the j^{th} attribute value, $g_{j,l}$ is the projection of the j^{th} attribute value on the l^{th} factor and λ_l is the eigenvalue of the l^{th} factor.

5.2.3 MCA Visualization pipeline

Visualizing the MCA results now follows the classical scatterplot technique used for MDS: we take the two factors e_x and e_y along which the data has most variance, and plot all observation projections, i.e. factor scores, along e_x and e_y. In contrast to Multi Dimensional Scaling (MDS), we can also draw the *attributes* in the same plot: these are simply the projections of the J points having one for a particular attribute value and zero for all others. Recall that the original table was enlarged, by adding a binary column for each distinct attribute value. As a result, the attribute projection contains a point for each of the J distinct values as opposed to the K distinct attributes.

Fig. 5.3 outlines our approach. We start with a table containing categorical and/or numerical attributes. Next, we refine this table by quantizing numerical (ratio and interval) attributes to convert them to ordinal attributes. From this refined table, we construct the indicator matrix. MCA extracts correlation information from this matrix, which we use to create our visualization.

Quantization is a required step to incorporate numerical attributes into the analysis. For example, an *Age* attribute can be quantized to a five-class ordinal attribute [$0 :< 20,\ 1 : 20..30,\ 2 :$

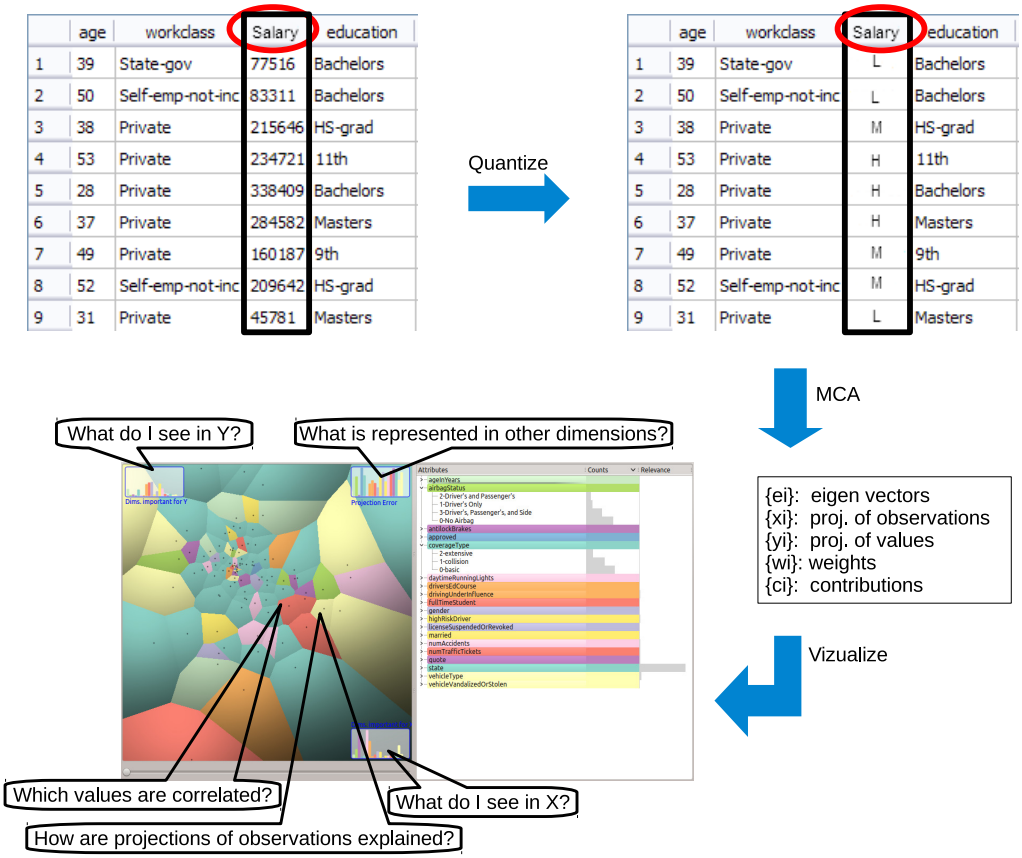


Figure 5.3: MCA visualization pipeline. Input: multidimensional table with numerical and categorical data. Numerical columns (e.g. salary in three levels: L, M and H) are quantized. MCA is performed on the quantized table. MCA results are used for visualization.

$30..40,\ 3\ :\ 40..50,\ 4\ :>=\ 50]$ (years). The number of categories, and the quantization method (constant range or constant area in histogram) is configurable for each numeric attribute. Quantization settings are application-specific, in [181] an interactive technique is presented for quantification of numerical and categorical attributes. It should be noted that by quantizing numerical attributes, some information is lost. It is not possible though to include numerical attributes unchanged in the indicator matrix. This is due to the fact that this would lead to comparison or analysis of two different kinds of information (frequency data and ratio/interval data). The results of such an analysis would be meaningless. Additionally, one should recall that this loss of information also happens in DMSs. We detailed in Sec. 2.3.2 how decision tables, a common part of DMSs, treat numerical data as categorical data. Therefore, in the context of DMSs, it makes sense to start with the quantization of attributes that mirrors the limits used in the decision tables of the DM. Once the analyst has a first impression of the consequences of such quantizations, he can use approaches such as presented in [181] to analyze different scenarios.

5.2.4 Interpretation challenges

As outlined above, MCA creates a plot containing both observations and attributes. Interpreting this plot is based on proximity of points of the *same* kind: two *observations* plotted close to each other imply that they have similar attribute values or, for categorical data, that they *share* several attribute values (since two categorical values can either be equal or different). *Attributes* plotted

close to each other are interpreted differently in CA and MCA. In CA, columns are actual different attributes in the input data. Hence, when two attribute points are close, observations *tend to be similar* with respect to these attributes. In MCA, two columns can be either (categorical) values of the same attribute or values of two different attributes from the original data, given the bit structure of the indicator matrix $\mathbf{X}$ (Sec. 5.2.2). Close plotted points *for values from different attributes* imply that observations tend to select these values together. For example, when $Brand : BMW$ and $Antilockbrakes : true$ are close together, it means that most of the times BMW cars tend to have antilock brakes. Close plotted points *for different values of the same attribute* imply that observations select either of these values and are similar with respect to the other attributes. For example, when $Brand : BMW$ and $Brand : Audi$ are close together, it means that BMW and Audi cars tend to have similar properties. That is, they typically fall in the same price range and select similar security features.

Although the mathematics of MCA is relatively straightforward, interpreting MCA plots is clearly not. This is firstly due to the abstract nature of the computed quantities, which do not directly map to the user's world (observations and attributes). Secondly, for many observations and/or attributes, 2D scatterplots of observation and attribute factor scores get cluttered. Thirdly, on a technical level, data outliers can influence the factors (eigenvectors): The 2D plot space gives too much space to outliers and too little space to 'interesting' observations.

Without adequate tooling, potential insights delivered by MCA risk being lost. Hence, we want to provide intuitive interactive visualizations of MCA analysis results, to address the following questions:

- How to link the MCA results (factors, factor scores) to the meaning of the original data (observations and attributes)?
- How to show the meaning of the projected dimensions?
- How to explain the grouping of projected observations?
- How to eliminate irrelevant (outlier) dimensions or outlier values of a dimension?
- How to get an overview of values that occur together?

5.2.5 Visualization overview

To address the above goals, we propose a visualization with two main views: the *dimensions view* and the *projections view*. These support the steps of observation classification and observation exploration: First, one wants to *classify* data to a granularity level suitable for the task at hand. For example, in a car insurance dataset, finding that students, expensive cars, and many accidents are strongly correlated, leads one to classify such observations as "students causing accidents in expensive cars". Once users have a clear picture of the classes occurring in the data, they next explore the observations to give sense to clusters and understand outliers.

The *dimensions view* shows the attributes and their domains. It serves both as an analysis entry point and as a legend for the more complex *projections view*. The *projections view* shows the factors computed by MCA; it targets questions related to correlations and variances of observations and attributes. The two views are linked via shared colormaps and selection, to support asking questions in one view and using the other view to understand the results (Fig. 5.4). As an example, we next use a set of 5000 US car insurance quotations, with 19 attributes per observation. Table 5.6 shows these attributes, and how numerical attributes have been reduced to categories by binning.

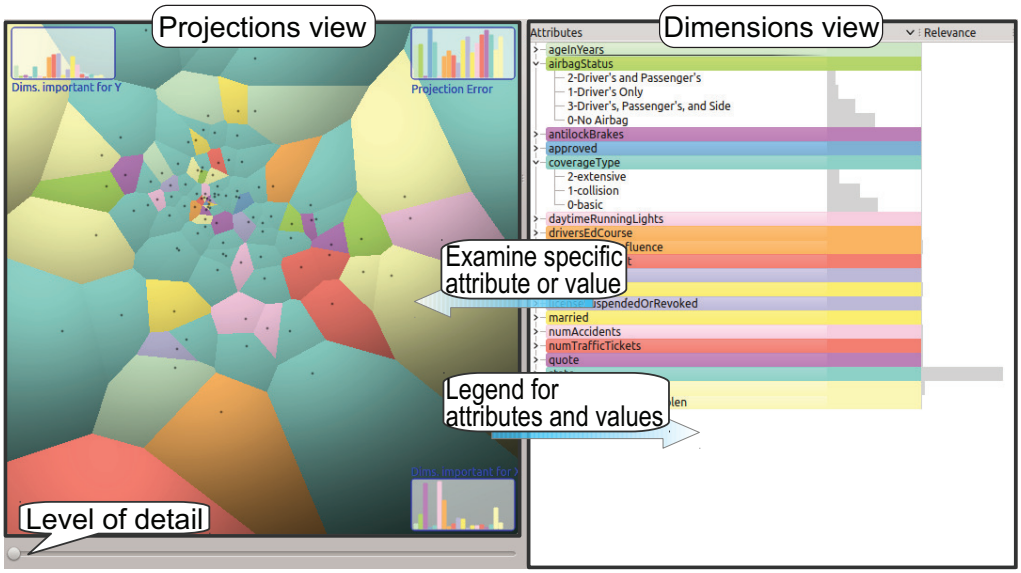


Figure 5.4: MCA visualization overview.

Table 5.6: Data types for the US car insurance dataset. Names between brackets are the labels used in images.

Attribute	Type	#Bins	Values
ageInYears (age)	integer	4	$< 35, 35..53, 54..72, > 72$
airbagStatus	category	4	none, driver only, front seats, all
antilockBrakes	boolean	2	true, false
approved	boolean	2	true, false
coverageType	category	3	basic, collision, extensive
daylightRunningLights (lights)	boolean	2	true, false
driversEdCourse	boolean	2	true, false
drivingUnderInfluence	boolean	2	true, false
fulltimeStudent (student)	boolean	2	true, false
gender	category	2	male, female
highRiskDriver	boolean	2	true, false
licenseSuspendedOrRevoked	boolean	2	true, false
married	boolean	2	true, false
numAccidents	integer	4	0, 1, 2, >= 3
numTrafficTickets	integer	4	0, 1, 2, >= 3
quote	USD	4	$< 310, 310..619, 619..1043, > 1043$
state	category	50	AL, AK, AZ, . . ., WY
vehicleType (vehicle)	category	9	compact, sedan, luxury, sport, pickup SUV, sport-luxury, collection, van
vehicleVandalizedOrStolen	boolean	2	true, false

Dimensions view

The *dimensions view* (Fig. 5.5) shows the attributes present in the raw dataset which is the input of the MCA analysis. Recall that a K-dimensional dataset yields an indicator matrix with J binary attributes, where each binary attribute shows whether an observation selects a given value (Sec. 5.2.2). We show the raw data using a two-level tree: attributes and values. The first level shows all original attributes. On the second level, each attribute has J_k children, i.e. all its categorical values $\{v_k^i\}$. Attribute nodes are colored as follows. First, we sort attributes based on

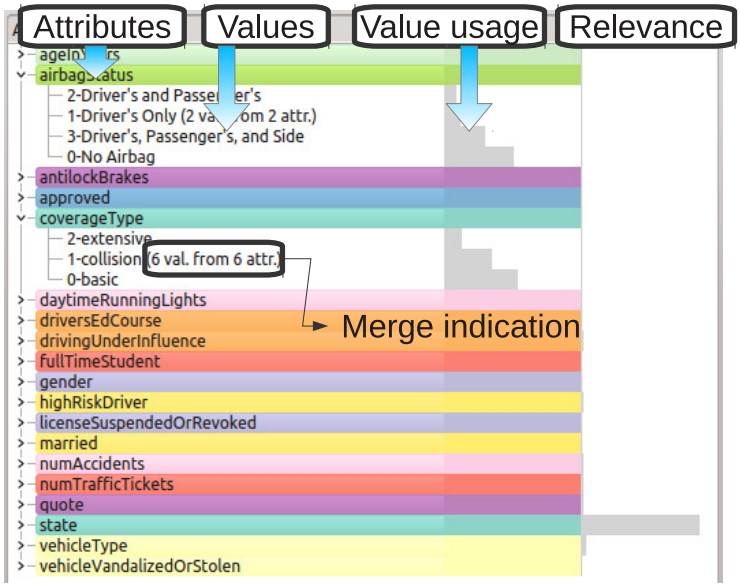


Figure 5.5: *Dimensions view* for the insurance dataset

decreasing relevance (Eq. 5.8) and assign them colors cyclically from a fixed categorical colormap with $C = 10$ hues [182]. Next, we set the nodes' color saturations to their attributes' relevance. Given the attribute sorting, even if two nodes have the same hue (for datasets with more than C attributes), their colors will differ in saturation: most important attributes are bright, and less important ones are dull. We stress that color mapping is not a main contribution of our work: if available, better techniques should be used. Attribute nodes are labeled by their dimension names. Value nodes are labeled by a textual description, see, e.g. the *ageInYears* integer attribute (Fig. 5.5 top) which is quantized in 4 values (< 35, $35..53$, $54..72$, > 72 years). Value nodes show three additional properties:

- The percentage of observations with that value, as a bar. This shows which values occur most in the dataset. We later refine this insight to find if such values are indeed discriminative for the correlation of observations or not.
- The attribute value weights w_j, or relevances, computed using Eq. 5.8. Large weights show attribute values which are important for discriminating between observations.
- Value and attribute merging.

Sorting the *dimensions view* on the value usage column shows the distribution of values for a particular attribute. To find attributes and values which discriminate between observations, the view can be sorted on the relevance column. This relates to value column: Values which are rarely used by the observations may provide more information for discriminating between observations and are therefore more relevant; frequently used values are less interesting [179]. The *dimensions view* serves as a *legend* for the more complex *projections view*, which we present next.

Projections view

This view displays projections of both observations and attributes computed by MCA. It helps finding correlations and variances in the input data, i.e. answer questions such as which attributes contribute to a given factor; along which attributes are certain observations most (or

least) similar; and what is the meaning of a factor. We use the classical MDS approach: We draw a scatterplot by projecting all observations $\mathbf{x}_i$ and attribute values v_k^i on the two most important factors computed by MCA (Sec. 5.2.2). We next add several visual enhancements to this plot, as follows.

Recall that close projections of values of the *same* attribute mean that observations selecting any of these values are similar *vs* their other attributes (Sec. 5.2.4). Close projections of values of *different* attributes imply that observations tend to have these values for the respective attributes together. In both cases, we want to find (a) the relative distances between projected values and (b) how these values are grouped within categories.

We support this task by drawing a Voronoi partitioning of the 2D plot space, with the projected values as sites. A Voronoi partitioning of the space results in cells that contains the points that are closest to the site that identifies the cell. Cells make reasoning about distances easier and also provide handles for interaction. The former is particularly important for the observations plot, where the cells help to identify which attribute value(s) are closest to an observation. Using a larger area, while guaranteeing no overlap, make it more convenient for the user to interact with individual values, projected in the plot. Cell colors show their categorical attribute, as in the *dimensions view* (Fig. 5.5). To separate small cells of similar colors, we use parabolic shaded cushions, akin to [183]. Finally, we label cells with their categorical values. Labels are centered and clipped to fit in the inscribed circle in each cell. Tooltips with the full labels are shown when brushing over the cells. Additionally, brushing links the *projections* and *dimensions views*.

Fig. 5.6 shows the *projections view* for the car insurance dataset. As the *state* attribute (light blue) has many values (50) relative to other attributes, we see many such cells. In the center we see a cluster of small non-state cells (different hues than light blue) surrounded by state cells (light blue). Among these non-state cells are *quote:* < 310, *quote:* $310..619$, *vehicle type: sport*, *vehicle type: van*, and *#accidents:* < 1. Since these cells are small, their projected values are close, so we infer that many observations select these values together. Further, we infer that customers from states surrounding these cells, e.g. CA, FL, NJ, tend to have such a profile, while customers from states that are at the periphery, e.g. SD, AK, NV, have different profiles. Note that the distance metric is important here: the exact locations of the Voronoi cell borders *vs* the observation projections is not decisive; the distance from the attribute projections to the observation projection is. The Voronoi cells serve as a means to determine which is the closest attribute projection for any given observation projection. This is determined by observing in which cell an observation projection is located. By definition of a Voronoi tesselation, the observation projection is closest to the attribute projection that serves as site of the cell in which the observation projection resides.

In Fig. 5.6 right, we see cells for the values *(daytime running) lights: true*, *airbag status: all seats*, and *vehicle type: SUV, luxury, sport-luxury*. On the left, we see *lights: false*, *airbag status: none*, and *vehicle type: collection, compact, pickup*. This shows that the X axis of this view maps the car class (left = cheap cars with few options, right = expensive cars with many options). This pattern could be related to wealth of the insured persons. Since wealth is not an attribute in our data, this is an interesting finding.

At the top of Fig. 5.6 we find cells for the age categories $35..53$ and $54..72$ and married people. At the bottom, we find people below 35 and who are full-time students. Hence, the Y axis maps the phase of life people are in. Finally, outlier cells, such as students (Fig. 5.6 bottom-left), show that observations that select this value, *i.e* full-time students, share less values with the other observations as compared to observations that select values in the central cells.

In brief, the attribute plot can be interpreted as follows:

- values in central cells are used by the average person type;
- values in periphery cells are used by outlier persons;

- the Y axis reflects life phase, with senior people at top and young people at the bottom;
- the X axis reflects car prices, with more expensive cars at right and cheaper ones at left.

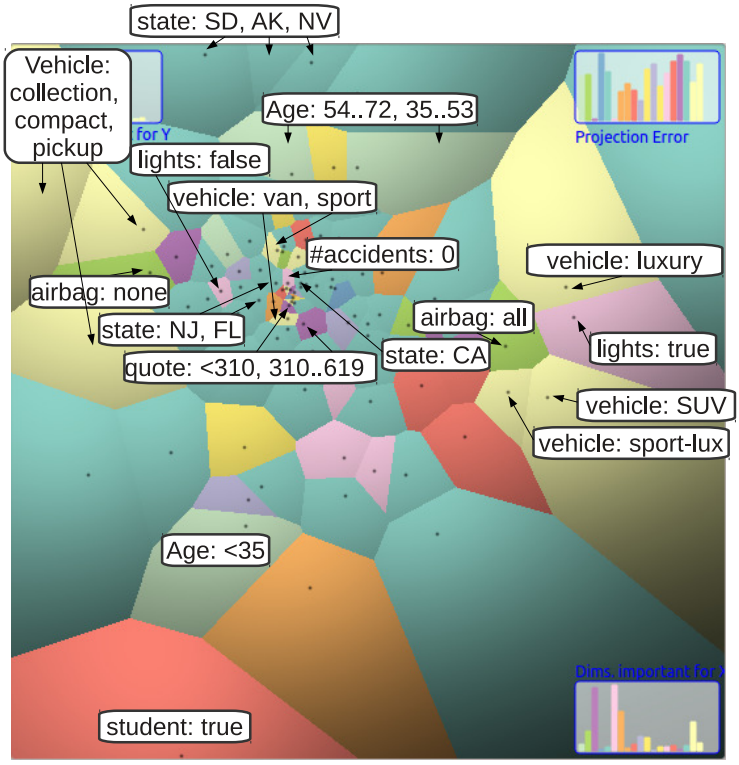


Figure 5.6: *Projections view* with attribute values. Labels and arrows are added here manually for illustration purposes.

Finding meaningful clusters by value cell merging

The *projections view* (Fig. 5.6) can easily get crowded, since it shows as many cells as there are different attribute *values* in the input dataset (104, in our case). One task we want to address is classify data in clusters at a level that is meaningful for the analysis goals at hand. To support this, we provide four ways to cluster and filter data:

- Leave out attributes from the analysis;
- Cluster attributes in the attributes view based on distance;
- Cluster values of one user-selected attribute;
- Filter observations based on attribute value.

An attribute can be left out when it is of no importance for the analysis. This creates more space for the remaining values. For instance, when we remove the *state* attribute from Fig. 5.6, there are 50 cells less in this view. However, this may result in information loss, so users should decide which attributes are relevant for each analysis on a case-by-case basis.

As explained, values who project closely show that observations are very similar with respect to these values. Hence, we are not interested to examine such values separately – instead, we

want to find *clusters* of values at various levels of detail, which show us the properties that define a homogeneous subset of our observations.

To find such clusters, we add a level-of-detail option, controlled by the slider shown under the *projections view* (Fig. 5.4). The slider maps a distance δ in 2D (projection) space. When the user changes δ, we iteratively merge pairs of value-projections which are closer than δ into a new cell whose barycenter is the average of the merged cells' barycenters.

More precisely, this clustering is based on the observation that the smallest cell is the center of a cluster. Therefore we sort the cells of the Voronoi partitioning by size and start the merging process with the smallest cell. Each of the neighboring cells is merged when its site is within distance δ from the site of the current cell. The current cell and its neighboring cells are marked as processed. This process is repeated until each cell is processed. Next a new Voronoi tessellation is generated where the sites are the barycenters of each group of merged cells from the previous step. The whole process is repeated until no merges take place. In the extreme case this means that all projected attribute values are now in one single cell, in all cases in between it means that no two sites are within distance δ of each other.

Fig. 5.7 shows the effect of merging: Most small cells at the center of Fig. 5.6 have now been merged, by grouping attribute values which are selected by the average person. The merged cells can now represent (a) either values of the *same* categorical attribute, or (b) values of *different* categorical attributes. Outlier cells, however, stay roughly unchanged. Hence, we use less cells to show the concept of average person, but keep the cells that show outlier persons.

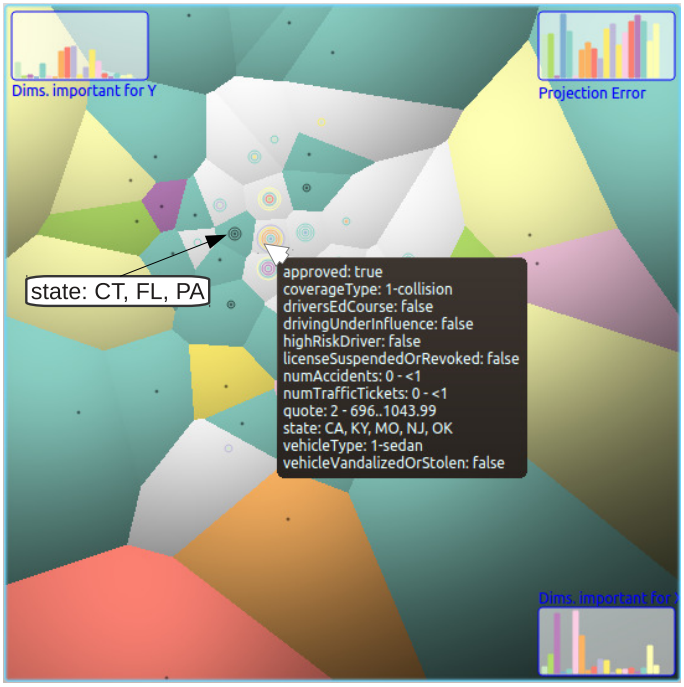


Figure 5.7: *Projections view* with merged value cells. White label added manually for illustration purposes.

To show which values get merged, we draw a set of concentric rings around the merged cells' sites. The number of rings equals the number of merged values within a cell. Cells containing only values of the same attribute are colored using that attribute's hue, as before, and the rings are colored black. For example, in Fig. 5.7 we see a cell grouping all states CT, FL and PA. Cells containing values of different attributes are colored in light-gray, a reserved color not used in the attribute colormap, to show that they groups different attributes. Rings in such cells are colored

by the colors of the merged attributes. We read this visual encoding as follows:

- cells with many rings contain many attribute values;
- non-light gray cells with many rings contain merged values of the same attribute; the cell's color shows the attribute;
- light gray cells contain merged values of different attributes; the rings' colors show the attributes;
- cells with no rings encode individual, non-merged, values.

In Fig. 5.7, the light gray cell (under the mouse) contains many rings, i.e. many merged values. The rings' colors show that this cell groups values from the attributes *coverageType*, *drivingUnderInfluence*, and *vehicleType*. The tooltip shows details on demand, i.e. the merged values: *coverageType: collision*, *drivingUnderInfluence: false*, and *vehicleType: sedan*. From this data, we infer that this cell groups people who drive safely (no accidents, no traffic tickets, not caught for driving under influence) but who still request a collision-coverage insurance, which is an interesting finding. We also show merging information in the *dimensions view*. Fig. 5.5 shows how (at a different merging level) $collision$ is merged with six values from six attributes. When clicking on a cell in the *projections view*, values merged in this cell get highlighted in the *dimensions view*.

Value filtering and merging

Merging value cells reduces the cell count while keeping the information encoded by the merged cells in the view. However, the user controls this process only globally, via the projection distance. For finer-grained ways to reduce the cell count, we provide a filter/merge view (Fig. 5.8). Filtering and merging follows three simple steps: (1) select an attribute; (2) select one or more values thereof; (3) perform filtering or merging.

When an attribute v_k is selected by clicking on its cell in the *projections view* or tree item in the *dimensions view*, the filter/merge view shows all values v_k^i of v along with their cell sizes in the *projections view*. Sorting this list lets one pick the largest cells, which typically appear at the periphery of the Voronoi diagram, and thus take considerable space that could be used to show more detail in the crowded areas. After the desired attribute values are selected, one can filter or merge the data based on this selection.

Filtering removes observations which have any of the selected attribute values. For example, to get more insight in student characteristics, we select the *fulltimeStudent: false* cell, filter, and thus remove all non-students from the view. After filtering, MCA is recomputed automatically on the filtered data. This updates the views with a new projection with removed outliers, thus more space for the interesting observations. In our example, our analysis will now only concern students.

Merging simplifies the visualization by replacing several values v_k^i of an user-selected attribute k with *one* new value v_k^{new}. Like for filtering, MCA is done anew after merging and all views are updated. Unlike filtering, merging n attribute values will remove exactly $n-1$ cells from the *projections view*, since there is exactly one cell per attribute value. For example, consider the *states* attribute, which has fifty values. Recalling the earlier analysis in the *projections view* section, we have found cells on the right of Fig. 5.6 as high-income-related, and cells to the left as low-income-related. If we accept this meaning, we can now merge states on the right of Fig. 5.6 to a new value *high-income states*, states to the left into a new value *low-income states*, and the remaining (center cells) states *moderate-income states*. Fig. 5.9 shows the updated view. The view has a similar layout as before merging (modulo a rotation which is an unfortunate side-effect of MCA), but offers now more space to other values than states, since we now have three state values instead of fifty.

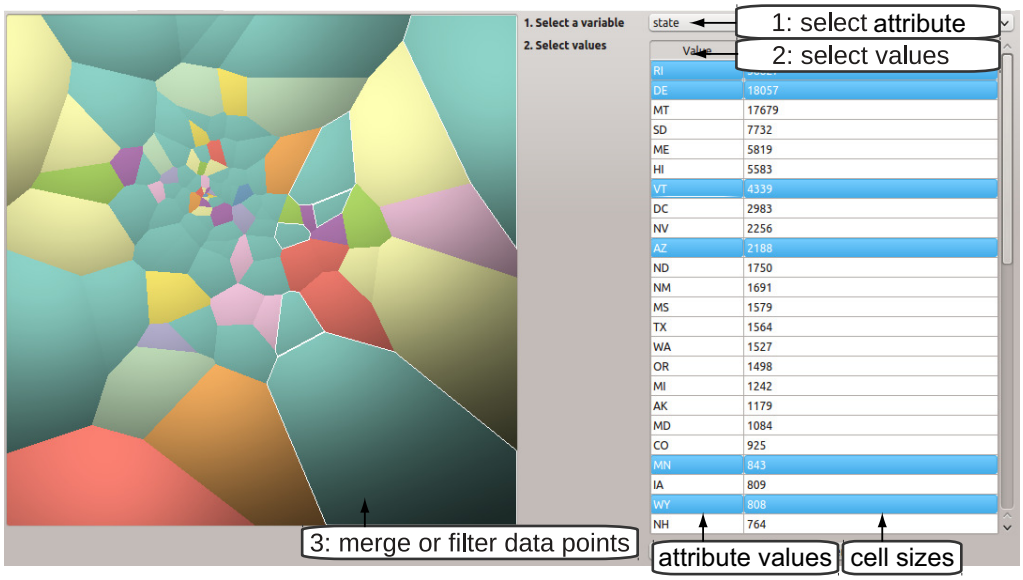


Figure 5.8: The merge/filter view.

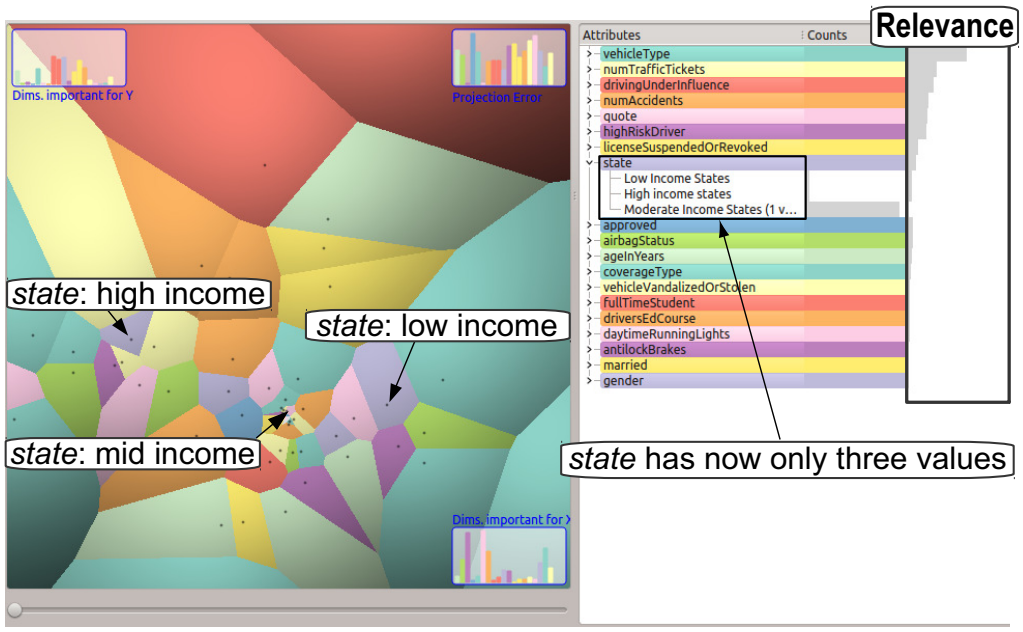


Figure 5.9: Merging states into three different groups.

The relevance metric for attribute values, shown in the *dimensions view* (Fig. 5.9 right), serves here two purposes. First, we can use it to select which attribute values we want to filter or merge – the less relevant ones. Secondly, this metric tells us how attributes change their relevance (for distinguishing between observations) after a filter or merge was applied. This helps iteratively reducing the dimensionality of the dataset by incrementally merging less relevant values into higher-level concepts, and also helps users focus on the most relevant concepts at a given level of detail.

Projection legends

Dimensionality reduction techniques like MDS or MCA typically project the data along $N \in \{2, 3\}$ eigenvectors, and draw projections as N-D scatterplots. However, such plots can be hard to read by many business users. One issue is that *axes* have no explicit meaning: These are factors, coming from the SVD in the MCA case. Ideally, we would like to explain the axes in terms of the variance of attributes and attribute values.

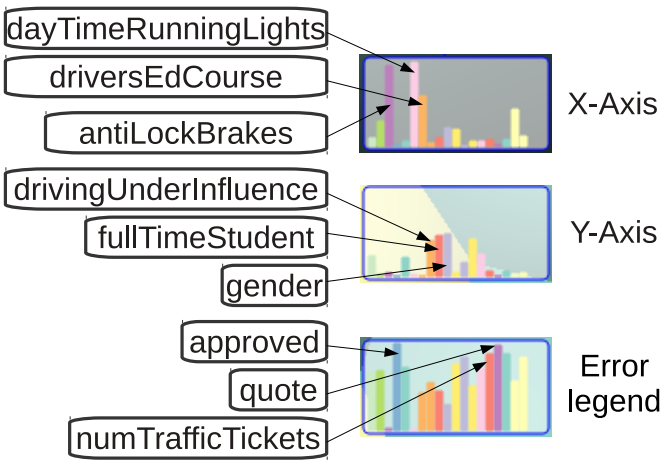


Figure 5.10: Zoomed-in *projection legends* from Fig. 5.6 with attribute contributions.

As explained in Sec. 5.2.2, for each attribute value its importance for a given factor can be calculated (Eq. 5.9). Using this equation we can construct the contribution vectors $\mathbf{b}_x = \{b_x^i\}$ and $\mathbf{b}_y = \{b_y^i\}$ for the two factors used to draw our 2D scatterplot. The values b_x^i and b_y^i give the contributions of all values of attribute i to the x and y plot axes. We can now *explain* what the x and y axes mean in terms of a mix of attribute values from the input data. Still, $\mathbf{b}_x$ and $\mathbf{b}_y$ are not in the optimal form for interpretation: Since MCA uses one column for each attribute *value*, our vectors $\mathbf{b}_x$ and $\mathbf{b}_y$ have J elements, one for each attribute value. We simplify the contribution vectors by summing up all values that correspond to the same attribute. The resulting contribution vectors $\tilde{\mathbf{b}}_x$ and $\tilde{\mathbf{b}}_y$ have now K elements, i.e. as many as the number of input attributes. Their elements indicate the contribution of each separate attribute (and not attribute value) to the plot axes.

We show these values by interactive barchart legends [184] on the x and y plot axes. This approach is somewhat similar to [185]. However, we only show the plots for the projected factors and add interaction to the barcharts, as explained the following section on *Observations plot*. Each bar is colored by the hue of its corresponding attribute, as in the *dimensions view* and *projections view*.

Fig. 5.10 shows a zoom-in of the *projection legends* for the insurance dataset in Fig. 5.6. The x legend has two large bars for the attributes *antilockBrakes* and *daytimeRunningLights*. If we brush the view, we see indeed that these attributes have extreme values at the left and right of the x axis respectively – see e.g. the cell *daytimeRunningLights: true* right in Fig. 5.6.

MCA shows the input data projected along the two most relevant factors. However, datasets may be inherently of higher dimensions than two [110]. Hence, a MCA (or similar) 2D projection may convey false insights if much of the data variance occurs along the 'discarded' dimensions. We show this by a third barchart: the *error legend* (Fig. 5.10). This barchart is built similarly to our previous ones, but it shows the sum of contributions of all factors except the two used for the actual projection. We read this chart as follows: Short bars show attributes whose variance is well captured in the 2D projection. Long bars show attributes whose variance is captured mainly

by factors *not* used in this projection. Seeing such large bars, users can either (a) continue the analysis, but refrain from making judgments about these attributes; or (b) select one of the x or y dimensions in the current view to use the factor that has the largest variance for the attribute of interest. This can be done by shift-clicking on the respective attribute bar.

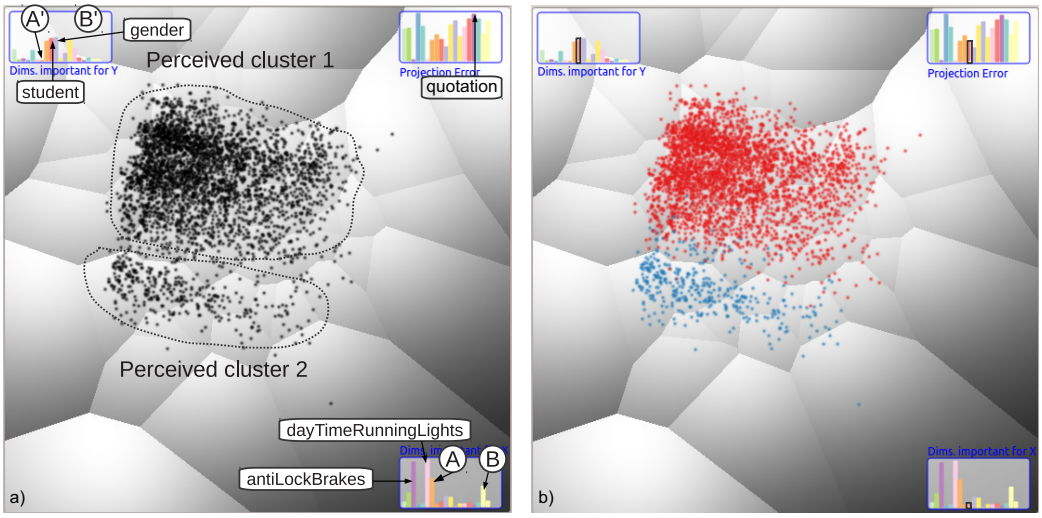
Observations plot

As explained before, both observations and attribute values are projected in our 2D plot space. So far, we showed how attribute values are visualized.

Fig. 5.11 shows the *projections view* used to explore observations, a view we call the *Observations plot*. Typically, one has many more observations than attribute values. To remove clutter, and show observation density, we draw observations using additive alpha blending. Attribute cells are shown in the background, but grayed out, so we can use colors to show the observations' attribute values. The relation between attribute cells and observations is as follows: If an observation $\mathbf{x}_i$ is closer to an attribute value j than to other attribute values $k \neq j$, then $\mathbf{x}_i$ will more likely select value j than select values k *relative* to the other observations. Showing the attribute cells helps assessing such relations without having to visually locate attribute value projections, which is hard given the dense *Observations plot*. To find more information about a cell close to observations of interest, we can switch the view to the attribute plot. The cell layout stays the same, so users keep their mental map.

Observations tend to form clusters (groups of closely packed points) in the *Observations plot*, based on similarity. A standard analysis task is to explain such clusters. We assist this by adding functionality to the *projection legends*: When clicking a bar in the x, y or error barcharts, observations are colored using a categorical colormap on the values of the bar's attribute. This colormap is different from the hue mapping used in the *dimensions view* and *projections view* and has a different purpose: The hue map shows the *identity* of an attribute, i.e. links the *projections view* with the first tree-level in the *dimensions view*. The value colormap shows the different *values* of an attribute, i.e. links the observations plot with the second tree-level of the *dimensions view*.

Fig. 5.11a shows an example. Two separate clusters are apparent. Both spread along both x



a. b.

Figure 5.11: *Observations plot*: (a) without selection; (b) with 'full-time student' attribute selected (blue = student, red = non-student). Legends help confirming that the clusters reflect the student status attribute.

and y axes, i.e. along the two factors used to create this projection. Hence, if these clusters are determined by some attribute, this attribute contributes to *both* the x and y factors, otherwise the clusters would be one-dimensional (lines). We use this hint and the *projection legends* to explain the clusters, as follows. First, the clusters cannot be explained by the two long bars in the x *projection legend*, *antilock brakes* and *daytime running lights*, since these attributes contribute almost fully to the x axis, as shown by their long bars which reach almost 1. The next two longest bars in the x *projection legend*, A and B, are about half height, so they contribute only 50% to the x factor. However, they have no contributions to the y factor (very short bars A' and B' in the y *projection legend*), so they cannot explain the spread along y. In the y *projection legend*, we see three attributes that contribute almost equally to this axis. The longest one, *gender*, does not explain the clusters, since it is short in the x *projection legend*. We have now two remaining possibilities. Clicking the second-longest bar in the y *projection legend* (*fulltime student*) colors observations based on this attribute, i.e. students=blue and non-students=red (Fig.5.11b). The colors match the perceived clusters, so we conclude that the clusters reflect the student status.

In Fig. 5.11b, we see that students are mostly present in the lower left area of the plot. If we read the attribute labels for the cells in this area (Fig. 5.10), we find that students have a

- lower probability of being married;
- higher probability of being under 35;
- higher probability of being caught driving under influence;
- higher probability of having there license suspended;
- higher probability of causing accidents.

Such findings are evidence of an increased risk for accidents under students. Analysts could use this to adjust insurance quotations. Let us see if this was the case in our data. Looking at the error *projection legend* in Fig. 5.11, we see that the insurance *quotation* is high, i.e. it contributes very little to the x and y factors, and a lot to the other factors not used in this projection. If quotation and student status were correlated, the quotation attribute should have contributed visibly to the y axis which, as we saw, explains the student status attribute. As this does not happen, it means the quotation is not correlated with student status, even though student status is correlated with accident risk.

5.2.6 Discussion

Our visualization, in contrast to MDS techniques, can technically handle both categorical and numerical (binned) data. The main value of MCA is that it enables us to have attributes, attribute values, and observations all in the same projection. In turn, this allows linking attributes with observations, which helps explaining the meaning of projected observations. This addresses one problem of MDS-like plots.

For the correlating values we also presented a clustering technique which is based on the perceptual space, rather than the data space. This technique is particularly suited for visualization because it clusters values which are perceptually close to each other. Hence, it reduces the visual clutter and, in our context, forms meaningful clusters of related concepts.

The *projection legends* allow seeing which attributes contribute to the x and y projection axes; which are weakly reflected in the projection; and how values of a selected attribute map to projected observations. Understanding the meaning of a scatterplot and/or its clusters requires much less user interaction (clicking a few attribute bars in the *projection legend*) than in classical

MDS plots where one usually has to cycle through all attributes and color projections based on the selected attribute.

Scalability is covered at several levels: Space-filling Voronoi cells show relative locations and distances of attributes and also which observations most likely sample these. Cell merging, done distance-based or by attribute values, removes understood or uninteresting observations to give more space to project the remaining ones. Computational scalability is good: MCA is $O(J^2I)$ for J distinct attribute values and I observations (Sec. 5.2.2), under the assumption that $J < I$.

Limitations

Several limitations exist, though, as follows.

Colors: The categorical colormap scheme used for the *projections view* and *dimensions view* cannot show more than roughly 10 distinct attributes. Even though the problem is alleviated by using colors to emphasize the most relevant attributes, i.e. the ones which are most likely to discriminate between observations, and also by merging cells, the issue still exists. A general solution that can handle datasets having hundreds of attributes, out of which a large subset could be equally relevant, is still required.

Voronoi cell size: Voronoi cells partition the 2D plot space to place multivariate information atop projections in a non-overlapping manner. As a by-product, outliers (e.g. at the plot periphery) get large cells. Cell area is, thus, a by-product of inter-projection distance, and does not encode data values. Although large cells help locating outlier attribute values, the strong visual salience of area can have undesired effects, e.g. users comparing the areas of two cells to draw wrong conclusions about their attribute values. A related issue is the Voronoi cell adjacency: The fact that two cells are adjacent does not carry any additional information besides the fact that they are spatially close, i.e. that observations tend to select their respective attribute values together, as explained in Sec. 5.2.4.

Observations *vs* cells: A separate challenge relates to interpreting observations *vs* Voronoi cells in the *observations plot*. As mentioned in Sec. 5.2.5, if an observation $\mathbf{x}$ is closer to an attribute value j than to other attribute values $k \neq j$, then $\mathbf{x}$ will more likely have value j than values k relative to the other observations. Thus, if $\mathbf{x}$ falls within the Voronoi cell of some attribute value j, it only means that $\mathbf{x}$ will *more likely* have value j than other attribute values. Cell borders are thus only indicators of a change in attribute-value likelihood for observations, and not a precise indication of actual attribute values for observations. Hence, observations that fall within a large cell and are far from the cell borders are very likely to actually *have* the attribute value of that cell. In contrast, for observations that fall close to cell borders, or are located in small, densely packed, cells, we can only say that they more likely take *one* of the attribute values of the respective cells than values of far-away cells. This interpretation challenge is clearly not trivial, and a recognized limitation of our visual encoding.

Usability: Although linking our views to concepts and questions from the application domain is arguably easier than for existing MDS plots, there is still some effort and learning curve required. Making the mapping between questions and views even simpler and more explicit is a main point for future work. Also, investigating the use of MDS techniques, e.g. [110] instead of our current MCA technique, would extend the scope of our explanatory visualizations to a larger area.

Possible expansions

We developed the techniques above in particular for MCA, which is a dimensionality reduction technique for categorical data. Now, MCA is mathematically based on Singular Value Decomposition (SVD), meaning that our methods are at least partially suitable for all other SVD based methods, such as PCA, CA and classical MDS, as well. To adapt our techniques to other dimensionality reduction techniques, we distinguish the following issues.

Categorical versus numerical data: As we have details in Sec. 5.2.2, each observation transformed into a bit pattern, making the data numerical. As will be explained in more detail in Sec. 6.3.2, what one is basically comparing, when two observations are compared, are frequency profiles. Thus there are two numerical issues which must be taken in account: the explosion of dimensions, recall that each categorical attribute results in as many columns as it has distinct values, and the comparison of frequency profiles. MCA rescales the variances to account for the additional dimensions and uses the chi-square distance metric as opposed to the more common distance metrics such as the Euclidean distance. Because each categorical attribute is expanded into many dimensions, the projection gives information about the structure at the level of *attribute values*. This is not the case for numerical techniques, such as PCA, which will give structural information only at the attribute level. Other than that, MCA is conceptually not different from other techniques and therefore our techniques should map to numerical techniques as well.

Projections: Our *dimensions view*, shows the projected observations or attributes only for the first two factors. This is a design choice we made in our particular business context. It is not hard to see though, that this view can be extended to three dimensions. This would lead to the typical problems of 3D projections on 2D screens related to occlusion. Also, it would require adaption of our merging algorithm to correctly merge point in 3D space as opposed to our 2D merging approach. In a 3D projection it would be natural to also provide interactive means to rotate the point cloud. A rotation of the projected point cloud should of course be reflected in the legends as well.

Legends: Performing aMCA returns, among other pieces of information, for both projections the contributions of either the observations or the attribute values to the factors. We use the contributions of attribute values to the factors of the attribute projections in the *dimensions view*, to construct the legends. In all generality dimensionality reduction techniques could be defined as a function that maps a point from a high-dimensional space to a lower dimensional space. Therefore, even though not always trivial, it is theoretically always possible to retrieve the contribution of a high-dimensional point to each dimension of the projection space. Thus in theory, the legends can be constructed for all other techniques as well.

Labels: We use labels in the attribute plot and clip by the cell boundaries to avoid overlap and show the full label in a tooltip. These labels consist of an *attribute* and a *value* part. This gives a fine granularity with respect to the correlations in the data which is not possible with numerical data. In the latter case the label would just consist of the attribute name and a finer level of structure cannot be reached, though this is inherent to the dimensionality reduction technique. In the case of categorical data labels could be compressed in various ways, both with and without data loss, to reduce the required screen space for the label. Some compression techniques are proposed in Sec. 6.4.

5.3 Rule Triggering Analysis

The above presented methods support the exploration of relations between concepts (**DS1**) using the instance data (**DS2**) of the taken decisions. So far, we only explored the insight that could be extracted by MCA on our decision model. Correlations between input concepts are, however, dictated by reality, e.g. students tend to be younger on average; or mid-aged people have a higher chance of being married. Correlations between decision concepts and input concepts on the other hand, result from the executed business logic expressed in the production rules. This distinction pertains precisely to our core question: How can we examine how the reality (as captured by the decision model) *diverges* from the reality encoded by the actual decisions taken? We next approach this question from the rule triggering perspective.

As discussed before, Baudel and van Ham [174] took the following approach: given a rule what are the interesting attributes? This line of reasoning follows a logic to data path, given a rule what are interesting patterns/trends/outliers in the data. From a business perspective, the decisions being made over time are determining the business performance. Therefore, supporting an exploratory approach should start from the decision outcomes, i.e. what are the interesting rules for business cases that result in decision X. For example in our insurance case, what rules are interesting for those business cases that resulted in manual processing?

5.3.1 What are interesting rules?

To answer the question of which rules are interesting, we first need to discuss the metrics that can be extracted from rule triggering. When we look at *all* past decisions, we can calculate various metrics:

- **Absolute rule trigger count**: Rule X is triggered N times.
- **Proportional rule trigger count**: Of all rule triggers, $N\%$ is rule X.
- **Decision rule triggering ratio**: Rule X is triggered for $N\%$ of the decisions.

A decision execution trace contains all rules, in the order that they where fired, that led to the final outcome. The **absolute rule trigger count** is simply calculated by iterating over all traces and incrementing a counter for each rule that is part of the trace. Once the absolute count for each rule is determined, the **proportional rule trigger count** is calculated by dividing the trigger count of each rule by the overall trigger count. The **decision rule triggering ratio** is based on the observation that a particular rule can be triggered multiple times for the same decision. It is calculated by incrementing a counter for each rule, each time it is encountered for the first time in a decision.

In general, absolute counts are mostly not an interesting measure. When millions of decisions are taken, absolute counts end up being large numbers which are hard to comprehend and to compare. Therefore, initially the proportional trigger count is a more appropriate measure to observe. It still begs the question, what is interesting? Is it interesting that a given rule is triggered often or triggered very little or not at all? The answers on these questions is: it depends, either on the formulation of the rules or on contextual knowledge.

Let us suppose we have a simple DM to determine if a person is eligible for a car insurance or not. This hypothetical model is determined by some simple rules:

```
IF the age of the person is less than 18
THEN set the status of the person to ineligible

IF the age of the person is at least 18
THEN set the status of the person to eligible

IF the person has a driving license
THEN set the status of the person to eligible

IF the person was involved in more than four accidents
THEN set the status of the person to ineligible
```

Now, when we would look at the rule trigger counts *only* for the decisions that have the ineligible outcome, it should not come as a surprise that the first rule takes a relevant proportion. Similarly, it is also not a surprise (nor interesting) that the second and the third rule are not triggered at all for the ineligible decisions. Thus, the absolute count nor the proportional count are interesting in these cases, they state the obvious. Rules that directly relate to the decision by the fact that there action results in the decision could therefore be considered unimportant.

The fourth rule is a different case. Let us assume that this rule is only triggered if the person is above 18, i.e. there is no need to fire this rule if the person is already ineligible due to his age. When this rule would take a large proportion in the overall trigger counts for ineligible decisions, it means that many persons that otherwise might have gotten an insurance and therefore become a potential risk have now been turned down. However, if in the more likely case, this rule was triggered never or only a couple of times it would be interesting as well. It means that people involved in four or more accidents are the exception. From a business perspective this means that the chosen limit, i.e. four accidents, might be too high in this case. When we look at a subset of the decisions, such as only the ineligible decisions, we can calculate additional metrics:

- **Absolute rule trigger count**: Rule X is triggered N times for decision outcome Y.
- **Proportional rule trigger count**: Of all rule triggers for decision outcome Y, $N\%$ is rule X.
- **Decision rule triggering ratio**: Rule X is triggered for $N\%$ of the decisions for decision outcome Y.
- **Differences**: For each of the above measures the difference between the value for this subset and the overall set.

Finally, there are some cases where the absolute count is interesting, namely when the trigger count is low. When a rule is triggered only a couple of times or so little that the proportion rounds to 0.0%, the rule is actually representing corner cases. Being able to identify that these corner cases occur can help identifying problems with the DM.

5.3.2 Generalizing the problem

Until now the discussion about analyzing rule trigger counts has been revolving around the overall set of decisions and subsets of decisions that represent a certain decision outcome. However, we can generalize this approach due to the observation that the decision outcome is only one of the many possible attributes that define a business case and the resulting decision. Instead of selecting a subset of decisions based on the decision outcome, we could also select decisions

based on the other attributes. For example, by selecting all decisions for persons that fall in the age category 18..25 and who are full-time student.

As we have seen, the usefulness of the counts (absolute and proportional) and the differences between counts of a selection and the overall set of decisions is depending both on the formulation of the rules and external context. In the generalized case, we can add an optional filter to reduce the information that would be presented to the user. This filtering is similar to the action filtering discussed before. Based on the observation that rules which test on an attribute that is used to make a selection will bias the metrics for these rules. For example, when we have a rule that tests on age and we select decisions based on an age category as well, it is likely that the rule is either:

- **overrepresented**: the proportion of this rule in the selection is significantly higher than its proportion in the overall set of decisions.
- **underrepresented**: the proportion of this rule in the selection is significantly lower than its proportion in the overall set of decisions.

Therefore, removing rules that test for one of the attributes that were used to make the selection of decisions could put the focus on more unexpected patterns. Presenting the results of the analysis is a simple matter of listing the rules in a table with a column for each of the metrics. Column sorting can be used to focus on a particular metric.

5.4 Conclusion

We have presented a set of visualization and analysis techniques that allow for analysis of a DM beyond the limits of the techniques presented in CHAPTER 4. The first technique, presented in Sec. 5.2, describes a visualization and a workflow for exploratory analysis of categorical data. In contrast to classical numerical MDS, we use MCA to create 2D projections which display attributes, attribute values, and observations. We introduce several visual encodings which help correlating values, observations, and observations with values. We showed how our techniques can be used to find non-trivial insights with limited effort in a dataset from the insurance industry. A standard tool in sociology, MCA is rarely used for information visualization of multivariate data. Yet, categorical data is very common in datasets concerning business processes. These visualization and analysis techniques are a solution to address research question **Q3**, about gaining insight in the structure of the data. As the labels can be used to distinguish business case and decision data, we also have an initial solution for question **Q4**.

Secondly, we presented a method for analyzing rule trigger patterns in Sec. 5.3. By extracting various metrics for the overall set of decisions and comparing those with metrics for a selection of decisions interesting differences can pop up that might help the analyst to understand the functioning of the DM. This method serves as a partial solution to research question **Q5**, how do we get insight in the logic for a given set of decisions.

At this point we have just introduced techniques for getting insight in a few aspects of the various data spaces of a DM. However, we have not provided an integrated approach to analyze the aggregated effects of many decisions. In the following chapter we will tie together the presented techniques in order to provide an exploratory environment that answers even closer the main questions stated in CHAPTER 1 and CHAPTER 2. In detail, we will combine the techniques presented in this chapter to provide a work flow which enables combined analysis of correlations in the business case data and of the logic that lead to the decisions for these business cases.

Published as: Decision Exploration Lab: A Visual Analytics Solution for Decision Management
Broeksema, B. and Baudel, T. and Telea, A. C. and Crisafulli, P.
in IEEE Transactions on Visualization and Computer Graphics, 19(12)

Chapter 6

Decision Exploration Lab: an exploratory environment for Decision Management Systems

> A technicistic approach would leave the power to decide to information processors. In a non-technicistic approach, this power stays in the hands of humans. Technique would at most help humans to maintain once made agreements.
>
> *Aan babels stromen*, (Translated from Dutch), ROEL KUIPER

In this chapter we present the Decision Exploration Lab (DEL), an integrated toolset for combined verbal and visual analysis of DMS datasets. Its design and implementation aims to provide direct support for the core tasks related to DMS as outlined in Sec. 2.6.

Currently there are no tools that specifically target the analysis of the functioning of a Decision Management System (DMS). To analyze automated decisions, a cumbersome process is followed which does not take into account the relationships between the three data spaces. In the context of this thesis, we integrate with IBM's Operational Decision Manager (ODM), a software product for managing day-to-day automated enterprise decisions. ODM allows to export either the execution traces or the business case and decision data. The exported data are stored in a data warehouse in a tabular format and then analyzed and monitored with the typical analytical and business intelligence tools such as IBMs SPSS and Cognos or alternatives such as SAS business intelligence, Oracle BI and SAPs Business Objects. Often the resulting analysis is partial, both in the data that is analyzed and in the analyzed relationships between different data spaces.

We integrate the visualization techniques, presented in CHAPTER 5, with IBMs ODM. Controlled natural language [14] plays a major role in IBMs ODM, as it is next to decision tables the main way to express business rules. As a consequence we designed an exploratory system that provides a verbal mode in which the business user can query decisions using controlled natural language and a visual decision exploration mode which serves as an interface for the automated analysis of the decision data.

The goal of integration is to design or construct a system that acts as a coordinated whole. In CHAPTER 5 we presented Visual Analytics (VA) techniques for the data at hand, but we did not detail how these sub-solutions are tied together in an unified workflow. Therefore, in this chapter, we discuss the architecture and details of our toolset. In CHAPTER 7 we next present the application of our system to real-world data-sets along with some visual refinements that took place during integration.

6.1 Architecture

For a good separation of concerns, DEL is designed using a layered architecture as shown in Fig. 6.1. The first layer consists of IBMs ODM which is the software product that is put into use for automating business decisions. At this point decisions are either simulated using historical

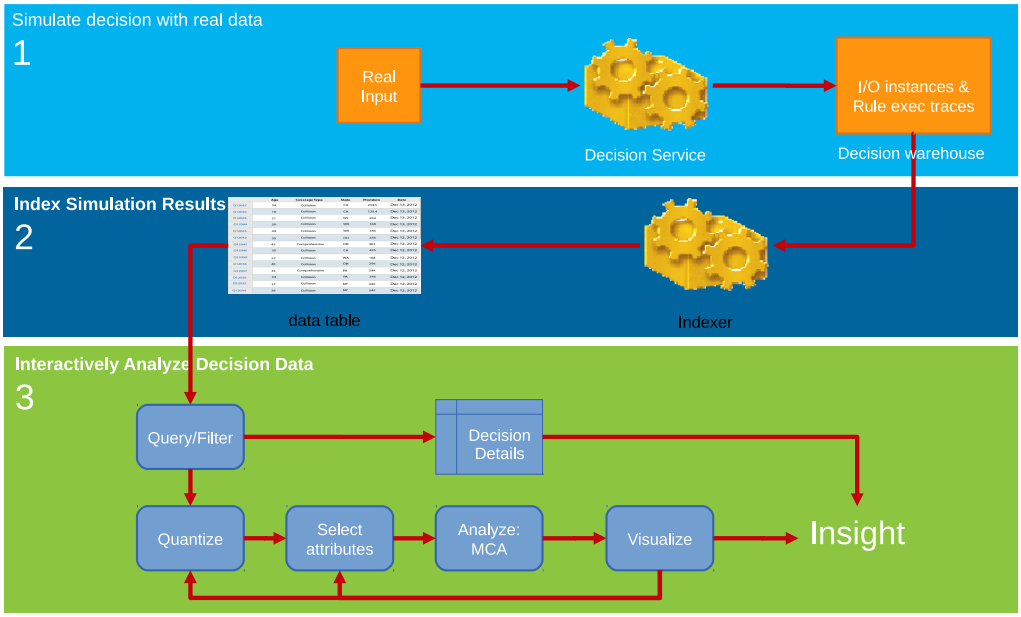


Figure 6.1: Layered architecture of the DEL

business cases, or existing decision data are used. Simulation is needed in cases where the decision itself is not stored or where additional logging such as execution trace logging is turned of. Additionally, normally enterprise critical systems should not be put under additional burden just for analysis purposes. Simulation enables reusing existing business cases, retrieving the original decisions accompanied with the required logging data. The resulting business cases, decisions and execution trace data are stored in a database using an XML format.

In order to be able to query and analyze the decision data efficiently, it must be stored in a suitable format. This is done in the second layer, which reads data from the decision data store and transforms and indexes it. All fields of the business case and of the associated decision become columns in the new data table. One row is added to this table for each decision in the decision warehouse. Additionally, a multivalued column is added which contains the list of triggered rules for each decision. Storing decisions like this enables efficient querying of decisions and therefore also allows us to construct the data structures required for analysis in an efficient manner.

The final layer contains the analytical and visualization components. These components perform the quantization of attributes based on user input and provide the means for exploratory visual analysis of the decision data. This layer itself is based on a client-server architecture. The User Interface (UI) is implemented as a web-client while the analysis and data querying is handled by a web-server. It provides the working modes: verbal (Sec. 6.2) and visual decision exploration (Sec. 6.3). Overall it supports an iterative approach, where data are filtered and attributes are selected for analysis or deselected such that the analyzed data match the level of detail required at any given point in the analysis process.

6.2 Verbal mode

In Sec. 2.6.2 we listed six high-level tasks that should be supported by the system. The two tasks **T2**, create selections of decisions, and **T5**, finding Key Performance Indicator (KPI) related opportunities, can be related to more specific tasks where some prior knowledge can be assumed.

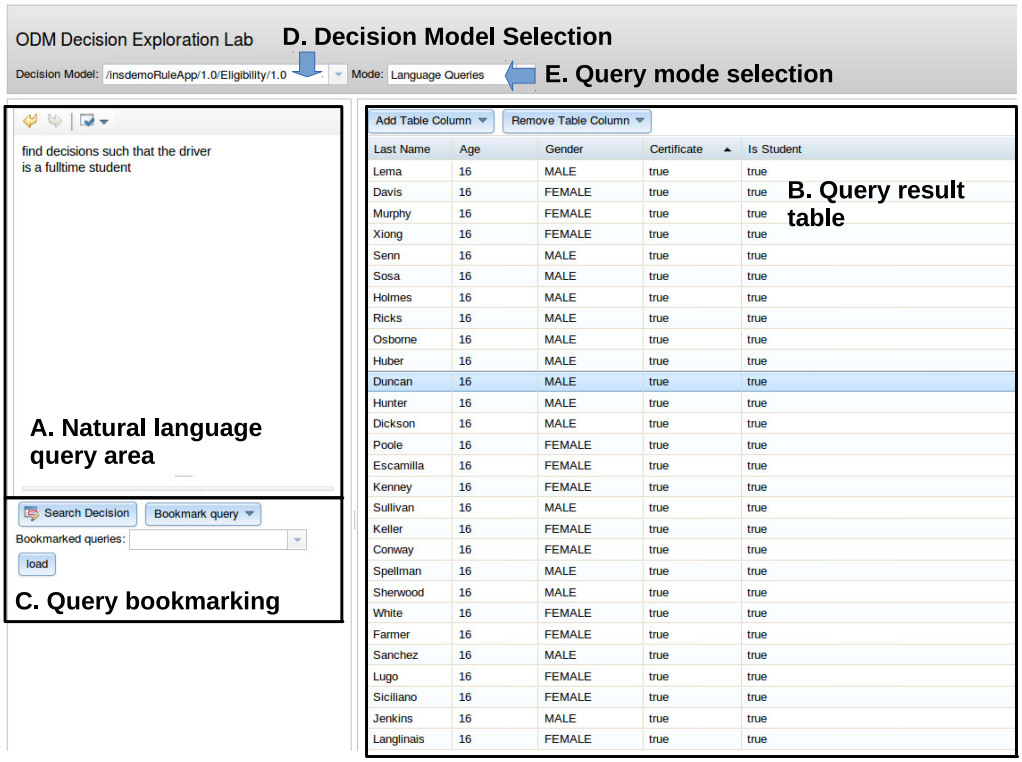
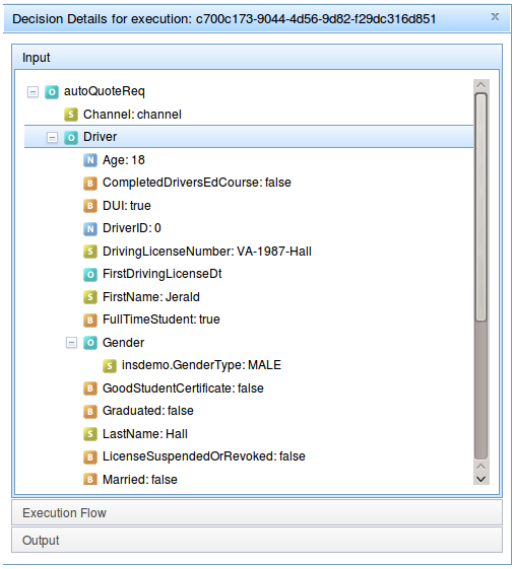


Figure 6.2: The Decision Exploration Lab in verbal mode.

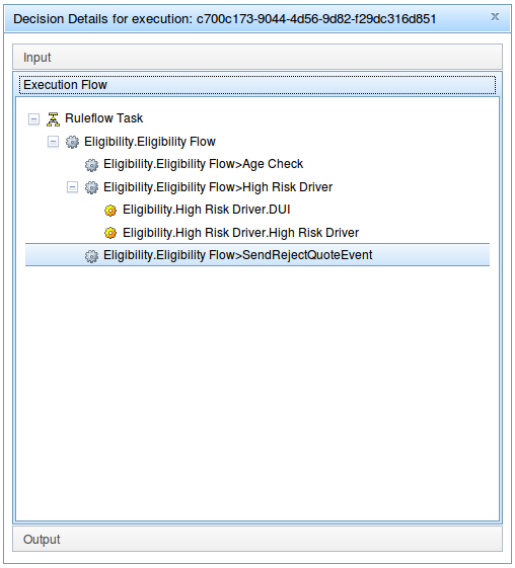
The first of these specific tasks encompasses troubleshooting, which can only be done when at least some basic information that would lead to the problematic decision is available. That is, a customer has called that he was rejected for an insurance, but he does not understand why or agree with the decision. Details provided by the customer, such as name and address details, can be used to find the particular decision. The second concrete task relates to KPI-based improvements of the Decision Model (DM). An analyst knows how the KPI is calculated and therefore also knows which decisions he wants to analyze. He could be interested, for example, in decreasing the number of insurance requests that are processed manually. In that case he would like to analyze at some point only the decisions that had a manual outcome. Both cases require that a user is able to find a particular decision or selection of decisions and to analyze details.

As a first means to this end, we added a *verbal mode* to our system as shown in Fig. 6.2. We call this the verbal mode because the user explores the decision data using controlled natural language (Sec. 2.3) The two combo-boxes at the top allow the user to select which DM he want to analyze (Fig. 6.2D) and which analysis mode he wants to use (Fig. 6.2E). In verbal mode, the main view is a table which lists decision details (Fig. 6.2B). The user can add and remove columns to this table based on the information that the he wants to see.

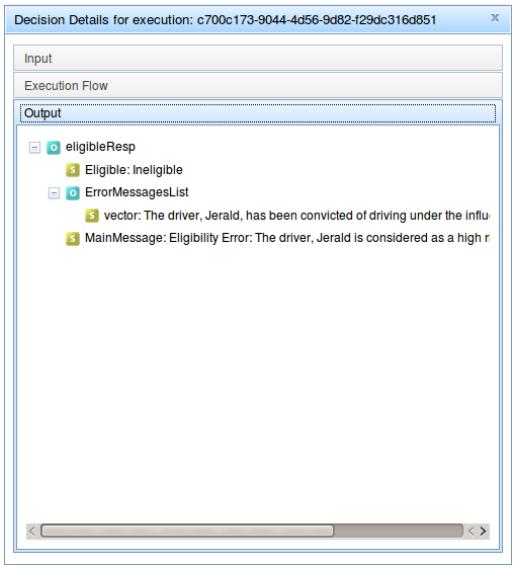
When a user selects a particular decision, a dialog opens in which he can inspect the full details of the decision as shown in Fig. 6.3. In this dialog he can inspect all three parts of a decision execution by clicking the panel title that holds the information he is looking for. The input panel (Fig. 6.3a) contains the original object tree for the business case. It follows the structure of the business case as defined in the ontology and presents the attributes and their corresponding values in a hierarchical manner. The decision outcome is presented in the output panel (Fig. 6.3c) in the same way. Finally, the trace panel (Fig. 6.3b) displays the execution trace. As detailed in Sec. 2.3.2, the trace is a hierarchical structure when the user has defined decision control flows.



a. Details of the business case



b. Triggered control flow tasks and businss rules



c. Details of decision

Figure 6.3: Decision details dialog showing the details of an individual decision: attribute values of the business case (a), triggered decision tasks and business rules (b) and the attribute values of the resulting decision (c).

A flow consists of tasks and each task represents either another decision flow, a decision table or a selection of business rules. Each of these is represented in the tree by their corresponding symbols (a flow chart symbol, a grey and a yellow cog).

To focus on a particular subset of decisions, the user can specify a query (Fig. 6.2A), using controlled natural language. When a query is performed the table is updated and only lists the decisions that match the query. These queries can be bookmarked for later reuse (Fig. 6.2C).

6.2.1 Querying and filtering

Given that the number of decisions can be very large, querying lets the analyst focus on an area of concern, rather than being confronted with too many facts. Business analysts working with DMSs are used to express production rules in controlled natural language [14]. We extended this approach to allow for queries on decisions that are understandable for a business user. In the query field, the user can express queries on decisions to search for decisions based on the checked preconditions:

```
find decisions such that
the vehicle has antilock brakes
```

That is, in this particular scenario, the fact that a vehicle has anti-lock brakes is modeled as being part of the business case. Because anti-lock brakes is a property of a business case and not of the resulting decision, this property is only checked in the preconditions of a rule and not altered in the action part of a rule. On the other hand, queries can also check for actions taken by the production rules:

```
find decisions such that
the quote has at least one 5% discount
```

That is, discounts are added to a quote by certain production rules. Thus when querying for such facts, decisions are selected based on actions taken by the DM.

This approach is usable for simple queries on the decision warehouse when users know what to ask for. A typical use-case is auditing or troubleshooting a decision which can be reasonably easy identified by some known criteria. These criteria typically come from a customer who calls, providing e.g. name, surname and order id, or from an auditor who wants to see decisions for a particular population, e.g. females who have been rejected. A second use-case is creating selections of decision for monitoring or validation (**T2**).

Additionally, we use the querying mechanism as a preliminary filter for the visual exploration mode. The query is used to reduce the data under analysis in visual exploration mode. This lets the user reduce the set of decision instances in a way that is most suitable for the kind of questions he wants to answer. One can think of many starting points for an analysis that lend themselves for this approach, such as:

- persons between 18 and 25;
- decisions that involve a car of brand Peugeot or Citroën;
- decisions that have more than two discount items;
- decisions that have a quote lower than $ 150.

When the analyst wants to gain a deeper understanding of a selection of decisions, he enters the query selecting these decisions in the query field. Next, the visual decision exploration mode will restrict the analysis to the decisions selected by the current query. This way he can really dive into the details of a very specific population which is important for finding targeted improvements or new business opportunities (**T5**).

6.3 Visual decision exploration

The verbal mode does not help the analyst in fully understanding the functioning of a DM. With possibly millions of decisions it is also hard to give a meaningful "overview first", following the Information Visualization (InfoVis) mantra by Shneiderman [186]: "Overview first, zoom/filter,

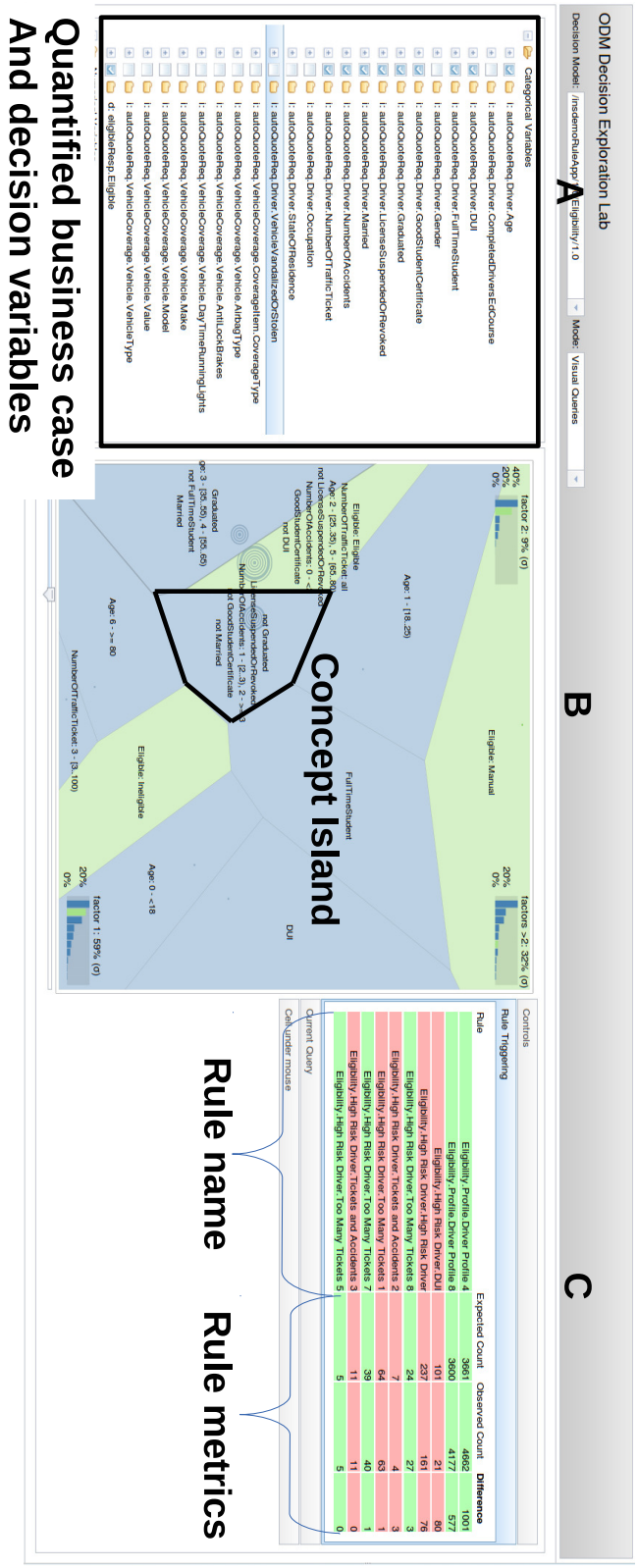


Figure 6.4: The Decision Exploration Lab in visual decision exploration mode, with the *dimensions view* (A), the *decision map* (B), and the *rule trigger view* (C).

details on demand". Therefore we designed an additional approach which we call the visual decision exploration mode and which we describe in this section. The design of this mode follows the extended guide proposed by Keim et al. [187]: "Analyze first, show the important, zoom/-filter, analyze further, details on demand". It integrates the VA techniques for categorical data presented in Sec. 5.2.2 and rule triggering analysis from Sec. 5.3. The visual decision exploration mode of the DEL (Fig. 6.4) helps finding two kinds of relations:

1. Relations between attribute values of business cases and decisions.
2. Relations between (groups of) attribute values and the decision logic.

As starting point for our exploratory approach, we chose the first: relations between attribute values of business cases and decisions. This mirrors the construction of a DM, which also has to start with modeling the business cases and the decisions. That is, one can only reason about something (e.g. a person, a car or a transaction) when this something is defined in the first place. Now, we can break down our exploratory process following the mantra by Keim et al.:

- **Analyze first:** It is natural to think about business cases and decisions in related concepts and attributes. We apply Multiple Correspondence Analysis (MCA) to find such relations based on past decisions.
- **Show the important:** Related concepts of the DM, found by the MCA are visualized in the *decision map* (Fig. 6.4B) and grouped together to show the important clusters of related values. Examples that show up in our scenario:
  - Persons which are student tend to be younger and unmarried.
  - Expensive cars tend to have both anti-lock brakes and multiple airbags
  - Ineligible persons tend to be over eighty or have a high number of tickets.
- **Filter:** To gain a deeper understanding of a subset of decisions, the verbal mode (Fig. 6.2) can be used to filter unrelated decisions. Additionally, by using brushing concept islands in the visual decision exploration mode, the set of decisions can be filtered even further to focus on a very specific subset of decisions.
- **Analyze further:** Filtering the decisions using verbal mode, triggers MCA anew and results in a more detailed analysis of the decisions of interest in visual decision exploration mode.
- **Details on demand:** In verbal mode individual decisions can be inspected to retrieve details (Fig. 6.3).

However, this still does not incorporate the business logic. To this extend we integrate the rule trigger analysis as follows. In Sec. 5.3 we described how various metrics can be calculated for the overall set of decisions and subsets of decisions. We did not detail though, how to retrieve subsets of decisions that could lead to meaningful insights in our integrated approach. Such subsets are created interactively by combining the current query in verbal mode with brushing in visual decision exploration mode. For example, the analyst could first reduce the decisions by focusing on students. Next, in the visual decision exploration mode he finds that students who are married tend to have a lower number of speeding tickets. By brushing the *Married: true* and *Number of tickets: low* values, the decisions will be further filtered and the rule triggering metrics will now be calculated for students (by the query which is specified in the verbal mode) who are married and have a low number of tickets (by brushing these values in visual decision exploration mode).

The DEL has three main components in visual decision exploration mode: The *dimensions view* (Fig. 6.4A) shows all Business case and Decision attributes. The *decision map* (Fig. 6.4B) helps understanding correlations between these attributes. The *rule trigger view* (Fig. 6.4C) shows how production rules map on actual executions. To implement these components, we build upon the techniques presented in CHAPTER 5. However, the techniques presented in Sec. 5.2.2 have been initially introduced in the general context of analyzing multivariate categorical data. We adapted and extended these techniques to our more specific context of analyzing ontologies, rule sets, and the execution thereof, as follows.

6.3.1 Dimensions view

To support analysis on ontology instances from the decision model, i.e. business cases and decisions, we reuse and refine the *dimensions view* from Sec. 5.2.5, as follows (see also Fig. 6.4A). Each ontology attribute of basic data type (e.g. boolean, string, numeric) becomes a tree node. Parent nodes are created following the object hierarchy. For example, the input parameter *InsuranceRequest* has a property of type *Person*, which in turn has a numeric attribute *age*. This leads to a branch $input.InsuranceRequest.Person.age$ in our tree.

We separate variables in three sub-trees: categorical, numerical and temporal. We added check-boxes to categorical variables so that they now can be added or removed from the analysis interactively, as opposed to our earlier approach where all categorical variables where always taken in account. All variables together define the data space that comprises our decision instances. Numerical and temporal variables can be binned into categorical variables. This way, they can be directly handled by our underlying decision-map analysis which uses the MCA technique (Sec. 5.2.2). When binning numerical variables, users can choose the number of bins to match the precision needed for targeted questions. Temporal variables can be binned with various time spans such as year, quarter, month or day. Binning a variable can be done as often as needed, until the binned variable meets the expectations or needs of the analyst.

To each categorical variable node, we finally add, as leafs, all possible values that the respective category can take. For numerical and temporal variables, we display statistics (minimum, maximum, average, and standard deviation) to inform the analyst about the basic properties of these variables. For each value of a categorical variable, a bar is shown in the tree, representing the number of times this value was encountered in the dataset. The bars form together a histogram of the variable (see Figs. 6.4 and 7.2).

6.3.2 Analyzing Categorical Data

As outlined above, we reduce the inherently hierarchical structure of the ontology of a DM to a tabular structure. Using the *dimensions view*, the analyst can quantize numerical attributes in a way that fits his expectations or the kind of questions he wants to answer. Quantizing an attribute results in a new ordinal attribute that shows up in the *dimensions view*. The analyst can next select attributes from the categorical subtree in order to start the analysis process. Each time an attribute is selected or deselected, the analysis is performed anew.

Our initial approach, presented in Sec. 5.2.2, was based on the indicator matrix. Recall, that in this matrix each business case and decision was represented by one row. Additionally, each possible value of each of the attributes becomes a column in this matrix. When a certain value of an attribute is selected by the business case, the matrix cell matching the case and the value is set to one and zero otherwise. It is easy to see that the size of this matrix is bounded by the number of business cases as this will grow over time, while the number of distinct attribute values is fixed.

	Car.Type			Car.Value			Car.Airbag		
	Sedan	SUV	Lux.	$< 2.5K$	$2.5K - 5K$	$> 5K$	None	D	D + P
Sedan	1500	0	0	1200	200	100	400	1000	100
SUV	0	1000	0	300	500	200	250	600	150
Lux.	0	0	400	20	100	280	0	50	350
$< 2.5K$	1200	300	20	1520	0	0	340	865	315
$2.5K - 5K$	200	500	100	0	800	0	180	455	165
$> 5K$	100	200	280	0	0	580	130	330	120
None	400	250	0	340	865	315	650	0	0
D	1000	600	50	180	455	165	0	1650	0
D + P	10	150	350	130	330	120	0	0	600

Table 6.1: An example Burt matrix for three variables.

As detailed in Sec. 5.2.2, one of the steps MCA comprises a Singular Value Decomposition (SVD) which decomposes a matrix $\mathbf{A}$ into three matrices (Eq. 5.3). For the visual approach presented in CHAPTER 5 all three matrices are required. As discussed by Golub et al., [188, p. 254], the complexity of this operation for a matrix $\mathbf{A}_{mn}$, with $m \gg n$, is $\mathrm{O}\left(4m^2n + 22n^3\right)$. Thus, in order to be able to perform real-time analysis we must reduce the size of this matrix.

MCA can be thought of as a weighted Principal Component Analysis (PCA) on either rows or columns [121]. It treats rows (observations) and columns (variables) symmetrically and thus allows analysis of both instances and variables. Besides the computational issues as pointed out above, when having millions of observations, it is questionable if the scatter plot approach works without applying techniques such as sampling [189] or splatting [190]. As a consequence, another approach is taken that only gives the analysis for the columns (attribute values) of the indicator matrix. To support interactive, real-time visual analysis, we thus base our framework on the Burt matrix $\mathbf{B}$, which is defined as

$$\mathbf{B} = \mathbf{X}^\top \times \mathbf{X} \tag{6.1}$$

where $\mathbf{X}$ is the indicator matrix. Applying Eq. 6.1 to the earlier example of an indicator matrix in Table 5.5, results in the Burt matrix as shown in Table 6.3.2. In this matrix each sub-matrix (identified by the borders in the table) has the same sum, 2900 in this case. This sum represents the number of observations, thus number of rows in the original indicator matrix $\mathbf{X}$. Each row, and therefore each column because the matrix is symmetric, represents the frequency profile of a particular value. For example, the first row represents Sedan cars, of which there are 1500 in the dataset. From these 1500 cars, 1200 have a value below $2500 and 100 have airbags for driver and passenger (D + P).

Using the Burt matrix for analysis as opposed to the indicator matrix brings various computational advantages, making it possible to perform real-time analysis of large sets of decisions. Firstly, the Burt matrix is a symmetric square matrix whose rows and columns are the attribute values. As a result, the size of the matrix is bounded by the number of *distinct* variable values and is independent from the number of decisions. In our context, this number is several orders of magnitude smaller than the number of decisions. For example, the eligibility DM, from the car insurance scenario (Sec. 2.2), contains 23 attributes with a total of 158 distinct values, while it was used to perform 100000 decisions. Secondly, the Burt matrix itself can be constructed efficiently by performing J queries on the data table constructed by the indexer, where J is the total number of distinct values. These queries are called faceted queries and are supported by search platforms such as Apache Solr [191]. Finally, because the Burt matrix is symmetric it is no longer needed to

compute the full SVD. It is now sufficient to compute only $\mathbf{P}$ and Δ, which is $\mathrm{O}\left(2mn^2 + 11n^3\right)$ [188, p. 254], with $m == n$ and $m_{burt} \ll m_{Indicator}$.

As before, MCA results in three pieces of information for analysis:

1. The projection of the original data points on the factors f_i, or eigenvectors of the Burt matrix. These are sorted by explained variance, i.e. f_1 explains most of the variance, f_2 explains second most of the variance, and so on.

2. The amount of variance explained by each factor f_i.

3. The contribution of each attribute to a factor, or how much of the variance of a given factor is explained by a certain attribute.

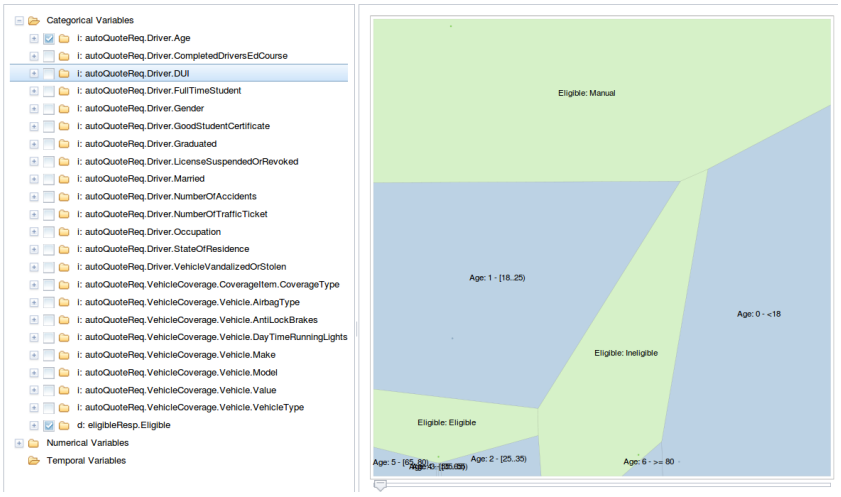
6.3.3 Decision map

The *Decision map* is a dynamic map analogous to the one presented by Zizi et al. [154]. However, while Zizi et al. assign space to areas in the map based on instances, *Decision map* maps data attributes. It follows the classical scatterplot technique used for MDS (see Fig. 6.4B): We take the two factors f_1 and f_2 along which the data has most variance, and plot all attribute value projections, i.e. factor scores, along f_1 and f_2. Hence, the f_1 and f_2 factors correspond to the x and y axes of our 2D scatterplot. Similarity between attribute values is reflected by proximity between 2D plot points. In other words, we use in the *Decision map* the same technique as in the *projections view* from Sec. 5.2.5, but now we make its semantics more explicit and thus more useful by linking the meaning of Voronoi cells with domain specific concepts. We see that the techniques from CHAPTER 5 become less abstract and therefore more usable when we apply them in a particular application domain.

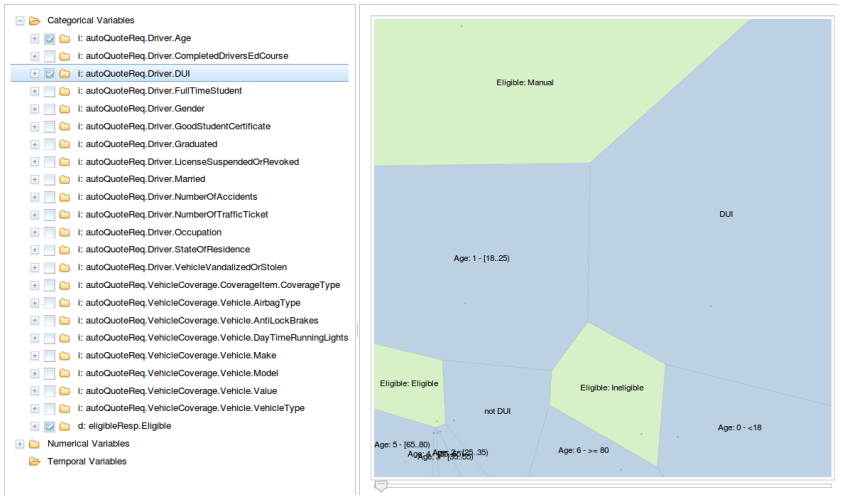
MCA, like similar techniques such as Multi Dimensional Scaling (MDS) and PCA, tries to mirror the projection (2D) distance with the distance in the original data space. By design, MCA uses the chi-square distance metric. This metric is based on relative frequencies of variables, adjusted by the contribution of an attribute to the average instance. Each of the resulting factors is a combination of the variables used in the analysis. When most of the variance is explained by the first two factors f_1 and f_2, MCA is good for exposing the original data structure: Proximity between projected attribute values means that those values are correlated. Additionally, contributions of each attribute can be calculated, to explain how much the x and y plot axes are determined by a certain attribute.

However, the complexity of the chi-square distance metric makes the interpretation of the x and y plot axes hard for business analysts. As such, we leave these axes out, and use the 2D scatterplot points as sites for a Voronoi diagram. Due to the strong relation with concepts from the ontology, we name Voronoi diagram cells *concept islands*. A concept island containing one concept represents all decisions that have this concept. Islands with more than one concept represent a set of decisions that have *all* these concepts.

As mentioned, each factor explains part of the variance in the data and is a combination of the original variables. To make the *Decision map* easier to read, we add three bar plots to it (for f_1, f_2, and for all factors $f_{i>2}$). Bars in the first two plots show the contribution of each attribute to the x (f_1) and y (f_2) axes respectively. Bars in the third plot show the contributions of all variables which have *not* been captured by the plot, i.e. contributions captured by the factors $f_{i>2}$. Contributions in each plot, i.e. bar lengths, are sorted decreasingly, so we can locate the most important variables that map to the x and y axes or which are not captured by the 2D plot at all.



a.



b.

c.

Figure 6.5: Interactive session with the *decision map*. First the analyst selects age and eligibility (a). Next he adds Driving Under Influence (DUI) (b). Finally he adds number of accidents and number of tickets and uses merging to get meaningful clusters (c).

6.3.4 Coloring

The MCA visualization presented in Sec. 5.2.5 uses a categorical color map to show the identity of the mapped variables. This technique was initially designed as a generic tool for exploratory analysis of categorical data and presented as such [192]. Applying these techniques to the specific context of analyzing decision data exposes various problems, among which the used coloring scheme.

Even though, the usefulness of this color scheme scored relatively high (see our evaluation described further in APPENDIX A.4) in the preliminary evaluation, there is a number of problems with this approach. It uses a fixed amount of ten colors to map identity of attributes. Ten is already a high number as early research found that out of 210 different colors only eight colors and white could be named consistently [193]. When using color for labeling categories, estimates for the optimal number of colors vary between five and ten [194]. Additionally, when there are more than ten attributes, cycling is used mapping the same or very similar colors to different attributes. This forces users to perform additional interaction to find out which attribute(s) are represented by a concept island.

We thus further refine the *projections view* from Sec. 5.2.5 in the context of decision management, by using a two-color map. This leads to two advantages. First and foremost, it allows us to show the relation between business case concepts and decision concepts. More generally, we can now see dependent *vs* independent variables, where the user determines which variables are of which type. Secondly, we avoid all above mentioned color mapping problems which occur when color maps attribute identity. The *decision map* in Fig. 6.7 shows how the three values (eligible, manual and ineligible) of the decision concept *eligibility* (green concept islands) relate with respect to values of input concepts (blue concept islands).

6.3.5 Interaction

We support real-time exploration and filtering of decisions by several interaction techniques. First, we use the *dimensions view* to select the variables of interest, by (un)checking their respective checkboxes in this view (Fig. 6.4A). When variables are added or removed, MCA is performed anew on the selected variables and the *decision map* is updated. Given the efficient computation of the Burt matrix (Sec. 6.3.2), this process works in real-time even for large datasets. The way of working is simple: An analyst starts with a selection of variables of interest determined by e.g. textual search (Sec. 6.2.1). From this selection, variables which do not contribute to the structure of the data (e.g. show up having small contribution in the *projection legends*) are next iteratively removed. This yields a *decision map* where real data correlations become more visible.

Secondly, we provide a *merge slider*, which merges (see Sec. 5.2.5 for details) concept islands that are close together based on the distance configured by the slider. Merging results in new concept islands that represent not one value, but a selection of correlated values. When a decision concept is merged with an input concept, e.g. like the *eligible* concept island in Fig. 6.7, the concept island will be colored green as well. Merging helps users to segment the decision instances based on correlated properties.

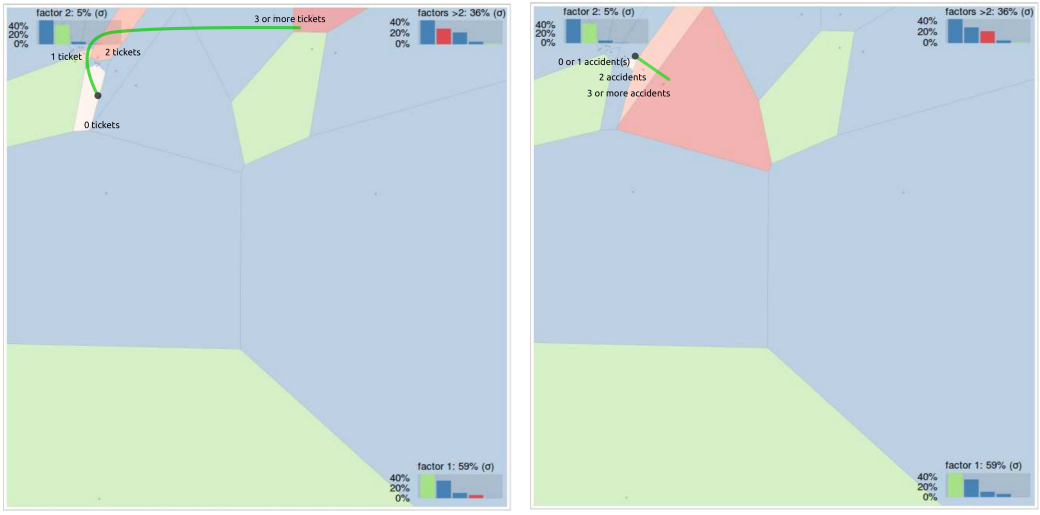
These first two techniques can be seen in action in Fig. 6.5, we disabled the *projection legends* for clarity. In this small session the analyst wanted to learn about the relationship between age, eligibility and risk factors: number of tickets, number of accidents and Driving Under Influence (DUI). Initially he selects age and eligibility which results in the *decision map* as shown in Fig. 6.5a. He continues to add DUI and the map is updated as shown in Fig. 6.5b. Finally, he adds the ”number of tickets” attribute and the ”number of accidents” attribute to the analysis as well. He notices that a lot of labels overlap in the upper part of the screen and uses the merge slider to merge closely projected values. This results in the map as shown in Fig. 6.5c, where he now

sees meaningful clusters at the top. Top-left he finds eligible people, who fall mainly in the age category $25..80$, were not caught for DUI and have a low number of accidents. Top-right he finds the ineligible people, which is most strongly related to persons below 18. Other values in this area are *more than three traffic tickets* and *persons above eighty*.

Thirdly, we add interaction to the *projection legends* to enhance understanding of the *decision map*. When a bar, representing a variable, is hovered or brushed in one *projection legend*, bars in the other *projection legends* representing the same variable are highlighted. For example, in Fig. 6.6a the user brushed the bar representing number of tickets, which is now highlighted in red in all legend plots. This helps users to see how much a given variable is explained along the x and y axes, and also how much is not explained by the plot at all. Thus, in Fig. 6.6a, the user sees that number of tickets is only somewhat represented in the x-axis (about 5 percent) and almost not at all in the y-axis. And, unsurprisingly, it has a large share in the error-legend.

At the same time, all concept islands that belong to the hovered variable are colored using a gradient color scheme based on ColorBrewer [195]. This interaction feature is demonstrated in Fig. 6.6, where the analyst wants to find out the relation between number of tickets or accidents and eligibility. He first selects the number of tickets by clicking the appropriate bar in the legend which updates the *decision map* as shown in Fig. 6.6a. From the map, he quickly sees that highest number of tickets (darkest value) is closely projected to the ineligible decision. The other values are roughly equally close projected to the eligible decision. When selecting the *number of accidents* (Fig. 6.6b), he sees that the *number of tickets* is increasing when moving farther away from the *eligible* decision, resulting in a gradient in the top left quarter of the figure, marked by a green line. Looking at the *projection legends* however, he learns that both attributes (*number of tickets* and *number of accidents*) do not explain much of the variance for either the first nor the second factor. This comes as a surprise to him as he would have expected that the DM would enforce a stronger correlation between *number of accidents* or *number of tickets* and the *eligibility* of a person.

Finally, each concept island can be hovered and brushed. At hovering, a detail panel, not shown in the figure, is updated to show the full name of this island. This way, we can show partial labels or no labels for small concept islands and concept islands that contain multiple values in



a. b.

Figure 6.6: Interacting with the *projection legends* to see the spread of values. The green lines are added manually to emphasize how the values are spread in the *decision map*, lowest value at the dot. (a) The number of tickets is selected. (b) The number of accidents is selected.

the *decision map* to avoid clutter. Brushing implements dynamic queries [24] on the decisions by selecting one or multiple concept islands. By default, all decisions are shown in the decision table. When brushing concept islands, decisions are filtered based on the values represented by the brushed concept islands. This allows for creating selections of decisions (**T2**) based on correlations found in the data (e.g. mid-aged, married people). Brushing allows discovering correspondences between the decision instances (**DS2**) and the decision logic (**DS3**). In other words, brushing the concept islands provides a graphical way to perform queries on a dataset, based on individuals corresponding to specific concepts extracted from the dataset. This is in contrast to most brushing techniques which provide detail-on-demand or show the same data in different (linked) views.

6.3.6 Rule trigger view

The *decision map* helps the exploration of relations between concepts (**DS1**) using the instance data (**DS2**) of the taken decisions. So far, we only explored the insight that could be extracted by MCA on our DM, that is correlation between attribute values. We can distinguish three kinds of correlations:

- between values of business case attributes;
- between values of business case attributes and decision attributes;
- between values of decision attributes.

In the first case, correlations between values of business case attributes, the analyst finds correlations that are dictated by reality. For example, students tend to be younger on average; or mid-aged people have a higher chance of being married. In the second case, when values of business case attributes correlate to values of decision attributes, the analyst found a correlation that is a result of the business logic. For example, when the DM contains a rule that states:

```
IF the age of the person is below 18
THEN set the status to ineligible
```

all persons below 18 will be marked as ineligible. This results in a projection of those two values close together such as shown in the top-right corner of Fig. 6.5c. Sometimes business rules will interplay or are formulated in a more advanced manner making these relationships less clear. For example when we have the rule:

```
IF the number of traffic tickets is greater than 2
   AND the number of accidents is at least 1
THEN set the status to ineligible
```

then the correlation between the decision attribute value “ineligible” and a high number of traffic tickets might be less clear, an effect that was shown in Fig. 6.6. Clearly, rules like this formulate the expectations of an enterprise with respect to number of tickets, number of accidents and expected risk. If the analyst was to find that there are a lot of persons with two traffic tickets and at least one accident, the above rule should perhaps be reformulated.

In the case where two decision attribute values are projected close together, we can have two sub-cases. First, values of the same decision attribute are projected close together. Following the general MCA explanation, this means that the observations that trigger either one of these decision values, are similar with respect to their other attributes. For example, when both the “manual” and the “ineligible” decision outcomes would be projected close together, then apparently there are no clear rules to separate the two. An analyst would need to gain a better understanding of the observations that trigger those two decision outcomes. This should allow

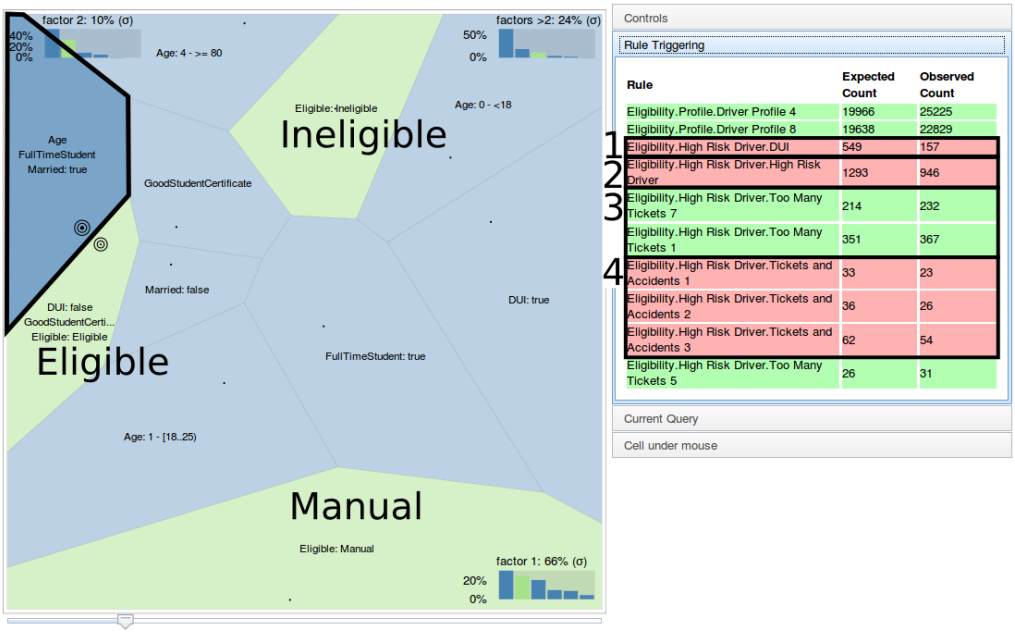


Figure 6.7: *Rule trigger view* updated for a selection of decisions. The selected cell represents people between 25 and 80 who are married and no longer full time student. These people tend to get less often marked high risk drivers and are less often caught for driving under influence.

him to add or rewrite business rules that better separate the two outcomes and lower the number of manual decisions (which are expensive). Second, values of different decision attributes are projected close together. This can happen when a decision has a more complex structure such as an insurance offer describing the insurance type, the base quote, the deductions, the surcharges and the final quote. When for example a low final quote value is projected close to the extensive insurance type value, a certain interplay between rules is causing an insurance that was meant to be more expensive to have a low quote.

These issues pertain precisely to our core question (Sec. 2.5): How can we examine how the reality (as captured by the decision model) *diverges* from the reality encoded by the actual decisions taken? As explained before, this distinction is shown in the *decision map* by using a two-color mapping. We further visualize this distinction using the decision execution trace information available (**DS3**) using rule trigger analysis as discussed in Sec. 5.3.

Recall that each decision is the result of the production rules that triggered for a given input. Given a random population of decisions, we can expect that each rule is triggered in the same proportion as for the overall population. Thus, selecting a particular decision subset and comparing the expected trigger count to the observed trigger count for this selection gives us insight on whether the decision logic of this rule is either over- or underrepresented for this input population segment. We sort the rules triggered for a subset of decisions by the absolute difference between the expected count and the observed count. Combining the correlation between properties with over- or underrepresented logic leads to a better understanding of

1. How a subpopulation of instances impacts the overall behavior of the decision.
2. How a particular formulation of rules leads to a unexpected behavior for a given subpopulation.

For this, we use a new view: the *Rule Trigger view* (Figs. 6.4C and 6.7). The view displays the rules triggered for a selection of decisions in a table. From left to right, the table columns

show the unique rule identifier, the expected trigger count, and the actual trigger count for the currently selected decisions. The table background is colored red when the expected count is below the actual count, and green when the expected count is above the actual count. In Fig. 6.7, the selected concept island represents people between 25 and 80 who are married and no longer full time student. From the corresponding *Rule Trigger view*, we learn that these people are less often caught for driving under influence (Fig. 6.7: 1), and get less often marked as high risk drivers (Fig. 6.7: 2). They do get tickets (Fig. 6.7: 3), but this occurs less often in combination with accidents (Fig. 6.7: 4). As the selected island is close to the *eligibility* cell, this enforces his expectation (**T3**, **T4**) that people who are eligible are indeed people who have reasonable driving habits.

We also considered showing rules that did not trigger for a selection. However, this leads to visual clutter. For example, when the selection contains only females, all rules that only apply to males will show up in this list.

6.4 Visualization refinements

In its most simple form a 2D scatter plot consists of 2 axes, both representing a variable of the dataset. In our case the projection of an attribute on the first and second factor, resulting from the MCA. Each data point is drawn on the intersection of the lines that are perpendicular to the axis at the location that represents the value of the variable represented by the axis for the observation. This simple form can be extended by encoding additional variables using color such as proposed by Ware et al. [196] or by the size of each visual object representing an observation.

Attribute values that are selected by the majority of observations are projected close to the barycenter of the 2D projections point-cloud by the MCA. When using the scatter plot approach, taking the first two factors and use these as x and y-axis for the scatter plot, this often leads to a strong cluster near the barycenter and some outliers surrounding it. From a business perspective it can be argued that the values in the center are initially the most interesting as they describe the properties of the average business case. Yet, less space is assigned to the attribute values projected in this region. As a consequence the points in these clusters cannot be labeled without causing clutter which enforces interactive techniques to identify the points.

In this section we present two techniques that address these problems. We first present a method to label areas of a plot in a meaningful way in Sec. 6.4.1. Next, in Sec. 6.4.2 we present two methods for automatically adjusting the scale of a scatter plot in order to reduce the screen estate that is devoted to outlier values.

6.4.1 Labels

When observations have a textual description, labels can be shown in the scatter plot near each point to identify individual observations in the plot. In our particular case each point in the *decision map* is labeled with the attribute and the value it represents. The *decision map* has an abstract nature as opposed to geographical maps where knowledge of the user can be of help to identify point features without labels. Due to this abstract nature labeling is essential to give the user an initial understanding of what he is seeing.

Labeling techniques can be classified into two categories: static labeling and dynamic labeling [197]. Static labeling means that a label is visually associated with each observation in the best possible manner. That is, it is clear which label belongs to which observation, is readable and does not hide any other important information. Dynamic labeling (also called tooltips sometimes) works on a subset of all points based on interaction with the view and is therefore bound to interactive visualizations.

Related work for labeling

Static labeling of plots or more general point features has been studied for quite some time, especially in the area of cartography. Christensen et al. [198] performed an empirical study of algorithms for point-feature label placement in which they also give an extensive overview of proposed algorithms. They distinguish two main classes for label placement algorithms: those that perform a global search for the most optimal label placement and those that perform searches on a local basis only. In general these algorithms, besides their computational complexity, are likely to leave out labels in dense areas or perform overlap.

Been et al. present a method for placing labels in dynamic maps which support continuous zooming and panning [199]. We have a fixed map and do not support zooming and panning.

There have also been some dynamic methods for labeling proposed. Fekete et al. presented Excentric labeling [197], where a focus region that is directed by the user, is used to label only a selection of points. Refinements for this technique are discussed by Fink et al., which propose various algorithms to connect data points from the focus region with their corresponding label [200]. These two methods leave the user in our context with no other information than a cluster of points. We would like to be able to provide some initial labeling in order to guide the exploratory process of the user.

Shneiderman et al. presents Direct Annotation [201], which employs a drag-and-drop strategy for labeling photos. This method is only useful when the objects that are to be labeled are known beforehand and can be recognized visually.

Area labeling method

We extend our initial labeling approach from Sec. 5.2.5 to provide more detailed labels. This extended approach for labeling MCA scatter plots is based on the observation that it is not the individual points (attribute values) of the plot that are of interest but the groups of closely projected points. In all generality we label areas of a plot with meaningful labels based on the points that fall into this area. Therefore, we loosen the challenge stated by Fekete et al., [197]: A label is non-ambiguously related to its graphical object. Given that we label areas of the plot it is not always possible to exactly identify each point. Nevertheless, in the context of displaying the results of a MCA this is not a real problem. The label method we present here comprises two steps: determine the areas which are labeled and perform label compression.

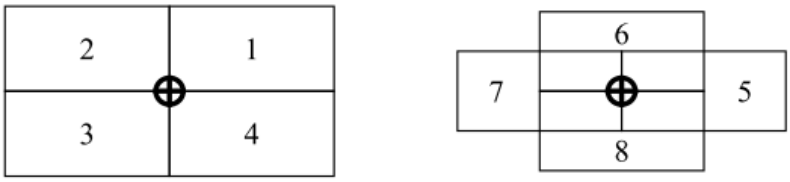
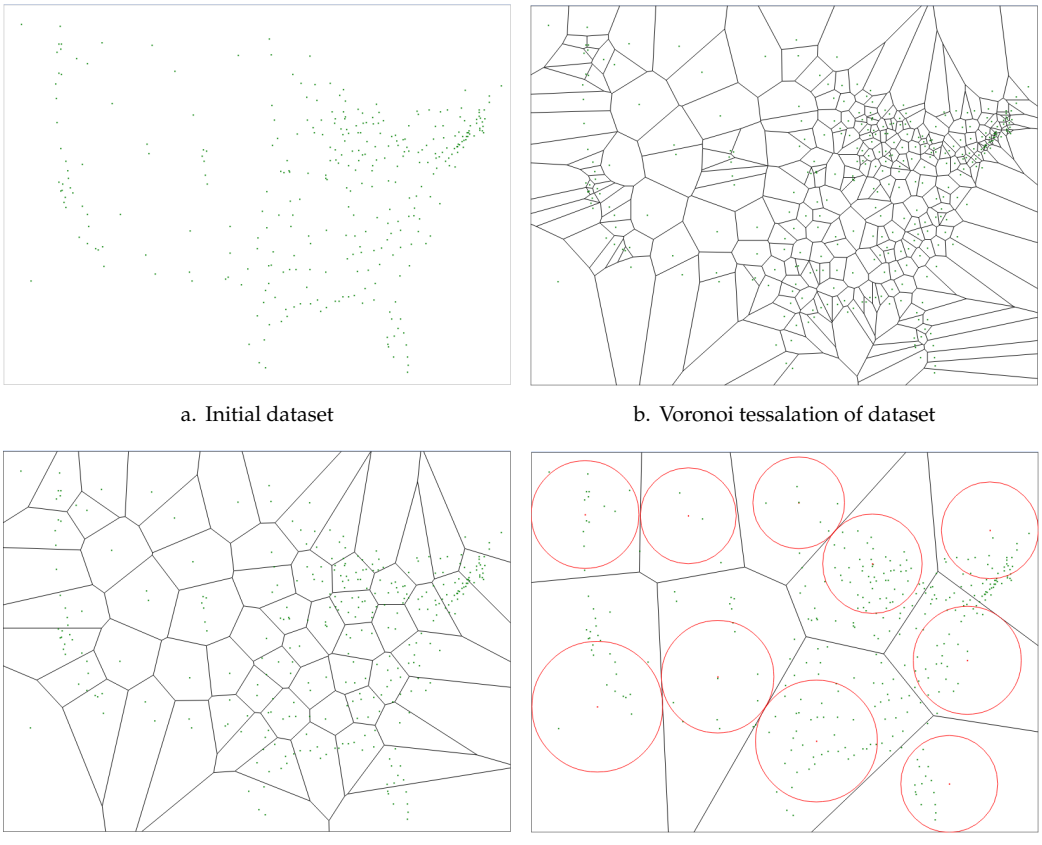


Figure 6.8: Possible positions for label placement when labeling point features. Image from [198].

Determine Label areas: One important feature of label placement is the prevention of label overlap. Many methods place labels at one of the eight locations as shown in Fig. 6.8. Overlap is prevented by alternating the label position for a given point until there is no or minimal overlap. Another way to look at this problem is as follows. When taking the Voronoi tessellation of a set of points, each cell represents the space that is closest to the point in it. In other words, when a label is placed within the bounds of a Voronoi cell belonging to a point, prevention of overlap is guaranteed. Fig. 6.11 shows a scatter plot of a dataset containing US cities. The longitude and latitude are mapped to the x-axis and y-axis of the plot. The labels, not shown in this plot, have the form ${state:city}$, e.g. “NY:New York“ or ”CA:San Francisco“.

At this point there are still two open problems. First, as can be seen in for example the top-right section of Fig. 6.9b, there are many data points that result in such small cells that there is not enough space for even one character, let alone a meaningful label. Second, assuming that we want to place labels in a horizontal manner, some cells are very elongated allowing for a meaningful label in a vertical way but not horizontally, while others are flat allowing for a label and yet others are squarish allowing for some text but not a full label. We address these two issues by using cell-merging to obtain larger cells in crowded areas and by using the maximum inscribed circle to determine the available space for a label.

Cell merging is applied under the assumption that a group of points that have a small distance can perhaps not be labeled individually in a static way but can be labeled with a meaningful summarizing label. We start with the observation that small cells are the center of a cluster, i.e. the cell is small because it is closely surrounded by other sites. Thus sorting a list of cells by size in ascending order means that the centers of clusters are at the start of the list. Next we proceed by iterating over this list and each cell is merged with all its neighboring cells, where all visited and neighboring cells are marked as processed. For each group of cells a new replacement site is calculated, which is the barycenter of the sites of the cells in the group. The resulting set of barycenters is used to create a new Voronoi tessellation. This new Voronoi tessellation is next used to repeat the process. The stop criteria for this merging process is visual and can be either



a. Initial dataset

b. Voronoi tessalation of dataset

c. Voronoi tesselation of merged cells

d. Maximum inscribed circles of Voronoi cells

Figure 6.9: Scatter plot of US cities data. (a) Plain scatter plot where X and Y represent longitude and latitude. (b) Voronoi tessellation of the plot using the data points as Voronoi sites. (c) Initial cells are merged to gain larger cells for labels. (d) Cells are merged even further and inscribed circles are shown to illustrate the space that will be used for labeling.

manual or automatic based on heuristics. In the manual case the user interacts with a slider to create a merging that results in a labeling that fits the presentation needs, very similar to the mechanism presented in Sec. 5.2.5. Alternatively, this process can be stopped based on heuristics. One obvious heuristic would be to continue until the smallest cell has reached a certain area. Another heuristic is to stop when the cells have roughly similar sizes, by choosing a merge factor that minimizes the variance of the cell sizes.

Voronoi cells are obtained from the intersection of half-spaces and thus are always convex. To determine the available area for a label in these convex polygons we used maximum inscribed circles centered on the barycenter of the cell polygon. Both the cell its barycenter and the radius of the inscribed circle, which is equal to the length of the straight line to the nearest edge, are easily calculated using standard geometry. Every text that falls into this circle is guaranteed to stay within the bounds of the cell as shown in Fig. 6.9d.

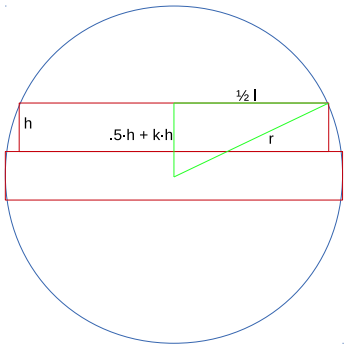


Figure 6.10: Calculating available lines of text in an inscribed circle. r: the radius of the inscribed circle; h: the line height based on font and font size; k: the index of the label; 1/2 l: half of the line length for a given line

Given the inscribed circle of a cell a label can be constructed by dividing the circle into lines based on the font height. The calculation of the length of a line is straightforward as shown in Fig. 6.10. The initial line is placed in the center of the inscribed circle taking full width. Consecutive lines are placed above and below this initial line. The length of each line is calculated by solving the triangular inequality

$$r^2 = (.5h + kh)^2 + (.5l)^2 \tag{6.2}$$

with r and h known, l unknown and k being the index of the line above the central line. Another approach, is sorting the labels on length, iterate over each label and put them alternately in two different arrays. Next, concatenate the first array with the reverse of the second. The resulting array contains the labels in a roughly circular or ellipsoid shape based on length. Based on the calculated lengths, label texts must be determined and compression must take place when these labels exceed the calculated lengths.

Label Compression: As stated before, each label has two parts: the attribute and the value. When grouping points together it can happen that the group of points contains values of the same attribute. To reduce the screen space required for the final label of this group of points various compression techniques can be applied both without and with loss of information.

A lossless compression technique is merging values of the same attribute. Instead of drawing the full label for each value, a single label can be constructed by taking the attribute name and append the values separated by a comma.

`Before:`

```
  Occupation: Adm-clerical
  Occupation: Prof Speciality
  Occupation: Craft-repair

After:
  Occupation: Adm-clerical, Prof Speciality, Craft-repair
```

When the attribute value descriptions are long, this approach quickly results in long labels. To overcome this, some loss of information could be allowed, by taking the first N characters of value instead of its full name. Here N must be sufficiently large to distinguish the values from each other.

```
Before:
  Occupation: Adm-clerical, Prof Speciality, Craft-repair

After: (N=4)
  Occupation: Adm-, Prof, Craf
```

Even though there is some loss of information, under the assumption that the user has a clear understanding of the domain, this will still provide enough information at first sight.

Another lossless way of compressing labels is based on the data type of the attribute, in particular for boolean and nominal attributes. In the case of boolean attributes the point is indicating that the attribute is true or false. These attributes are likely to be named in such a way that adding "true" to the label does not add any information. Therefore we omit the value in the positive case and we prefix the label with "not" in the negative case.

```
Before:
  CompeletedDriversEducationCourse: true
  CompeletedDriversEducationCourse: false

After:
  CompeletedDriversEducationCourse
  not CompeletedDriversEducationCourse
```

In the case of nominal attributes there is a ranking of the values. Therefore, if there are three or more consecutive values part of the same group, it is not needed to list all these values in the label. When the ordinal variable is the result of quantization, the upper and lower bounds can be used for a more precise label.

```
Before:
  Rank: 1,2,3,4,5,8,9
  Age: 25-35,35-45,45-55

After:
  Rank: 1-5,8,9
  Age: 25-55
```

Finally there is the case where all values of a particular attribute are merged into the same group. In the context of MCA this indicates that there are no particular features separating the observations with respect to this attribute. That is, for all other attributes these observations tend to have the same distribution of values. When looking at the results of a MCA we are interested in clusters of attribute values that separate part of the observation from the rest. Clearly, when all

values of one attribute fall into the same cell, this does not provide any distinguishing information. Therefore, the label can be left out completely but this can be confusing to a user who will see values at a certain level of merging which disappear at a coarser level of merging. To solve this latter problem, the attribute can still be displayed with a postfix indicating that all values are merged in the cell.

```
Before:
  Gender: Male, Female

After:
  {No - label}
  Gender: All
```

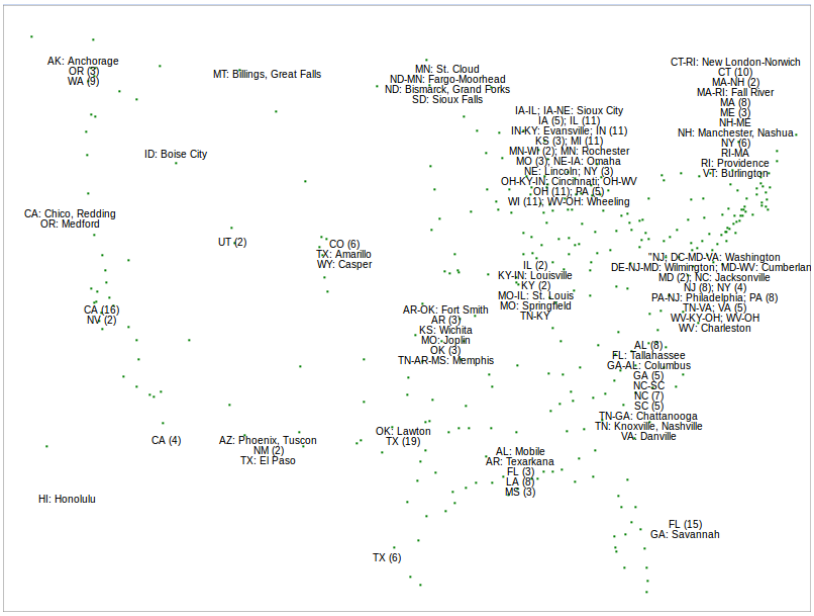
Some of the above discussed techniques are application specific, especially those that involve the data types. These are therefore not applicable to more general sets of labeled point features such as the above dataset of US cities. A generalization can be applied to perform similar compression to a general set of labels. Instead of depending on a $Category : Value$ format, one can still look at the longest prefix for a set of labels which is then used in a similar way as the category in our above discussion.

When strictly keeping labels within the bounds of the inscribed circles there will be no overlap of labels. However, as can be seen in Fig. 6.9d, this method does not guarantee the most efficient space usage. The cell in the bottom-right corner for example has an inscribed circle that leaves out quite some additional horizontal space. Additionally, when a merged cell contains many points, even a compressed label might not fit in the inscribed circle requiring leaving out information or not show a label at all resulting in a labeling as shown in Fig. 6.11a for the US-cities example. Still, when allowing labels to overflow the circle and using the manual stop criterion, a pretty good global labeling can be obtained in many cases such as shown in Fig. 6.11b. In practice this technique should be combined with interactive techniques in order to follow the "overview-first, details-on-demand" paradigm.

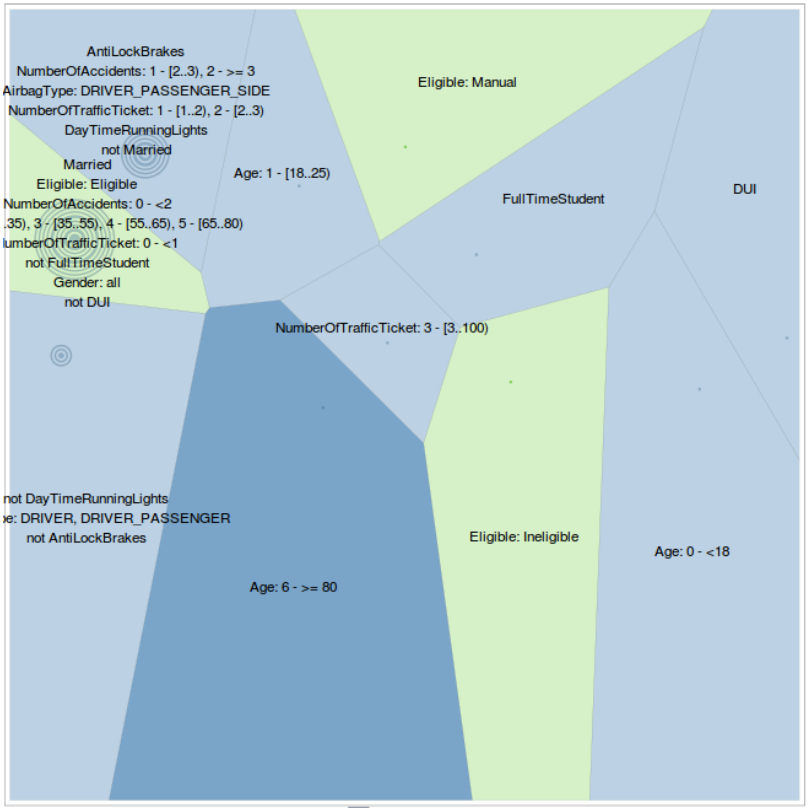
6.4.2 Scales

Scatter plots, or bi-variate plots, are such a common tool for both exploratory and explanatory visualization that rarely thoughts are given about how misleading they can be. Nowadays, these plots are automatically generated, sometimes from huge datasets, and used in business contexts with little caution about checking the significance of the visually extracted conclusions. They are often used to find features of a dataset [202] which are identified using relative distance between points. However, with a large number of points, problems arise as shown in Fig. 6.12. This scatterplot shows file access time (horizontal) versus file size (vertical) for 5200 files for a software project directory. It is clear that most activity is in the third quarter of the timespan covered by the X-axis. Yet, a handful of outliers, access time wise, in the first half of the timespan take up a lot of screen estate, therefore hiding interesting details in the second half. Similarly, only a couple of small files, make that the majority of points is cramped in the upper quarter of the plot, again hiding details for the majority of points.

The problem in this plot is that the density of points is not uniform along both axes, resulting in more than one data point per display unit. This problem also arises when multiple data points have the same value which can happen when data points are categorical or integer, in which case data points are plotted at exactly the same location. Plots needs to be carefully designed in order to overcome these kind of problems and a number of guidelines and design considerations exist [162, 203, 204, 205, 206]. Concluding, a common problem for 2D plots with a high number



a.



b.

Figure 6.11: Labeling results for two different datasets. (a) Labeling of the US-cities dataset, lots of information gets lost in crowded area such as top-right. (b) Labeling of a MCA result, nicely identifying the two clusters at the top.

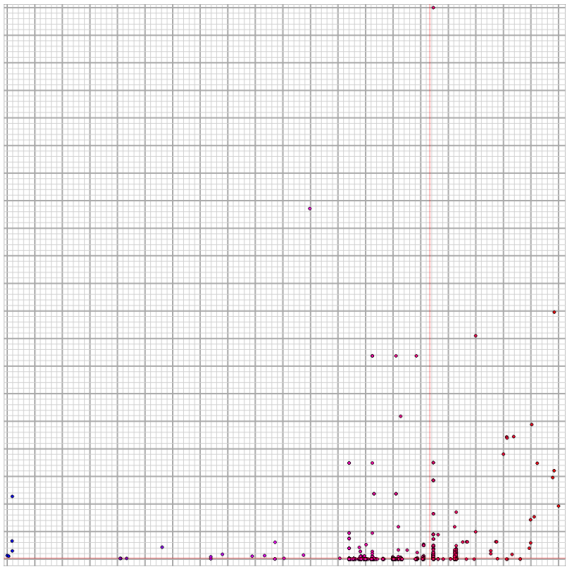


Figure 6.12: A scatterplot of file access times vs file size, containing 5200 data points. Horizontal the time at which files where accessed, vertical the size of the file that was accessed. Outliers for both access time and file size have a strong impact on the appearance of the plot. The red lines indicate average values for both axes.

of points is that structure, patterns and details easily get hidden by outliers possibly leading to false conclusions. The automatic generation of data-dependent scales, which we introduce, aims at transforming automatically the projection space to better reflect the relative distribution of points. With this approach we aim to support the user with:

- detecting the overall distributions of the points better by leveraging the scale's marks
- revealing local structures and patterns without the need for the user to find the correct scale that shows these features.

As an interesting complement, data-dependent scales provide support for other common tasks performed with 2D plots. For instance, when the plot is showing the results from a dimensionality reduction technique such as PCA or MCA, as is the case in our *decision map*, the analysis is driven more by the relative position of points on the map than by the overall shape of the cloud. The *decision map* therefore extends the *projections view* by adding data-dependent scales, which enable preserving relative positions while reducing visual clutter.

Related work for plot scales

In many cases it is common to use non-linear scales, for example log and power scales, provided by most charting packages. More advanced transformations are proposed in statistical graphics packages and literature [23, 163], such as the probit and logit scales, which more adequately adjust the scale when their distributions follow a symmetric and unimodal distribution (e.g. normal or logistic distributions). However, these scales have to be chosen manually based on first visual inspection or prior knowledge of the data. The general point of applying a scale different than linear is to assign more space to dense areas of the plot. This rescaling is based on the assumption that the relevant regions of a plot are the dense areas. Thus, spreading out the data in the high density regions will help to uncover hidden features and discard outliers: the rescaling we

propose is not based on correlations between values of the data in the two dimensions but on the density of the data along each axis.

More sophisticated, non-monotonous transformations, such as warping, popularized by Tobler's cartograms [207] can be applied to better spread the data on bi-dimensional plots. Our concern is to keep the type of transformation easily interpretable. Hence we discard transforms whose nature could not be conveyed on a single, one-dimensional scale.

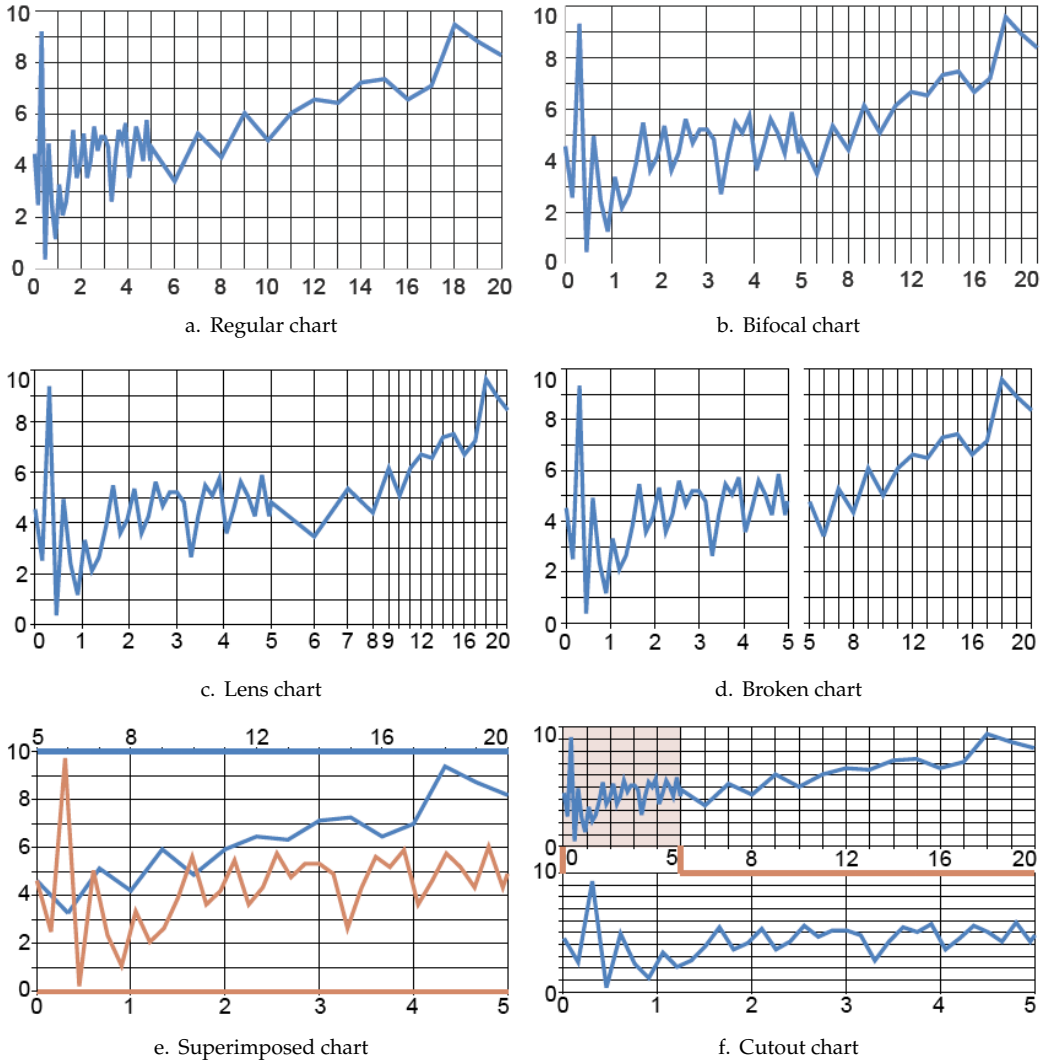


Figure 6.13: Various approaches for dual scale charts. Images from [208]

Another approach is to apply a transformation to the dense parts of a scale. In an empirical user study Isenberg et al. [208] evaluate five different approaches to dual scale charts, shown in Fig. 6.13. These charts divide the scale of a particular axis into two main scales, where one scale is used for the focus region and the other for the context region. Cleveland suggests [23, Ch. 2] that these kind of charts are displayed using a broken chart such as Fig. 6.13d. The superimposed chart (Fig. 6.13e) uses the full width of the chart for both the focus region and the context region showing the scales for each region on opposite sides of the chart. The cutout chart (Fig. 6.13f) shows both the full region and the focus region in two different charts, using visual clues to convey which part of the full region is shown in the focus chart. Finally, the bifocal chart

(Fig. 6.13b) and the lens chart (Fig. 6.13c) show the two different scales along the axis. This results in a chart where the focus region is enlarged to fill a larger part of the drawing space. The lens chart uses a transfer function that gradually moves from one scale to another, while the bifocal chart abruptly changes its scale at the end of the focus region.

For none of above approaches it is discussed how these scales should be chosen. With global approaches the scale is chosen after initial visual inspection of the data or based on prior knowledge or expectations. Local approaches seem to hint to an interactive implementation where the focus area can be moved around based on the needs of the user. In contexts where the main focus is on the dense areas initially, automatic methods to emphasize these regions could be of great help.

To address the issue of local clutter, a variety of interactive techniques such as Fisheye views [209, 92], Magic Lenses [210], and Excentric Labels [197] can be used. Those techniques allow to transform portions of the display interactively, which provide for deeper inspection of local areas. They are not yet very much encountered in common charting packages. Brushing and linking techniques (e.g. [211]) on the other hand appear to be a more common way to reveal underneath patterns or trends in a 2D plot.

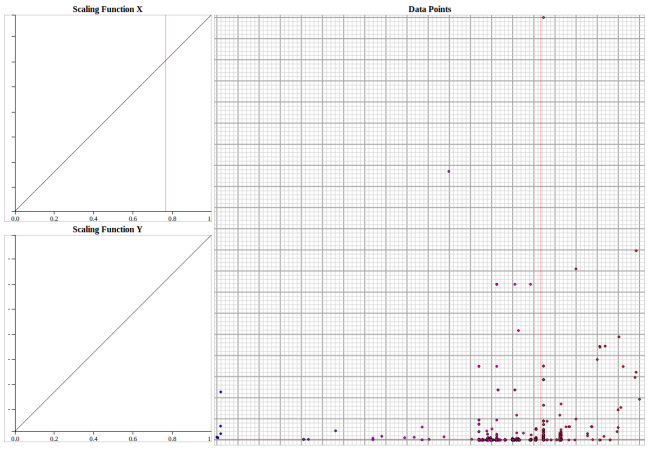
Data Based Scaling

The idea of finding the best scale type automatically by analyzing the dataset is the starting point of our work. We propose two different approaches to this problem, which both automatically rescale the axes without requiring the user to input any parameters. *Global scaling* preserves the global appearance of the data, in that distances between points do not change too much. *Local scaling* focuses on the local density, i.e. number of points in a given area, of the data and gives any subset of points a surface in the plot that is proportional to the number of points, at the expense that distance in the scaled plot can vary greatly from the original distances. Both approaches output a rescaling function that maps the coordinate of a point in the original scale to its coordinate in the new scale. All axes are rescaled independently from each other.

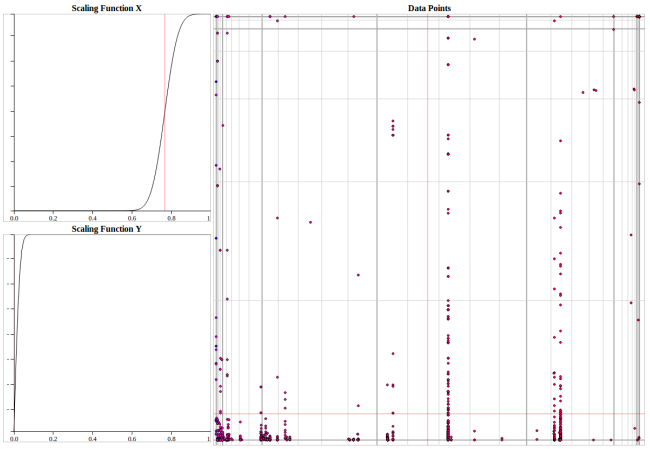
Global scaling We want to increase the surface assigned to dense regions of the data plot, using a smooth function to preserve distances. We do this by matching the actual distribution of points to distributions in a catalog and finding the best fit. This catalog contains the four most common families of distributions for rescaling data plots: uniform, power, log, and normal. First, for each of the distribution families in the catalog, we use the least squares method to find the best fitting parameters. Next, we keep the function that has the least squares distance from the actual distribution. Finally, the cumulative distribution function of the best fitting distribution is the rescaling function that will result in an optimal spread with respect to the functions in the catalog. Each distribution fitting has a complexity linear in the size of the input set, so this method scales very well for large datasets.

Local scaling When a distribution is multi modal, the global approach gives suboptimal results, unless the catalog would contain every possible function for multi modal distributions. To deal with this case, we have to abandon the requirement for a smooth scaling function and adopt a local approach. Cleveland [212] suggests to improve plots by showing a weighted regression line, which is also a local approach. We take a different approach by assigning for any subset of points an interval whose length is proportional to the density of the set. This will reveal parts of the plot that have a high density and a small spread in the original scale. The transformation of the scale is done as follows.

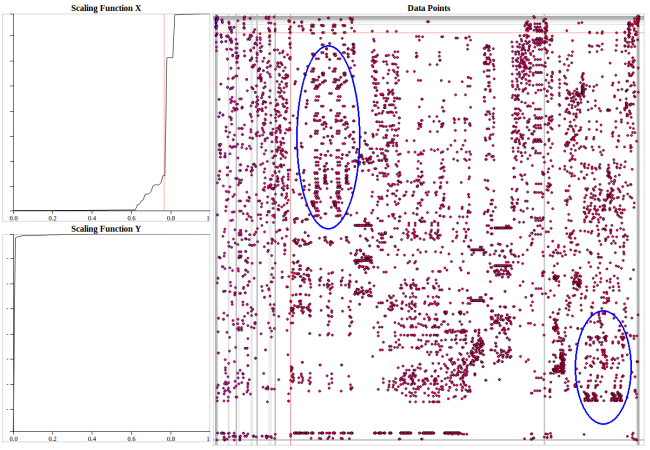
We have a collection P of N 2D points. For each $p_i \in P$ we call v_i the coordinate of p_i on the axis that is rescaled. The local rescaling function is defined as follows. First we sort the values



a.



b.



c.

Figure 6.14: Different scales for scatterplot of file access times (horizontal) vs file size (vertical). (a) Linear scaling: Details and patterns hidden due to extreme outliers, both time and size wise. (b) Global scaling: Some time patterns start to emerge, lots of activity in the beginning. (c) Local scaling: Fine-grained time/size patterns emerge.

v_i in increasing order. For each $i \in [0, N-1]$, if $v_i \neq v_{i-1}$ then the rescaling function maps $v_i \mapsto \frac{i}{N-1}$. The condition $v_i \neq v_{i-1}$ is to ensure that each unique v_i gets mapped to exactly one value. (We note that this rescaling function is only evaluated for the values P_v, hence there is no need to calculate values in between v_i and v_{i-1}). Because we sorted the v_i before the mapping, the function preserves the relative order of the points along the axis. This scaling method is dominated by the sorting pass, with a complexity of $O(n \log n)$, which ensures this method scales equally well for large datasets.

Discussion

We demonstrate the rescaling method on two datasets using a prototype that implements both methods. This prototype lets the user switch between the original, global and local scalings. In order to give him a visual clue on where points move, we have implemented a transformation animation of both the data points and the scale marks.

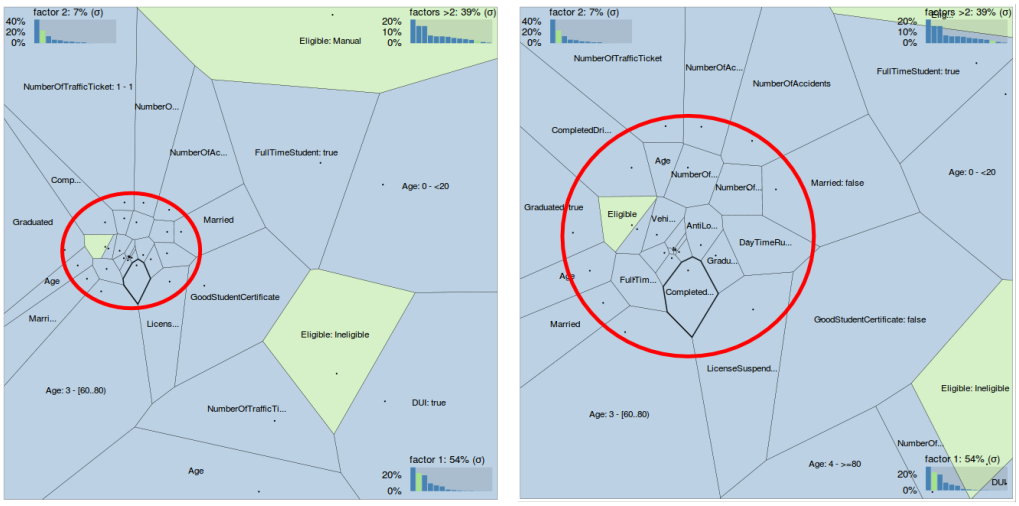
The first dataset represents the size and modification time of a set of 6200 files from a directory containing a software project. Fig. 6.14a shows this data with on the horizontal axis the modification time and on the vertical axis the file size. The two plots on the left in subfigures of Fig. 6.14, show the scaling functions for both axes. With a linear scale eight large files occupy almost three quarters of the plot, hiding all relevant and interesting information.

With the global scaling method we notice several things. The best scaling function in the catalog for the axis corresponding to the file sizes (vertical) is a log. From the linear scale it already became clear that there are many small files, with the global scaling (Fig. 6.14b) the emphasis now actually is on these files. On the horizontal axis (modification time) the normal distribution matches best the actual distribution and is used for rescaling. From the linear scale it also became clear that most files were changed in a relatively small timespan, which gets the emphasis with the global scaling. As a result we now see that at several points in time many files where changed in a very small time interval indicating active development and compilation. Even though a normal distribution is the best match for the actual distribution of points on the time axis, it does not spread out these clusters well. This prevents us from analyzing patterns within these clusters.

When local scaling is applied, the vertical clusters are spread out horizontally as well. Now fine grained details are revealed such as the patterns highlighted in Figure 6.14c. In the highlighted areas we see files of the same size having the same modification patterns, those correspond to identical subdirectories.

The second dataset is the result of a multiple correspondence analysis of a car insurance dataset. When performing MCA, correlated high frequent values form dense clusters. As a result there is little space for labels in these areas as shown in the highlighted area in Figure 6.15a. Although, local scaling would result in a most uniform spread and therefore give most space for labels, it also make interpreting distances hard. Therefore we apply global scaling, as shown in Figure 6.15b. As a result the overall structure of the data is kept intact, while there is now more space available for the central values.

We addressed the crowding issue by applying a transformation to the projected MCA values as follows. First we scale both factors f_1 and f_2 to $[0..1]$. Next, we calculate the variance and the mean value for the values of the scaled factors. These values are then used to configure two scaling functions, one for each factor, that are centered around the mean. An uniform distribution in the $[0..1]$ domain has a variance of $1/12$. We use this observation by assuming that when the variance is under $1/12$, the distribution is close to a normal distribution. Next, the scaling function behaves as a cumulative density function (CDF) for a normal distribution for variances below $1/12$ and as the inverse when the scale is above $1/12$. Overall, this scaling creates more space for clustered values while keeping the overall structure intact (Fig. 6.15), because both the CDF and its inverse are monotonically increasing.



a. b.

Figure 6.15: (a) Clutter in the barycenter of MCA plots. (b) The global transformation gives significantly more space to central values while keeping the overall plot structure intact.

Finally, we note that our scaling methods have some conceptual relationship to dimensionality reduction. Our scaling methods preserve local neighborhoods, but the absolute coordinates of the scaled points become meaningless. This is precisely what happens in dimensionality techniques.

6.5 Implementation

The Decision Exploration Lab is implemented as an extension for IBM Operational Decision Manager [9]. Its input data are gathered from ODM's decision warehouse. In detail, we extract the data from this warehouse in a separate Apache Solr search server [191]. This decouples the load caused by analysis from production systems. Secondly, this allows data to be indexed and stored in a suitable format for efficient querying (Sec. 6.2.1). With this architecture, MCA can be performed in real time for datasets with hundreds or thousands of decisions. Binning numerical variables, a task that goes over all decisions, takes more time when the number of decisions grows. However, this is normally done only at the beginning of the analysis process and hence does not have a large usability impact.

DEL consists of a backend, which is implemented using Java technologies, and a web based front-end. The front-end is implemented using the Dojo Toolkit [213] for the overall layout and standard widgets. For the *decision map*, we used D3.js [214] which provides the needed flexibility for custom widgets and interactions tightly bounded to the data at hand. The software is tested on a standard dual-core 2.5 Ghz machine with 4GB of RAM. On this configuration, DEL can easily handle 100000 decision instances in real-time.

6.6 Conclusion

We presented the Decision Exploration Lab (DEL), a visual analytics solution designed to address prevalent issues in the area of enterprise decision management. DEL is an end-to-end system which is integrated with IBMs Operational Decision Manager (ODM) and provides an exploratory work flow based on the data extracted from ODM. It provides two working modes

to support the exploration process of a business analyst. *Verbal mode* is a new extension which provides search and filtering facilities for the specific context of ODM. *Visual decision exploration mode* refines and extends the techniques presented in CHAPTER 5. We refined the *dimensions view*, by adding check-boxes to the categorical attributes in order to support changing the selection of attributes under analysis in an interactive manner. The *decision map* refines and extends the *projections view* in various ways. Firstly, it uses a two-color scheme to distinguish business-case attribute values from decision attribute values. Secondly, it provides a new labeling mechanism, which complements the tool tips of the *projections view* in order to provide more information without interaction. Finally, it uses data dependent scales as a mean to make better use of the visual space in relation to the data distribution.

The *decision map* is a refinement of the *projections view*, such that the focus is on the relation between business case and decision data. Therefore this technique now provides a better solution for research question **Q4**. Further refinements in the *decision map* by improved labeling and data dependent scales, result in more readable *decision maps*. Thus we improved our initial solution for the research question on data structure (**Q3**). Additionally, we integrated rule trigger analysis with the *decision map*, such that brushing the *decision map* gives rule trigger information about the decisions that match the brushed properties. This addresses the research question on the relation between business data and decision logic (**Q5**).

The industrial relevance of the problems addressed by DEL was expressed by one of the IBM ODM chief architects as follows: "A significant part of the ODM customers implements a feedback loop to improve the the logic of their automated decisions from the actual outcomes of decisions. They gain insight in the business efficiency of these decisions by monitoring, Key Performance Indicator (KPI)s and Business Intelligence approaches. But when it comes to figuring out what concrete parts of a decision model have to be improved or rethought to meet a particular goal, they can only rely on their knowledge."

The toolset presented in this chapter is designed to integrate with IBMs ODM. However, we are confident that to a large extend our approach is generic for other Decision Management Systems (DMSs) as well. As long as a DMS supports extraction of decision instance data and rule execution data, most of the techniques presented in this chapter can be applied directly. We store these two kinds of data in a separate back-end, optimized for the kind of analyzes we perform on it. The only feature that is specific to IBMs ODM is the usage of its framework for natural language querying. This is, in the context of this thesis, not considered a major contribution or feature and as such does not cripple the overall generality of our approach.

The originality of our approach is that we provide a toolset that supports combined analysis of rule execution traces and decision instances. This allows business analysts to explore decision models in the light of the accumulated facts about this model in the form of decision instances. As a result, analysts can verify if the decision model and business cases that have come up are diverging, and if so, understand the underlying reasons and take corrective and preventive measures.

Understanding the functioning of decision models is a challenging topic and much more work is to be done. Both the ontology and the production rules of a decision model contain a wealth of information that can be included in the visual analysis process.

Although we have presented a system to analyze the aggregated effects of many decisions, we have not shown yet if and how succesful this system is. Therefore we focus on evaluation in the next chapter: First we discuss what it takes to evaluate a system for gaining insight in DMSs. Secondly, we present a preliminary evaluation of our Multiple Correspondence Analysis (MCA) based analysis tool. Finally, we present two scenarios that demonstrate how the system we presented in this chapter can be used to verify expected and find unexpected functioning of decision models, using a Decision Model (DM) from the car insurance industry.

Chapter 7

Evaluation

Inspector: You have information for us.
V: No you already have the information,
all the names and dates are inside your head.
What you want, what you really need, is a story.
Inspector: A story can be true or false.
V: I leave such judgments to you inspector.

V for Vendetta, 2005

In Sec. 2.6.1 we identified business analysts as the target users for the Decision Exploration Lab. Typically, business analysts are working at different enterprises than the enterprises that develop a Decision Management System (DMS). This leads to the interesting problem that it is practically impossible to have conversations with analysts from all domains in which DMSs are applied. Consequentially, gaining a thorough understanding of their needs and wishes is equally hard. This makes evaluating possible solutions a daunting task.

In this chapter, we first discuss what it takes to evaluate a system that wants to provide insight in aggregated effects of a Decision Model (DM). Next we take three different approaches to minimize above mentioned evaluation problems while still gaining insight in the effectiveness of our approach. First, we did a preliminary user study with (under)graduate students as a first sanity check for the proposed techniques. Secondly, we constructed a realistic use case scenario based on a simplified application of DMSs in the car insurance industry. This scenario was used to presented our work at various stages to architects and product managers who have been at customers to integrate and troubleshoot DMSs in real world cases.

7.1 What does it take to evaluate an exploratory system for DMS

From the preceding discussion in this thesis it should be clear that we are not interested in the inner workings of a DMS itself. Thus, just having a DMS is not enough for an evaluation of the Decision Exploration Lab (DEL). The DEL extracts information from *decision instances* (**DS2**) and *decision execution traces* (**DS3**). Availability of this data implies the availability of two additional pieces of data: a DM and a set of business cases (Sec. 1.1).

A DM makes explicit how an enterprise deals with the situations particular to its business, how it reacts to certain customer requests or how it manages certain risks. It is therefore an asset that encodes how an enterprise keeps its strategic advances to its competitors. This kind of information is generally considered of high confidentiality and not readily accessible for third parties. In addition, these systems deal with highly specific markets and their DMs are sprinkled with domain specific information and abbreviations. As a result, one does not only need access to a DM but also significant investment of domain expertise to gain at least a basic understanding of what a model does.

We emphasize that both realistic business case data and a realistic DM are required for an evaluation. Each DM is constructed with certain knowledge gained by prior analysis and expec-

tations about the statistical properties of the business case data. In order to be able to test how well a DM models the decision it was designed for it should be analyzed with real data. That is, it should be tested with past business cases or with synthetic data that sufficiently reflects real data. In both cases the correct decision belonging to a business case should be known on forehand, either based on manual specification or based on an earlier (trusted) version of the DM. Random data will lead to random results, therefore nothing interesting can be learned from an evaluation approach that uses random data. On the other hand, a given realistic dataset, such as traffic accident reports provided as open data by a governmental organization, is not enough either. First of all, one needs to decide what kind of decision should be made based on the data at hand. Next, a realistic DM must be constructed which is not a trivial task in itself and likely requires domain expertise. Finally, once both realistic data and a DM are available, the analysis should preferably be performed by domain experts. They have the right level of knowledge to tell if a certain finding is expected or unexpected.

7.2 Data generation

For business case data we have similar concerns as we have for DMs. Business case data are often highly sensitive as it typically deals with information regarding insurance requests or claims, financial transactions, loan applications or customer spending behavior. This kind of data is of great value for enterprises in keeping their operations targeted and risks balanced. Additionally, often this data falls under privacy law and cannot be handed to third parties.

If a DM is available but not the business case data, data generation might become an option. Given the wealth of information that is available from various sources such as published census data, it becomes feasible to synthesize information that looks realistic but which is not real. Additionally, scenarios can be planted in this data, in the form of statistical biases, which can be used to evaluate if the tool is actually able to uncover such cases.

The need for generating data is not unique to our problem domain. Barse et al., generate data to train, test and compare fraud detection systems [215]. Jeske et al., generate test cases in order to be able to set a baseline for accuracy of Information Discovery and Analysis [216, 217]. Houkjaer et al., generate data for testing the performance of database management systems [218]. These approaches generate highly structured data. More homogeneous datasets are generated by the Threat stream generator of Withing et al., which generates scenarios containing text documents [219] as test datasets for visual analytics tools.

We found the most extensive discussion on the difficulties of generating artificial data in the work of Jeske and Lin [216, 217]. Generating artificial data that looks realistic is not just about generating data with a realistic distribution for each attribute. Attributes often have dependencies that should be taken into account when generating values for each attribute (e.g. younger people have a lower chance of being married in western countries). Jeske and Lin model these kind of dependencies with a semantic graph.

We also took the approach of data generation for our insurance scenario, where we had a realistic (although for presentational purposes simplified) DM of a car insurance application and quotation application. Various sources such as the U.S. census[1] and published data on car model production and prices where used to generate business cases for the car insurance DM. A difficulty that we have found, which is not discussed in depth in the cited work, is the following. When generating data, one often starts from already known sources, e.g. measured distributions of age, gender, education by census bureaus. These pieces of information come at different levels of aggregation. For example, one might find age distribution per state but car brand distribution only on a country level. Additionally, we found that there might be a difference between the data

[1] http://www.census.gov/

which is needed for generation and the data that is actually generated. For example, to generate a realistic income for a person, one might want to take into account the education of a person, even if the education itself will not be part of the generated data.

Our data generation experiences in the context of DMS has led to another interesting observation. On the one hand we introduced various hidden (from the DM perspective) variables. For example, we introduced the income of a person as a variable in order to determine the brand, model, value and security features of the car of the person. To obtain an income we also introduced an education level, which was than used to determine the income. Both income and education are variables that are not captured by the DM. We also introduced a risk variable which would be based on properties such as age and marital status and would in turn influence the number of tickets and accidents. The DM is of course constructed to reduce risk and will assign a risk score (unrelated to the one in the data generator) from properties such as the age and the number of tickets and accidents.

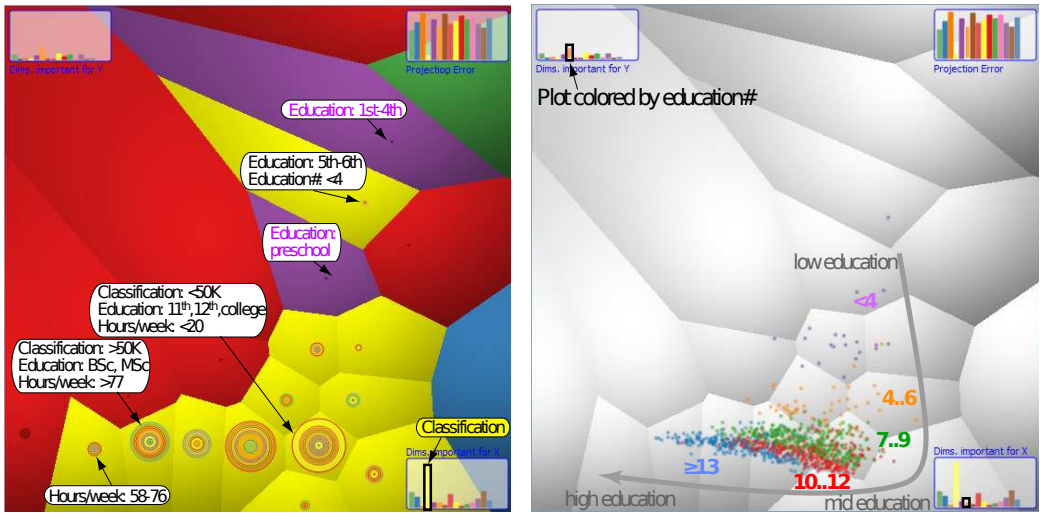
On the other hand, a very common pattern in DMs is to induce these kind of hidden variables from the information that *is* captured by a DM. In our particular case, the DM would capture whether a person is currently a student or not. And various properties of the car such as its value and security features. From this information the DM will deduct information about the persons wealth in order to propose additional insurance features to those who are expected to be wealthier.

Concluding, in our context we had a lack of multivariate data for a specific application domain with non-trivial correlations and constraints. Our goal was to generate data which are sufficiently realistic, meaning that we can show a scenario to a business analyst that conveys the usefulness of our solution. To this end an approach that integrates a dependency graph, similar to what Jeske et al. do, is required. However, as we found out, to construct this dependency graph, it is not the model of the data (i.e. the tables, table columns and relationships between tables) that serves as starting point, but the data describing the distributions for some of the variables. These external data sources for distributions come at different levels of granularity and therefore require a lot of manual work to construct a generator that integrates all this information. More research is required to see how the construction of a semantic graph based on external distribution information can be more automated.

7.3 Preliminary user study

As a first step in further evaluation of the techniques first presented in CHAPTER 5 we also used the *adult* dataset from [138]. This dataset has 15 attributes related to education in the US, including *education level*, *educations*, *work hours/week*, and *classification* (earning below or over 50K USD). After applying cell merging (Sec. 5.2.5) to find coarse patterns, the attribute plot shows a shape running from left to right and then curving upwards (Fig. 7.1a). The x *projection legend* shows that *classification* explains the x axis best: The $> 50K$ attribute cell is on the left and the $< 50K$ cell is at the right. Another left-to-right trend relates to *hours/week*, which is high on the left and low on the right, i.e. correlates with earnings. A third trend, which also causes the upward curve, follows the number of educations and education level. To the left, we find the most educated people (many *educations*, *education level*=BSc/MSc). Going right and then up, education decreases, with the least educated ($1-4^{th}$ grade) in the purple cell top-right. To confirm this, we use the observation plot (Fig. 7.1b), with observations colored by number of educations. We see here too the left-right-upwards trend starting with highly educated people, going through mid-educated people, and ending with a sparse cluster of low-education people.

To better understand our visualization's strengths and weaknesses, we conducted an ex-



a. b.

Figure 7.1: *Projections view* for the adult education dataset. MCA arranges data along a curve pattern following education (low..mid..high)

ploratory user evaluation[2]. The users were 14 computer science students (3 BSc, 7 MSc, and 4 PhD students), with 1..2 years of experience with general Infovis techniques, but no knowledge of Multi Dimensional Scaling (MDS) or MCA. They were given a detailed demo of our tool (45 minutes). Afterwards, each had to use the tool individually and answer three types of questions:

Q1: Find a meaning for cell groups to the top, bottom, right, left, and center of the *projections view*;

Q2: Explain the x and y projection axes in terms of attributes;

Q3: Find and explain salient clusters in the observation plot in terms of attributes.

The questions followed our own experiments (Sec. 5.2.5), so we could use our findings (unknown to the users) to validate results. For each question, users had to rank the usefulness and ease-of-use of the techniques (selection, brushing, color linking, *dimensions view*, *observations plot*, *projections view*, *cell merging*, *value filtering and merging*, and *projection legends*) on a five-point scale: very high (VH), high (H), low (L), very low (VL), and not used (NU). The assignment took under 2 hours and after that, the users could give additional oral feedback on their experience.

Table 7.1 summarizes the study's findings for 13 users (one of the BSc. students dropped out of the study). Overall, most users found the same cell groups, axis explanations, and clusters as ourselves. Color linking and brushing were found useful and easy to use. *projection legends* scored very well for Q2 and Q3 and were not used for Q1, in line with our design intention for this tool. Merging/filtering scored lowest, which can be explained by the relatively short training time put into this feature (5..10 minutes) and the fact that they require more involved choices (which values to merge or filter and merge distance, see Sec. 5.2.5). Finally, the usefulness and ease-of-use scores for the *dimensions view*, *projections view*, and observation plot indicate that most users perceived these (very) positively.

Although this exploratory study is far from a formal user evaluation, the results suggest that our techniques are relatively easy to learn for novice users, and can support the tasks and questions sketched in Sec. 5.2.5 up to a good extent.

[2]The complete evaluation can be found in APPENDIX A

Question	Tools usefulness and ease-of-use					Results	
		color linking	brushing & selection	projection legends	merging & filtering &		
Q1	VH	7	10			Found right group:	8/13
	H	4	3		2	Found left group:	7/13
	L	2		2	6	Found top group:	8/13
	VL			1	3	Found bottom group:	5/13
	NU			10	2	Found center group:	7/13
Q2	VH	6	9	1		Explained x-axis:	9/13
	H	5	2	4	3	Explained y-axis:	8/13
	L	2	1	5	6	Explained confidence:	
	VL			1	3	8 (sure); 2 (maybe); 3 (none)	
	NU			2	1		
Q3	VH	11	2	5		Found salient clusters:	10/13
	H	2	3	5		Found other clusters:	3/13
	L		3	3			
	VL		4		3		
	NU		1		10		

Table 7.1: Results of the user evaluation of the MCA visualization.

7.4 Car insurance scenario

We demonstrate our DEL toolset with the analysis of a business process from the car insurance industry with two decision models, introduced in Sec. 2.2. The input instances have been synthesized to avoid confidentiality and privacy concerns. The ontology on which the decision models operate consist of an AutoQuoteRequest. An AutoQuoteRequest contains information about the driver, the car for which insurance is requested, and the insurance type.

The logic of the risk management decision (Fig. 2.1, $DM1$) has 22 rules. These do various checks on the applicants, such as age checks and checks for high risk driving indicators. The logic of the pricing decision (Fig. 2.1, $DM2$) has 144 rules. A set of decision tables determines the insurance base premium based on the type of the requested insurance and car value. Additional rules determine which discounts apply for a request, e.g. anti-lock brakes and experienced-driver discounts. Yet other rules determine if certain surcharges apply, such as the old-vehicle surcharge. A final group of rules models region-specific insurance policies. These either override global policies or encode additional discounts and surcharges that only apply in particular regions.

We next illustrate the usage of our tool by two user stories. In both cases, the users are business analysts involved in deciding car insurance quotations. In the first story (Sec. 7.4.1), the business analyst discovers that too many applications are rejected and wants to find out why. In the second story (Sec. 7.4.2), the business analyst discovers that expensive cars do not get significant higher quotes.

7.4.1 Story 1: Why fewer than expected people are eligible

Using the *dimensions view* (Fig. 6.4A), the analyst examines the *eligible* decision concept. He sees that 88.6% of the requests are eligible, 10.4% are ineligible, and 1% are manually processed (Fig. 7.2). This is unexpected, because the analyst designed the decision model to have an eligibility rate between 90% and 95%. Thus, the analyst first learns that his model does not meet the expected performance (**T4**).

Using his knowledge about the decision model and business domain, the analyst now states that people below 18 and people above 80 are ineligible due to legislation and business policies.

He now wants to check this expectation against the actual age distribution of drivers. For this, he bins the numerical *age* variable into five categories, two for the lower and upper ranges of ineligible people, and three for potentially eligible people: < 18 - Not eligible, $[18..25)$ - Youth, $[25..65)$ - Adults, $[65..80)$ - Elderly people and $>= 80$ - Not Eligible.

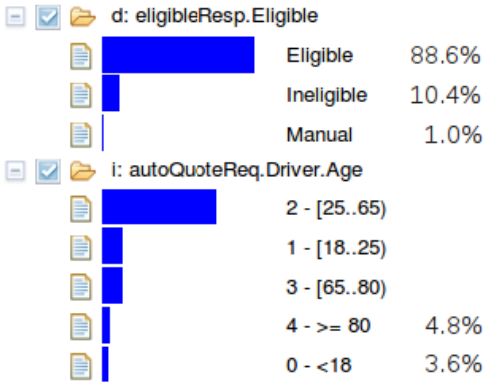


Figure 7.2: Details of the *dimensions view* showing the age and eligibility variables.

The binned *age* concept appears in the *dimensions view* under categorical concepts (Fig. 7.2). The percentages of the lower and upper age categories lead the business analysts to deduct that these two groups account for about 85% of the ineligible requests. Thus, 15% of the people that are ineligible (1.6% of the total) are so because of other reasons than age. This is a first clue that there is some room for decision improvement, i.e. maximizing the number of insurances sold. To find out how we can improve, we need to find out why people are ineligible for other reasons than their age (**T5**) .

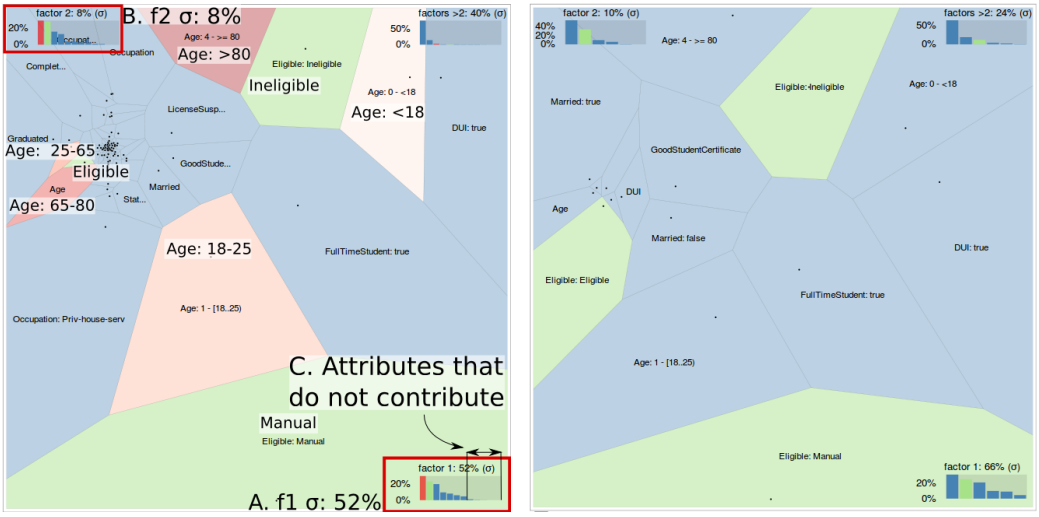
The analyst is next interested in the correlation between driver attributes and the decision (**T3**). He thus selects all driver attributes and the *eligibility* decision attribute in the *dimensions view*. The resulting plot (Fig. 7.3a) shows some structure, but is hard to examine due to clutter. Values above the top-left-bottom-right diagonal represent ineligible people. As expected, the $age < 18$ and $age > 80$ values fall into this area. Values below this diagonal are related to eligible or manually-processed requests.

From the *projection legends* (Fig. 7.3a: A, B), the analyst learns that only 60% of the variance is explained by the first two factors. Additionally, he finds that there exist several attributes that do not contribute to the structure of data (**T3**). These are the attributes in the bottom-right and top-left *projection legends* that are on the far right of these plots (Fig. 7.3a C). Among these, we have *state* and *gender*. This tells the analyst that state and gender do not influence the decision.

Next, the analyst iteratively removes such unimportant attributes until all remaining attributes contribute at least 5% to f_1 (horizontal axis). He now obtains a simplified map (Fig. 7.3b), with the attributes *age*, *eligibility*, *full-time student*, *married*, *DUI* (driving under influence), and *good student* certificate. This map now explains 76% of the variance and still shows the same diagonal structure as before. This enforces the earlier observation that the removed attributes do not contribute to the data structure.

To simplify the view even further, the analyst uses the merge slider to group correlated values in single concept islands. Left in the resulting *decision map* (Fig. 7.3c), he sees the concept island for *eligible* instances. The values of the two most important input attributes are $age : [25..65, 65..80]$ and $fulltimestudent : [false]$. Hence, the expectation that eligible people are mainly working adults is confirmed (**T4**). At the top in Fig. 7.3c, the analyst sees that ineligible people are correlated to age: $>= 80$. For instances where age is below 18, the most important correlated value seems to be DUI.

In the center of the *decision map* (Fig. 7.3c), the analyst finds a concept island for students between 18 and 25. This concept island is centered between the three decision values (*eligible*, *ineligible*, *manual processing*), showing that these decisions are about equally spread among students. Realizing that business policy state that extra effort should be made to make sure that students become customers (**T6**), he further investigates this population. He now sees that students are a group where some optimization may be possible, as these are not strongly correlated with any of the decision outcomes (**T5**). To find out more about students, the analyst selects the center cell to filter decisions and updates the *rule trigger view*. This view now shows only decisions for instances that have $age \in [18..25)$ and $fulltime\ student = [true]$ (5716 decisions).



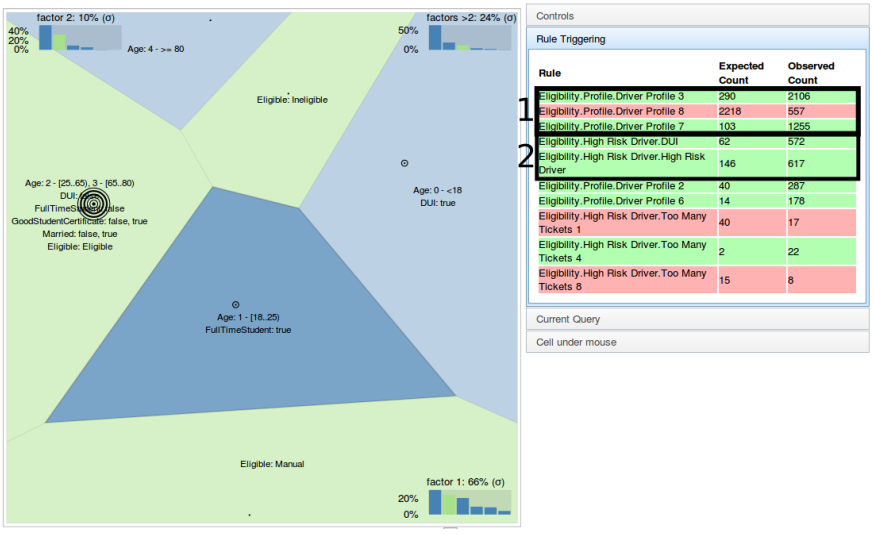


Figure 7.3: Interactive analysis of variables of interest: (a) The analyst has selected all available attributes related to drivers and clicked the age bar in the *projection legend* to see where the various age groups are in the plot. (b) The analyst simplified the plot by removing all variables that did not contribute to the structure of the data. (c) The analyst has merged values and selected the concept island of students to examine rule triggering for this set of decision instances.

	Gender	Age	State of Residence	Set Eligibility	
				Eligible?	Message
1	female	< 16		Ineligible	Sorry, you are too young to q...
2		[16 25[	CA	Manual	Females between 16 and 25 i...
3			*Otherwise*	Eligible	Congratulations! Your applica...
4		≥ 25		Eligible	Congratulations! Your applica...
5	male	< 18		Ineligible	Sorry, you are too young to q...
6		[18 21[	CA	Manual	Males between 18 and 21 in C...
7			*Otherwise*	Eligible	Congratulations! Your applica...
8		≥ 21		Eligible	Congratulations! Your applica...

Eligibility.High Risk Driver.DUI

```
definitions
    set 'name' to the first name of the driver ;
if
    the driver has been convicted of a DUI
then
    set 'high risk driver' to true ;
```

Eligibility.High Risk Driver.High Risk Driver

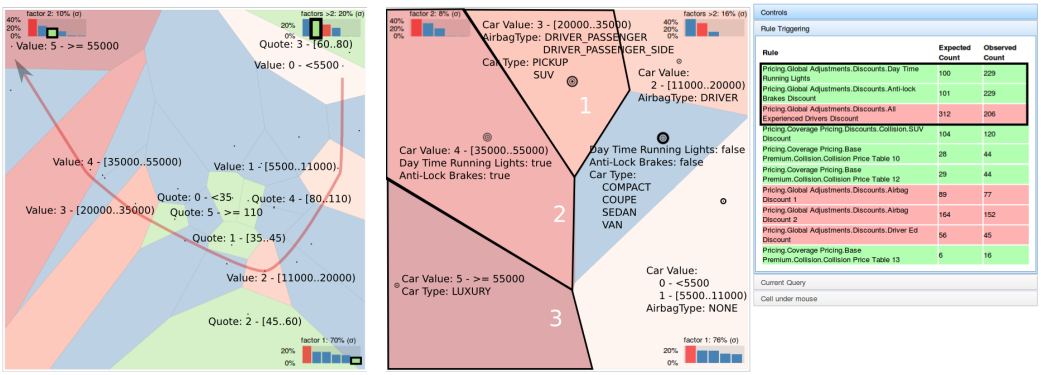
```
definitions
    set 'name' to the first name of the driver ;
if
    'high risk driver' is true
then
    reject this application with reason:
      "Eligibility Error: The driver, " + name +
      " is considered as a high risk driver" ;
```

Figure 7.4: Decision table for initial eligibility status based on age and the business rules that deal with high risk drivers. Note that the latter two, contain no tests on the age of a person.

Knowing the decision model, the analyst notes that the first three rules in the *rule trigger view* (Fig. 7.3c at 1) come from a decision table. For the definition of these particular rules, the business analyst turns to his rule authoring environment, IBM Operational Decision Manager (ODM) in this case [9]. In ODM, he looks up the decision table containing the rules he just found in the *rule trigger view* (Fig. 7.4). From their definitions, it is clear that these rules do a basic separation mostly based on age. In this decision table, the analyst sees that rule 3 and 7 affect people between 16 and 25 not living in CA, and rule 8 represents males above 21. Given that he is looking at students, it is not much of a surprise that rule 3 and 7 are overrepresented, while rule 8 is underrepresented for this selection of decisions (**T4**).

The two rules from the High Risk Driver package (Fig. 7.3c at 2) are revealing an interesting fact about the working of the decision model. Both the *DUI* and the *high-risk driver*, for which the definitions are given in Fig. 7.4, rules were not in particular written with students in mind. However, students trigger these rules more than the analyst should expect, given that they were not written for this group *in particular*. The analyst knows that, when an insurance request is flagged as being from a high risk driver, it will always be rejected by the decision model. However, he also knows that business policies state that additional effort should be made to make students customer. Using the above analysis, the business analyst decides to write an additional rule in the decision model which marks high-risk students for manual processing. This way, such students will be analyzed by salespeople (rather than automatically), and thus have a higher chance to become customers (**T5**).

Summarizing, the analyst first found a hint that there is space for improvement in the *dimensions view*. Next, he used the *decision map* to find clusters of correlated values based on the accumulated decision instances. Finally, he uses external contextual information about business policies with respect to retaining students to find ways to improve the decision model.



a. b.

Figure 7.5: Car values correlated to quotes in an unexpected way. a The highlighted values show a clockwise trend for car values with low valued cars top right and high valued cars top left. b After removing the quote and performing some merging the same trend is still visible (though counter clockwise). Additionally, The triggered rules view reveal some cumulating discounts and an underrepresentation of experienced drivers.

7.4.2 Story 2: Why expensive cars get low quotes

The KPI of the insurance quoting decision-model (Fig. 2.1, $DM2$) is computed as follows. First, the sum Q of all calculated insurance quotes is computed. Next, for each insurance, two probabilities ψ_1 and ψ_2 are drawn from a car-accident distribution and a claim-value distribution respectively. ψ_1 and ψ_2 are combined to calculate the probable loss for the insurance, which is then subtracted from Q. Monitoring the trend of this KPI over time will show *whether* the business is profitable, but not *why*. Also, the KPI is less precise than it could be, given that the car-accident and claim-value distributions do not consider the particulars of certain insurance instances.

To increase insight on the business performance, the analyst decides to analyze the insurance quotes. For this, he bins the numerical attribute *vehicle.value* and *quote* as follows. The car value is binned in the same way as it is done in the production rules that determine the quote base premium (< 5500, 5500 - 11000, 11000 - 20000, 20000 - 35000, 35000 - 55000, $>= 55000$). Next, the analyst assumes that quotes are correlated to car values (**T6**). Hence, he bins the *quote* variable in proportional ranges to the binning of the *value* variable.

Next, the analyst selects properties of drivers, cars, and the quote variable. The *projection legends* show long tails of non contributing attributes. Brushing next points out that those are driver properties. This does not come as a surprise, as the analyst assumes that car properties, especially car value, are the most important factors for the quote. To clean up the plot, he now iteratively removes such unimportant variables, as explained for story 1 (Sec. 7.4.1). After removing the driver attributes from the plot, the analyst finds, to his surprise (**T4**), that there is no strong correlation between car value and quote. To learn how car values are distributed in the *decision map*, the analyst now clicks the "Car Value" bar in one of the *projection legends*. The gradient coloring shows its value for ordered attributes, such as age and car value in this particular case. As visible in Fig. 7.5a, a clockwise gradient (marked with the red arrow) shows low-priced cars placed top right; increasing car values in-between; and expensive cars top-left. Once this pattern is clear, the analyst now understands how spatial areas relate to car values. The *quote* attribute values, on the other hand, are almost all centered, and do not show a clear vertical or horizontal distribution. This observation is enforced by the *projection legends* which show that the *quote* attribute (green bar) does not contribute much to either f_1 (x axis) or f_2 (y axis).

The analyst now removes the *quote* variable from the plot and performs merging in order

to get clusters of related values (Fig. 7.5b). He notices that car options such as all-side airbags (concept island 1), anti lock brakes, and daytime running lights (concept island 2) correlate with the more expensive cars. At this point, he selects the concept islands related to expensive cars (islands 1, 2 and 3 in Fig. 7.5b). Now, he finds two interesting things in the *rule trigger view*: First, the rule that gives a discount for experienced drivers is underrepresented. For this selection, it is expected that this rule should trigger 312 times, but it is only triggered 206 times. Thus, expensive cars are less often driven by drivers that are considered experienced by the business logic. Secondly, the discount rules for daytime running lights and anti-lock brakes are overrepresented. The daytime running light discount is triggered 229 times times instead of the expected 100; and the anti-lock brakes discount is triggered 229 times instead of the expected 101. When the analyst selects each of these three concept islands individually, he finds two more overrepresented discounts: In concept island 3, the passenger-and-driver airbag discount rule is overrepresented; in concept island 1, the SUV discount rule is overrepresented.

From all the above, analysis the business analyst has now learned that:

1. The car value is, unexpectedly, not strongly correlated to the quote (**T3**, **T4**).
2. Expensive cars are less likely to be driven by experienced drivers (**T4**).
3. While the base quote calculation takes into account car values, expensive cars still get too low quotes due to an accumulation of discounts (**T4**).

As a result, the analyst has discovered a potential revenue leak in the business logic, which could be compensated, for instance by setting up some "luxury car" surcharge rule and using different calculations for the base quote (**T5**). This decision is subject to various checks with actuary departments and shall require further investigations before being implemented. Nevertheless, our tool shows here its potential for insight creation.

7.5 Conclusion and further evaluation

Using two scenarios based on a car insurance use case, we have shown how an analyst can combine the insight he gets by using the DEL with his own knowledge to find opportunities for improvement. Based on the insights he gains and his own experience he understands how he has to change the model to make it more effective for a specific population. Therefore, we have shown that the DEL provides a solution for the research questions related to these issues (**Q6** and **Q7**).

Due to earlier mentioned difficulties with respect to testing our framework in a production environment, the closest industrial evaluation of our toolset consisted in presenting our tool, and findings obtained with it, to various domain experts at IBM. The Decision Exploration Lab was therefore presented to IBM ODM's product managers and chief architects. As a preliminary validation of our work, we here quote some of the feedback we received:

Product Architect (1): "The Decision Exploration Lab would be a key asset for the ODM product to be able to assist our users with the right level of tooling and targeted analytics. From my point of view, in this proof of concept, the algorithms are sound and the visualization is great. We will need more use case investigations, refinement on the visual design and some task-based approaches to adapt the ease of use to the level of understanding of our users. But I'm sure this mid-point exists, all the more as our users, in these times of big data and business analytics, are becoming more advanced and less intimidated by analytics in general."

Product Architect (2): "The research presented in this thesis, opens perspective on decision analytics. Big data and analytics are areas of high interest for companies with the goal to better understand and optimize their business. The presented work makes innovative use of analytics, applied to operational decisions. This approach opens the door to better understand segmentation of data used in automated decisions. It provides insights about how a business rule based decision is coupled to request and outcome data. Such a analytical exploration of decisions is original and presents value to the IBM Operational Decision Manager customer base on the top of product capabilities.

In a smaller and quicker world speed in understanding your operations and optimize them is key for success. For an insurance company, decision analytics allow to measure the match between the customer, product and price segmentations in their decision operation. With these insights you are able to detect possible enhancements to improve your decision logic and operate with more efficiency. Most of ODM customers and prospects are looking in this direction and expect progress in a near future."

Senior product manager: "This thesis introduces a new analytical approach for improving IBM Operational Decision Manager (ODM) decision services. The main principle is to use past decisions to understand why a decision service is not preforming as expected or to discover how to improve business outcome by modifying business rules. The prototype that was developed was very convincing: it showed that this approach is intuitive for businesses users and does not require deep knowledge of the data analysis techniques that are used. This new concept will help IBM enhance existing ODM capabilities and allow customers to continuously improve their decision services using their past experience."

At the time of writing this thesis, there is ongoing work to prepare a field evaluation, which would serve various purposes. First, we want to apply the DEL in a different domain, one that is not known to us in order to verify that business analysts can apply the framework to their particular problem to verify expected and find unexpected functioning of a DM. Secondly, it will provide a deeper understanding of the current level of tooling that is used and point out how well our approach fits in the current environment of business analysts. Finally, we want to test and verify in particular our specific visual analytics approach with its target audience.

Chapter 8

Conclusion

> And what have you got at the end of the day?
> What have you got to take away?
> A bottle of whisky and a new set of lies
> Blinds on the windows and a pain behind the eyes
> Scarred for life - no compensation
>
> *Private Investigations*, MARK KNOPFLER

One of the key reasons behind the work of this thesis is the potential challenge related to automated massive execution of enterprise decisions with a Decision Management System (DMS): small unexpected side effects of individual decisions add up to a large combined effect if many decisions are taken. This might cause sub-optimal performance of a business decision at best and pose high financial risk to both the enterprise and society at worst. In response to this challenge, this thesis explored various ways to visualize and analyze the data sources of a DMS in order to help a business user to gain a deeper understanding of automated decisions.

8.1 Review of thesis contributions

The central problem addressed by this thesis is the development of a system which supports a business analyst in supervising automated decisions, in order to improve his understanding of the operating and effectiveness of such decisions.

In CHAPTER 2 we have introduced DMSs and the challenges that come with such systems. DMSs are suitable for automating highly repetitive enterprise decisions. To place our solution in a concrete context, we introduced examples of such decisions: eligibility for and quoting of car insurances. We detailed how a decision model consists of two parts. A domain model, which describes the concepts that are subject to the decisions and the decision logic, which describes how a decision is taken for a given business case. We have argued that a decision model cannot model the full complexity of reality and that therefore, a divergence might take place between the expected functioning of a DM and the actual results of decisions taken over time. We also have pointed out the similarities of comprehension tasks in the context of DMSs with program comprehension tasks.

As a first solution direction we presented the application of well-known Visual Analytics (VA) and Information Visualization (InfoVis) techniques (CHAPTER 4) for artifacts that come with a DMS. We have set a first step to further refine the treemap approach of Baudel [140], which visualizes both the domain model and the decision logic. Treemaps proved to be an interesting topic in itself because, we needed to adapt the layout for each type of data that we wanted to visualize with treemaps. A generic approach to visualization of treemaps was implemented in Discovery [139]. Our work led to a formal definition of this approach, expressed as a design space for rectangular space-filling layouts. In addition we presented a generic layout algorithm based on this design space for rectangular layouts, encompassing all treemap layouts as well as other rectangular layouts, such as 100% stacked bar charts and mosaic displays. Thus we refined a possible solution for the research question related to gaining insight in the structure of the Decision

Model (DM) (**Q1**). Additionally, we applied the well-known circular edge bundling technique of Holten [18] in the context of DMS. We used it to visualize the impact of domain model changes to the decision logic which addresses our research question on change impact (**Q2**). We concluded though, that these techniques, however interesting, are merely display techniques as opposed to problem solving techniques.

In CHAPTER 5 we presented a set of visualization and analysis techniques that allow for analysis of categorical data, which is common in decision data. The first technique, presented in Sec. 5.2, describes a visualization and a workflow for exploratory analysis of categorical data. In contrast to classical numerical MDS, we use Multiple Correspondence Analysis (MCA) to create 2D projections which display attributes, attribute values, and observations. We introduce several visual encodings which help correlating values, observations, and observations with values. We showed how our techniques can be used to find non-trivial insights with limited effort in a dataset from the insurance industry. These visualization and analysis techniques are a solution to address research question **Q3**, about gaining insight in the structure of the data. As the labels can be used to distinguish business case and decision data, we also have an initial solution for question **Q4**. Secondly, we presented a method for analyzing rule trigger patterns in Sec. 5.3. By extracting various metrics for the overall set of decisions and comparing those with metrics for a selection of decisions interesting differences can pop up that might help the analyst to understand the functioning of the DM. This method serves as a partial solution to research question **Q5**, how do we get insight in the logic for a given set of decisions.

In CHAPTER 6 we presented the Decision Exploration Lab (DEL), a visual analytics solution designed to address prevalent issues in the area of enterprise decision management. DEL is an end-to-end system which is integrated with IBMs Operational Decision Manager (ODM) and provides an exploratory work flow based on the data extracted from ODM. It provides two working modes to support the exploration process of a business analyst. The *verbal* mode provides search and filtering of decision for the specific context of ODM. The *visual decision exploration* mode refines and extends the techniques presented in CHAPTER 5. We refined the *dimensions view*, by adding check-boxes to the categorical attributes in order to support changing the selection of attributes under analysis in an interactive manner. The *decision map* refines and extends the *projections view* in various ways. Firstly, it uses a two-color scheme to distinguish business-case attribute values from decision attribute values. Secondly, it provides a new labeling mechanism, which complements the tool tips of the *projections view* in order to provide more information without interaction. Finally, it uses data dependent scales as a mean to make better use of the visual space in relation to the data distribution. The *decision map* is a refinement of the *projections view*, such that the focus is on the relation between business case and decision data. Therefore this technique now provides a better solution for research question **Q4**. Further refinements in the *decision map* by improved labeling and data dependent scales, result in more readable *decision maps*. Thus we improved our initial solution for the research question on data structure (**Q3**). Additionally, we integrated rule trigger analysis with the *decision map*, such that brushing the *decision map* gives rule trigger information about the decisions that match the brushed properties. This addresses the research question on the relation between business data and decision logic (**Q5**).

Lastly, we identified what it takes to evaluate a system for analyzing the aggregated effects of automated decisions in CHAPTER 7. We have shown that that there are large hurdles to overcome, especially when it comes to data and users. The required data must contain both a DM and a set of business cases that serve as input. For a useful evaluation both the business cases and the DM must consist of real data or at least realistic artificial data. The lack real data availability was partly overcome by performing an early user study using a publicly available dataset. In this early user study we found that the participants were able to use our techniques to

find interesting insights after a short introduction. However, this preliminary study was limited in various ways. Firstly, the participants in this study, mostly students of MSc or PhD level, were a different audience than we targeted in our final system, namely business users. Secondly, we used a dataset on education and income of US people, which could serve as business cases for a decision system. However, we had no corresponding business use case and DM that would make some kind of decision based on this data. These problems have led to the construction of a scenario containing both realistic data and a DM in the context of the car insurance industry. Using two scenarios based on a car insurance use case, we have shown how a business analyst can combine the insight he gets by using the DEL with his own knowledge to find opportunities for improvement. Based on the insights he gains and his own experience he understands how the model must be changed to make it more effective for a specific population of business cases. Therefore, we have shown that the DEL provides a solution for the research questions related to these issues (**Q6** and **Q7**). Additionally, these scenarios where presented to architects and product managers at IBM to verify that these where sufficiently realistic and that our approach solves an open field problem. At the time of writing, we are in the initial phase of setting up an actual field study with a customer to further verify our approach and to get feedback on how to improve the DEL.

8.2 Limitations and future work directions

A DMS provides a wealth of data, and the solution we provide is just a start in exploring this data. Additionally, there are similarities to be found with data outside the domain of DMSs. In this section we provide some directions for future research that where triggered by our work on DMSs.

8.2.1 DMS specific refinements

Our approach mainly exploits two kinds of data: the business cases (instances of the domain model) and execution traces. However, both the domain model and the decision logic also have structure which we do not exploit in the DEL. In CHAPTER 4, we have been using treemaps to visualize the structure of both the domain model and the decision logic, but merely in a static way. Combining static structure visualization with the dynamic execution data could give an even deeper insight in how the dynamics interplays with the modeled knowledge. Although this might not be of direct interest for the business analyst, this direction of research might help decision service integrators to identify runtime performance bottlenecks.

Moreover, over time data is gathered which is not modeled in the DM but is of tremendous help in understanding the dynamics of the DM. Examples of such data are events collected from the following system dynamics:

- A decision service decides that a person is eligible for a loan, however the person becomes a defaulter after a certain amount of time.
- A decision service does not mark certain credit card transactions as fraudulent, however investigation points out that in fact they were.
- A decision service marks an insurance claim as invalid, however after investigation it turns out that it was a valid claim.

Being able to model such external data and take it in account when analyzing the aggregated effects of a DM would bring large value in many applications of DMSs. When these kind of cases can be included in the analysis, it becomes easier to find where the decision logic is imprecise.

Additional challenges arise when such data are included. First of all, it might not always be straightforward to link such data to the individual decisions. For example, the total amount paid for fraudulent claims over a certain time period, cannot be linked to individual decisions. The question then becomes how such aggregated data, can be incorporated in the analysis in such way that it still provides useful information in the process of understanding the problem at hand. Secondly, the data are likely to be imprecise or incomplete, e.g. some fraudulent transactions might go undetected. These kinds of uncertainties should be taken into account in a way that prevents business analysts from drawing false conclusions.

8.2.2 Visualization refinements

The research of this thesis was performed in a commercial context, therefore we discuss some considerations with respect to making this research into a product or part of a product. In this thesis we focused on a particular problem, relating correlated values of the business case and decision data to the decision logic. Normally, this process takes place in a larger context that we more or less neglected. Therefore care should be taken in two areas: issues related to the design of our tool and integration of the tool in the wider context of decision management.

There are many smaller and larger improvements to our tool that could be thought of that would improve its ease of use. First of all, the choice for using Voronoi might not have been the best choice, as we have seen in informal discussions that people tend to start reasoning about the boundaries of a cell. Another graphical metaphor might be easier to explain and reduce the tendency to reason about graphical issues that have no particular meaning. Secondly, due to label overlap and the fact that clusters of values in the decision map are of more interest than individual values, an automated default setting for the merge slider would be helpful. One approach to this problem would be to choose a merge distance which more or less divides the space in equally sized cells. Thirdly, the decision map is based on the variables that are selected in the DataSpace tree (Fig. 6.4). In this thesis it was assumed that these variables are selected manually, however there might be means to make an automated initial selection. Making an automated selection could be based on static analysis of the decision logic by selecting only the variables which are tested against in a certain DM. Another approach could be to perform MCA on all variables and select the first $X\%$ that explain most of the variance, where X is chosen based on some heuristic. Finally, we took the variables as starting point of our analysis, while another approach could be to start at the rules. That is, given a certain rule or a set of rules, perform the analysis on all decisions that have triggered these rules.

With respect to placing our work in the broader context of decision management, some thought on the work flow is required. What kind of events trigger the exploratory analysis that we support? Sometimes the trigger is technically related to the DMS, such as a changing Key Performance Indicator (KPI) which is measured by the DMS. In these cases, thought should be given to how such an artifact influences the initial values of the exploration process, in order to focus the exploration immediately on the relevant areas. Another work flow related issue that we have not discussed is the process of turning insight into change of the DM. Further research should point out if the exploration process can be used to generate recommendations to the user on which parts of a DM should change. A difficult issue here is that such suggestions might involve knowledge that is not explicitly modeled.

The *projections view* is based on the MCA dimensionality reduction technique. However, not all attributes are categorical and, as we have seen in CHAPTER 6, we had to drop the observation projection due to the large amount of observations. Therefore, other dimensionality reduction techniques could be used as basis for the *projections view*. This would also require work on improving the work flow, as most techniques require many settings, and therefore complicate the analysis process.

In various ways uncertainty plays a role and we have not addressed the issue of uncertainty at all. Firstly, there is uncertainty in the projection. Are two projected points (either observations or attributes) close together because they are similar in the original data or is the proximity an unwanted side-effect of the dimensionality reduction technique? A second source of uncertainty is caused by multiple versions of the same DM. When a new version of a DM is put into production, it is normally first tested with a test set of business cases for which the outcomes of the old model are known. The new version might result in different outcomes, some of them may be expected others unexpected. There may be also decisions that result in the same outcome, while the outcome should have been different with the new model. Thus, the outcomes of the new model have a certain level of uncertainty. Further research should be carried out to find methods for visualizing these kinds of uncertainty. Besides the uncertainty in the outcomes of the new model, there is also difference in the correlation of the old and the new model. This brings up the question on how to visualize the difference of two almost similar datasets. One approach might be to extend our *projections view*, for example by overlaying the MCA results of the old dataset in the *projections view* of the new dataset MCA results and connecting corresponding attribute-value points with lines. Long lines will than indicate a large shift in the *projections view* and short lines small shifts. Another interesting problem related to this would be the differences in clusters in the *projections view* for various merge distances.

8.2.3 Collaborative analysis and knowledge engineering.

One of the important aims of DMSs is to bring the knowledge of various domains, such as law, risk analysis and marketing, under one hood. Therefore, it can be expected that people from these domains need to be involved when analyzing the functioning of a DMS. To this extent a fruitful research direction would be extending our system with and assessing the effectiveness of various collaboration techniques. A useful starting point for this direction can be found in the work of Heer and Agrawala [220, 221] who provide numerous design considerations for collaborative visual analysis. This research should also include modeling how a certain analysis trajectory lead to a certain change of the DM or why a particular change was not made.

8.2.4 Time analysis

The exact moment in time when a decision is made by a DMS is generally speaking not very interesting because the decision could be made as part of a batch processing session. However, many DMs contain fields that represent date or time. This is not generally true for each DM, because it depends on the business context and the type of decision that is being made by the DM. Therefore a fully generic solution might not be possible. Still, it would be interesting to see how the decision map can be extended to provide insight not just in the related concepts of a DM, but also how these correlations change over time. Means should be found to analyze the decisions with different level of time aggregations that are common in the specific business context such as week, month, quarter and year. Not only the evolution over time is interesting, but also the difference of correlation between two time batches (e.g. this year versus last year). Another approach would be to combine time series analysis to find bursts of events (see e.g. [222, 223, 224]), with the decision map.

8.2.5 Relational attributes

This mix of multivariate data and relational data can be found in many contexts. A data entity has besides multiple property attributes, also relational attributes. In many cases, relational attributes and properties can be considered as equivalent. However, there are relation attributes which we

cannot translate to properties. For example, lets take the relation of salary being a consequence of age and ethnicity. Salary is not a direct function of the two, but varies when age and ethnicity vary. Consequently, statistical methods for properties cannot be used for relational attributes. Additionally, relational *types* cannot be aggregated, e.g. what does it mean when three entities have relation type subclass and one entity has relation type function? Therefore, when a dataset has both a multivariate character and relational attributes, we need new methods and techniques have to be devised to aggregate properties with relational attributes in a meaningful way.

8.3 Closing remarks

Decision management systems are applied in a wide range of industries, among which health care, commerce, insurance, finance and transportation. These systems make millions and millions of decisions each day, impacting the life of millions of people and impacting economies at a large scale. At this scale, individual decisions have little or no impact, that is, one fraudulent transaction going undetected, one likely defaulter getting a loan or one customer getting a lower price than expected will have little or no impact on the business performance. It are the aggregated results of many decisions which lead to financial risk or missed opportunities. Therefore, the focus of the responsibility moves from the effects of individual decisions to the aggregated effects of thousands or millions of decisions. This shifted focus makes the development of new tools, which help gaining insight in the models behind and functioning of automated decisions, invaluable and necessary. With this thesis we hope to have made first steps in building a bridge between the visual analytics community, which has the knowledge and expertise to design and build such tools, and the industry.

List of own publications

It wouldn't happen... There hasn't been one publication by a monkey.

The Ricky Gervais Show, KARL PILKINGTON

Papers

- B. Broeksema and A.C. Telea. "Visual support for porting large code bases", In proceedings of 6th IEEE International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT), 2011, pp. 1—8, 29-30 Sept. 2011
- T. Baudel and B. Broeksema. "Capturing the Design Space of Sequential Space-Filling Layouts", In IEEE Transactions on Visualization and Computer Graphics, vol. 18, no. 12, pp. 2593—2602, Dec. 2012.
- B. Broeksema, A. C. Telea and T. Baudel. "Visual Analysis of Multidimensional Categorical Datasets", In Computer Graphics Forum, vol 32, pp. 158—169, Oct. 2013.
- B. Broeksema, T. Baudel, A. C. Telea and P. Crisafuli. "Decision Exploration Lab: A visual analytics solution for Decision Management", In IEEE Transactions on Visualization and Computer Graphics, vol 19, no. 12, pp. 1972—1981, Dec. 2013.

Posters

- T. Baudel and B. Broeksema. A generic algorithm for sequential, rectangular, space filling layouts. In Extended abstract and poster presentation, VisWeek 2011, October 2011.
- B. Broeksema and A.C. Telea. PortAssist: Visual analysis for porting large code bases. In Extended abstract and poster presentation, VisWeek 2011, October 2011.

Patents

- Focus-change invariance in a graphical display, US 20130127870 A1, T. Baudel and B. Broeksema
- Data plot processing, US patent, accepted, being processed, T. Baudel and B. Broeksema

Curriculum Vitae

Bertjan Broeksema is born on the 10th of April in Enschede, the Netherlands. He retrieved his bachelor degree in Computing Science from the Hanze University of Applied Sciences in Groningen, where he studied from 2003 to 2007. Afterward, he studied from 2007 to 2010 at the University of Groningen, where he obtained his master degree in Computing science. As part of his master program he did a minor in philosophy. During his studies he has been an active contributor to the open source KDE [1] project, where he among other things developed an interest for software maintenance. As such, he got in contact with KDAB [2] where he did his master internship on the topic of automated porting and where he worked as an software engineer afterward.

The work of his master thesis resulted in an article at the VISSOFT workshop. It was recognized that this topic had some overlap with problems at IBM relating to Decision Management Systems. Therefore, he started a PhD in November 2010 at IBM France on the topic of Visual Analytics for the support of such systems. Under supervision of A. C. Telea, T. Baudel and G. Melançon, he developed various InfoVis/VA techniques. His work was presented at the InfoVis and VAST conferences and in the Computer Graphics Forum journal. Some of these techniques where combined in an integrated system for the exploration of decision data. This system was developed in close co-operation with domain experts and evaluated with realistic use case scenarios.

In November 2013 he started working at the Centre de Recherche Public Gabriel Lippmann as a full-time researcher on Visual Analytics. His current research interests include space-filling layouts, visualization and analysis techniques for categorical data and VA for data streams and time series.

[1] http://www.kde.org
[2] http://www.kdab.com

Appendix A

User Evaluation of the MCAView visual analysis tool

This section contains a detailed report of the user evaluation that was held at the university of Groningen. The evaluation was conducted by A.C. Telea who compiled the results into this report.

A.1 Preliminaries

14 people were selected to evaluate the tool. 4 were PhD students. 7 were MSc students. 3 were BSc students in their final year before MSc. All were CS students (male). All had a minimal (1-2 years) exposure to interactive Infovis tools (mostly, in the form of table lenses, scatterplots, and bundled graphs). All were familiar with tabular data with numerical and categorical data. However, except 2 of the PhD students, none was familiar with MDS, MCA, or parallel coordinates. The 2 PhDs mentioned were only familiar with MDS at a general level (dimensionality reduction principles). None of the students was experienced with insurance data. All were volunteers, not paid or rewarded in other ways for their work in the evaluation.

A.2 Way of working

The students completed the following steps:

A.2.1 Introduction

During this step, a staff member (co-author of the paper) held a presentation session with all students. In this session, he presented the main idea:

We are given a table containing a dataset with numerical and categorical attributes. We would like to answer several questions using a visual tool. Your role is to use the tool to answer the questions, document/explain your findings, and document any other observations on the tool (good/bad features, how much you used each feature, proposals for improvement).

A high-level explanation of the end purpose of such analyses was given: We have a car insurance dataset. Insurance agents want to find patterns (groups of people sharing some common characteristics). Based on such patterns, they ultimately want to tune their insurance policies to maximize revenues and minimize risks. The purpose of this explanation was to make users aware of the added-value of the proposed tool/technique in an actual application domain.

This step took about 10 minutes.

A.2.2 Data presentation

During this step, the presenter loaded the raw tabular data in the tool and executed two binning scenarios (with 3, respectively 4 bins). The students were told that binning is not part of their further assignment, and were instructed to note down the binning settings so they can reuse them. The resulting binned table was scrolled (by the presenter on a laptop connected to a beamer).

The presented commented to make the participants familiar with the columns, attribute names, and attribute values. This also gave a rough idea of the complexity and size of the raw data.

To check that the participants understood the table, a few questions were asked to the public, e.g. How many values has attribute daytimeRunningLights? Where can you see this? What means quote: 620..929? The students answered the questions quickly (under 1 minute), and showed they do understand the data. One attribute (DUI) was left unexplained, and the participants were told that this one is not important. Also, the participants were told that they do not need to know by heart all US state names (50 values).

This step took about 15-20 minutes.

A.2.3 Visualization presentation

During this step, the presenter explained all four views of the tool, in this order: dimensions view, projections view, observations plot, and merge/filter options. For each view, the following points were explained:

- color mapping: what do colors mean
- structure of the view: tree, scatterplot, diagram, and what these mean
- interaction options (click and brush)
- link to the other views (either by color mapping or selection)

The main part of the explanation covered the interpretation of elements in the views along the following lines:

- **dimensions view**: colors = attribute identities; tree = values of some category;
- **projection view**: colors = attribute identities; Close plotted points for values from different attributes imply that observations tend to select these values together. Close plotted points for different values of the same attribute imply that observations selecting either of these values are similar with respect to the other attributes. Meaning of Voronoi cells = closest points to a given site;
- **observation plot**: colors = values of a selected attribute; Two observations plotted close to each other imply that they have similar attribute values or, for categorical data, that they share several attribute values (since two categorical values can either be equal or different);
- **projection view and observation plot**: the x,y axes do not mean something obvious (like a given attribute). Only relative positions (distances) do mean something;
- **barchart legends**: length of a bar equals the variance of that attribute on a respective axis. In other words, big bars are attributes which are well explained along an axis (or not explained on any of the axes, for big bars in the error plot).

The explanations were supported by simple examples (brush/select item, explain what happens in the visualization) demonstrated publicly with the tool to the audience.

This step took roughly 20 minutes.

A.2.4 Assignment

After the training, an assignment was distributed to the students (see Sec. 3). The participants were told to read the assignment, and that they have to execute it during a separate session over a few days. The participants could use, on their own laptops, a binary of the tool compiled for Microsoft Windows or a binary for Mac OS X, out of the box (i.e., with zero installation effort). Both the binary and dataset were distributed to the participants.

The overall training was concluded by a Q&A section (5 minutes), where the participants could ask any clarification questions. Globally, all participants said that they believe they understand the purpose of the tool and assignment.

A.2.5 Experience sharing

After executing the assignment, which took 2 hours, all participants met the organizer again (separately, that is, not as a group). This took roughly 20 minutes per participant. Each participant explained the results, and commented freely on various findings. Some participants brought their laptops to show the results or snapshots of the tool usage. The organizer noted the feedback, but refrained from judging (aloud) the findings, in order not to bias the participants exposition. The aggregated feedback is presented next in Sec. 4.

A.3 Assignment

You are given a dataset containing car insurance quotes for various people living in the US. The structure of the data has been explained previously. Using MCAView, the tool presented to you earlier, complete the following tasks and answer the respective questions, in the order specified below.

Q1: Attributes and their values

Start the tool, load the data, and select the projections view. Using the earlier explanations, try to find a meaning for groups of cells in the plot to the right, left, top, bottom, and center. For each group such identified:

- describe the perceived meaning of the group (what kind of people share such attributes?) in one sentence; if you do not find any specific meaning for such a group, explain why. Answer: plain text
- Which techniques did you use? Mark them by perceived utility and ease of use (very high, high, low, very low, not used):
  - color linking
  - brushing and selection
  - barchart legends
  - merging/filtering

  Answer: multiple-choice table

Hints:

- recall that, in the projection view, close plotted points for values from different attributes imply that observations tend to select these values together. Close plotted points for different values of the same attribute imply that observations select either of these values and are similar with respect to the other attributes.
- brushing cells reveals more information on a given attribute value
- colors are explained in the dimensions view
- you can use the merge/filter view to simplify the projections view

Q2: Explaining the axes

Start the tool, load the data, and select the attribute plot view. Next, try to find an explanation, in terms of attributes, for the x and y axes. Answer the following questions

- what is the perceived meaning of the x axis? What kind of people do we have in the cells to the left, and to the right? Answer: plain text
- what is the perceived meaning of the y axis? What kind of people do we have in the cells to the top, and to the bottom? Answer: plain text
- how sure are you about the interpretation? Explain why. Answer: plain text
- which techniques did you use? Mark them by perceived utility and ease of use (very high, high, low, very low, not used):
  - color linking
  - brushing and selection
  - barchart legends
  - merging/filtering

  Answer: multiple-choice table

Hints:

- recall the meaning of barchart legends: length of a bar equals the variance of that attribute on a respective axis. Big bars are attributes which are well explained along an axis (or not explained on any of the axes, for big bars in the error legend).
- you can use merging and filtering to eliminate attribute values which you consider irrelevant
- one axis may require several attributes to explain

Q3: Understanding observation clusters

Start the tool, load the data, and select the observation plot view. You see a scatterplot of the observations (people). Answer the following questions:

- do you see any clusters (groups of close points which share some characteristics, and are reasonably well separated from other groups of points)? If so, explain the clusters you see in 1-2 sentences: What kinds of points are in each cluster? Answer: plain text

- which techniques did you use? Mark them by perceived utility and ease of use (very high, high, low, very low, not used):
  - color linking
  - brushing and selection
  - barchart legends
  - merging/filtering

  Answer: multiple-choice table

Hints:

- you can color observations by the value of one categorical attribute; attributes can be selected either directly from the dimensions view or using the barchart legends
- observations are plotted in the same space as attributes (revisit answers to Q1, Q2)
- there may be several ways in which you identify clusters in the data; try to find the most clear-cut separation (clustering), with clearly delimited clusters, and little overlap between clusters

A.3.1 Q4: General questions

- how much time did you spend on the whole assignment? Answer: number
- rank the tools functions in terms of utility and ease of use (very high, high, low, very low, not used)
  - dimensions view
  - attributes plot
  - observations plot
  - color linking
  - merging / filtering
  - x and y barcharts
  - error barchart
  - tooltips and selection

  Answer: multiple choice table
- which suggestions for improvement or other remarks do you have? Answer: free discussion (20 minutes)

A.4 Results

Of all 14 participants, 13 completed the full assignment. One BSc participant dropped off (apparently, not enough motivated/interested).

Below we show the aggregate results of Q1..Q4. Note that feedback was given in various ways (on paper, orally, by means of snapshots and tool live demos).

A.4.1 Q1

Describe the perceived meaning of the group:

8 of 13 found the right group of expensive vehicles as an outlier. They described it in various terms (expensive vehicles; high-end cars; limos) or simply by listing the attributes of these vehicles.

7 of 13 found the left group of lower-end vehicles. They described it in terms such as (family cars; cheap cars) or by listing the attributes of the vehicles.

8 of 13 found the top group of older people. They described it as (family people; older people; working people).

5 of 13 found the bottom group of younger people. They described it as (students; young people), with no extra qualification.

7 of 13 characterized the central group as the average people. **6 of 13** could not find any salient characteristic of this group (i.e., they did not reason that this group has to be the average people since the peripheral groups are outliers).

2 of 13 found the group of close cells related to the numAccidents attribute (3 greenish cells). The given interpretations were people that tend to have accidents share some [unknown] pattern and people that have one accident will have more next (this is, in my view, a wrong interpretation of the data).

Which techniques did you use? Mark them by perceived utility

- color linking: 7 (very high); 4 (high); 2 (low);
- brushing and selection 10 (very high); 3 (high)
- barchart legends 10 (not used); 2 (low); 1 (very low)
- merging/filtering 2 (high); 6 (low); 3 (very low); 2 (not used)

Notes

Overall, it is interesting to see that the majority of participants succeeded in finding the cell groups pointed into the assignment description and describe these in roughly similar terms to our own description (explained in Sec. 5.2). This supports the claim that the visual presentation used does not offer too much space for ambiguity.

A deeper evaluation would have relaxed the problem statement, e.g., not indicate that one has to explain groups of outlier cells (top, bottom, right, left) and average cells (center), but simply ask which are the groups of strongly-correlated cells, and what these mean. We did not conduct this more ambitious test here, as our intention was more defensive, i.e. first evaluate if, using the same dataset and tool, other people could replicate our findings. Such a more involved evaluation is planned, now that we have initial feedback that helps us into fine-tuning both the tool and evaluation procedure.

A.4.2 Q2

What is the perceived meaning of the x axis? What kind of people do we have to the left, and to the right?

9 of 13 found that the x axis somehow relates to the social position of persons (wealthy vs poor; entrepreneurs vs family people; rich vs non-rich). One person labeled the observations to the right as conservative people (because they have the daytime running lights on, and they tend to have more expensive vehicles). He also noticed that the presence of antilock brakes seems to be

correlated with daytime running lights. The other 3 participants did not find a salient meaning of the x axis. [1]

What is the perceived meaning of the y axis? What kind of people do we have to the left, and to the right?

8 of 13 found that the y axis somehow relates to family (married vs non-maried; older vs younger; studying vs not studying). Two persons said that they revisited their answers to Q1 after they had to complete Q2, since Q2 helped them (indirectly) to finding clusters at the top and at the bottom. 3 participants did not find a salient meaning for the y axis.

How sure are you about the interpretation? Explain why.

8 of 13 said that they are reasonably sure of their interpretation of the axes because they used different methods to arrive at their conclusion (brushing, color linking, displaying the observations colored by the value of an attribute). Of these 5 said that, after initial brushing and color linking, they started looking at the barcharts, to locate big bars for the x or y axes, and then tried to give a meaning to the axes based on the semantics of these bars. Only 3 participants said that they also considered the error barchart when interpreting the x and y barcharts.

2 of 13 said that they think their interpretation is correct, but are not sure (I think this is the meaning of the x/y axes, but I have no further hints as to whether this is correct; at least this is the most plausible meaning for me). The reasons for confusion stated here were the many state cells which apparently create noise in the data. Both persons also said that they thought that there are some specific state-related semantics (people in state X do Y) which they should use, but not being familiar with the US, they got blocked on this thinking path.

3 participants did not find a salient meaning for the axes, as noted earlier. Among reasons for this they noted also the confusion created by the state cells (what does it mean if I see all these state cells here?)

Which techniques did you use? Mark them by perceived utility (very high, high, low, very low, not used):

- color linking: 6 (very high); 5 (high); 2 (low).
- brushing and selection: 9 (very high); 2 (high); 1 (low)
- barchart legends: 1 (very high); 4 (high); 5 (low); 1 (very low); 2 (not used)
- merging/filtering: 3 (high); 6 (low); 3 (very low); 1 (not used)

Notes

Interestingly, color linking and brushing-and-selection are perceived to be roughly as useful for Q2 as for Q1. Also, we see now more people trying to use merging/filtering than for Q1, which can be explained as a learning effect (people get more confident with the tool and thus try the more complex options). The strongest note, though, regards the much larger use of barcharts as compared to Q1. This can be partially explained by the hint in the text of Q2 as to using barcharts, and also the learning effects mentioned earlier.

[1] Note that most participants (9 of 13) had a Dutch background. In the Dutch culture, there is a tendency of simplifying finance-related issues into rich and poor, e.g., labeling persons who own a larger house, car, or boat, as rich or upper class, regardless of other reasons which may explain the ownership.

A.4.3 Q3

Do you see any clusters (groups of close points which share some characteristics)? If so, explain the clusters you see: What kinds of points are in each cluster?

10 of 13 participants answered this positively, i.e., did find clusters. 3 others said they did find clusters, but are not sure which ones are right (i.e., are true well separated clusters). Interestingly, no-one ended this question with not finding any clusters.

Among the clusters reported: **10 of 13** did find the student vs non-student cluster as being the clearest one. The other 3 did find this one too, but for some reason wanted to find more complex, less obvious, clusters further in the data. Among mentioned candidates were daytimeRunningLights on-vs-off; airbagStatus (none vs all-other-options); married (false vs true); and ageInYears (value=54..72), which was also found as a well-formed cluster. Apparently, the intention was to find clusters which arent explained by a single attribute, but more attributes. This is explainable, since during the introductory presentation, it was mentioned that clusters may (need to) be explained by more than one attribute.

- Which techniques did you use? Mark them by perceived utility (very high, high, low, very low, not used):
- color linking: 11 (very high); 2 (high)
- brushing and selection: 4 (very low); 3 (low); 3 (high); 2 (very high); 1 (not used)
- barchart legends: 5 (very high); 5 (high); 3 (low)
- merging/filtering: 10 (not used); 3 (very low)

Notes

This task was completed well by all participants. This may be explained by the fact that finding/explaining clusters in scatterplots is something the participants are somehow more familiar with.

Interestingly, color linking was seen the clear winner technique here. This is explainable, since coloring observations by attribute values makes it relatively clear which observations have which value, so helps seeing clusters determined by such values.

Also, barchart legends scored very well here. The participants reported that they first tried random clicking on some bars, or clicking on bars from left to right in the order the attributes are listed in the dimensions view, after which they realized that selecting the longest bars is the most efficient, since such bars are most likely to show attributes that explain clusters. They also noticed that selecting short bars (in x,y) or long bars (in the error plot) typically creates (complete) noise, i.e. no clusters, which made them further focusing on the long bars. 7 of 13 students noted that they realized that they must use all the legends together. Of these, 4 said they made explicit use of the error barcharts.

Example storyline

A nicely commented discussion was supplied by one participant (see snapshots below), as a story-line for searching for clusters. The text reflects his notes, taken snapshots, and also face-to-face discussion. The text is translated from Dutch (participants native language) to English.

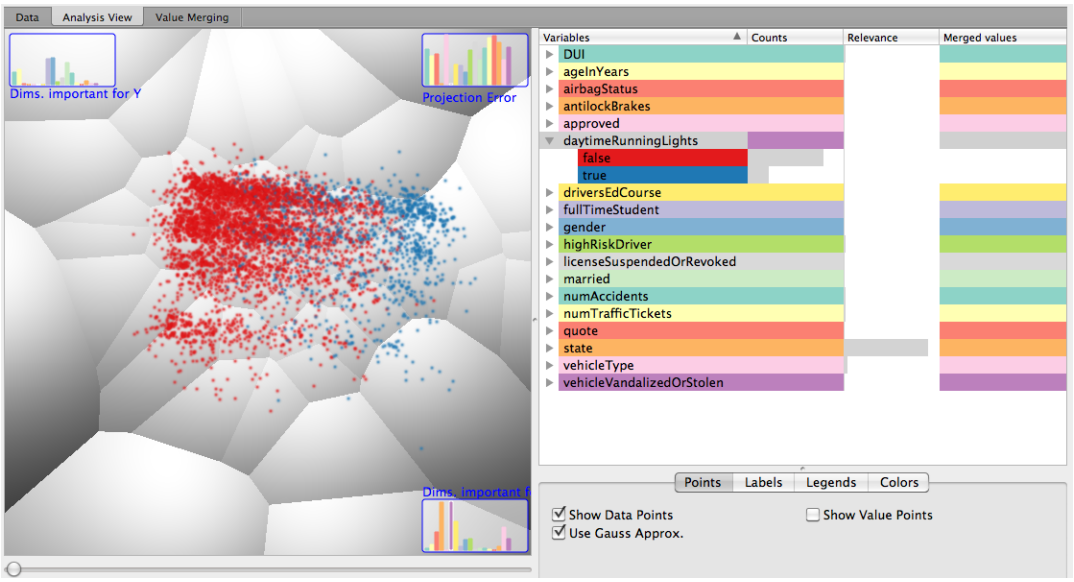


Figure A.1: After playing around a bit, I selected the daytimeRunningLights bar in the x plot, since its such a long one. There seems to be a cluster of lights=on to the right, and lights=off to the left. Wait, but I already knew this from answering Q1 and Q2. This is good, but it isnt a clear cluster separation, so I search(ed) further.

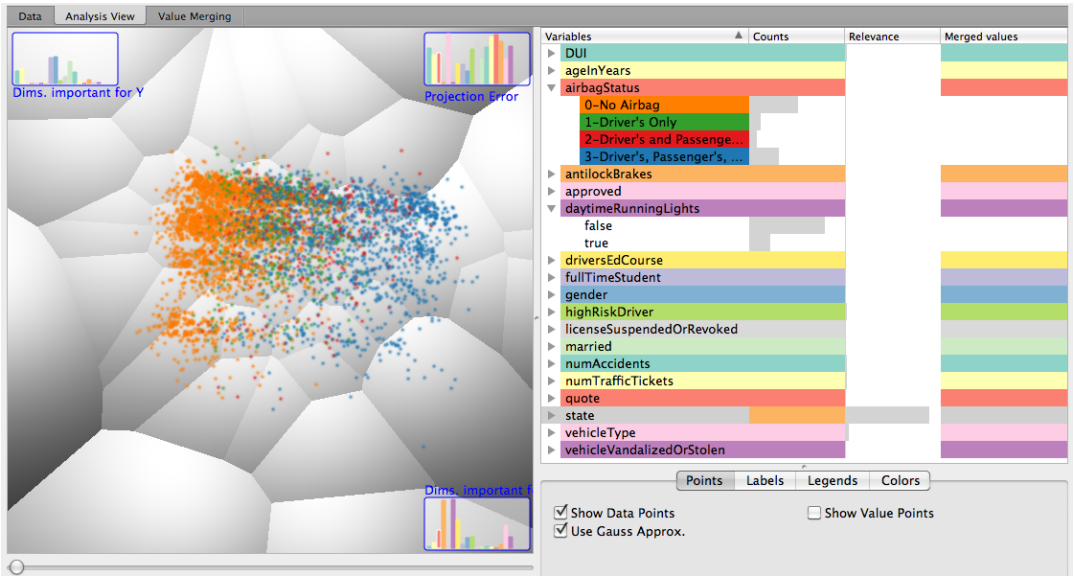


Figure A.2: I next selected the airbagStatus attribute in the x barchart, since this one also appeared in my analysis for Q1 as an outlier. I see now a clear(er) cluster to the left (no airbag), and a less clear cluster to the right (airbags everywhere). This seems to correlate nicely with the previous analysis people at the right are more conscious, drive safer, and people to the left are less safety-interested.

A.4.4 Q4

How much time did you spend on the whole assignment?

One hour (7 of 13); between 1 and 2 hours (4 of 13); 30-40 minutes (2 of 13)

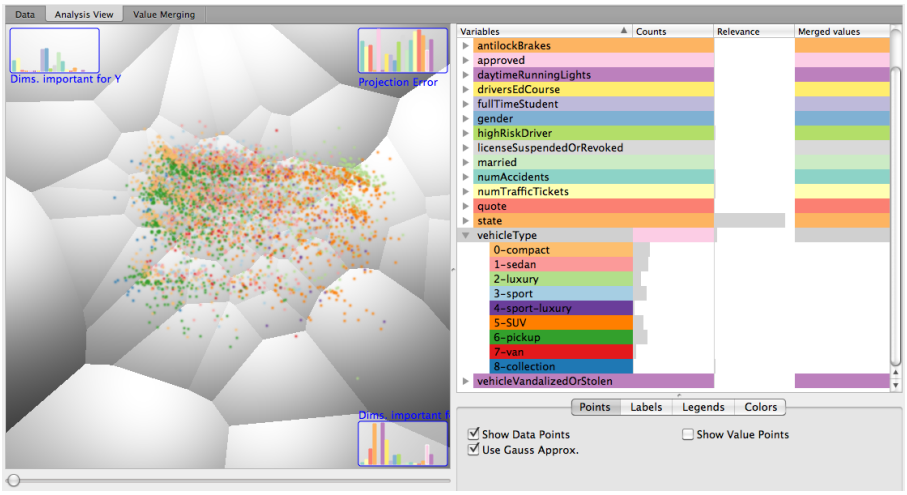


Figure A.3: I next selected the vehicleType in the x barchart. Hmm, this shows more or less noise. There appear to be some horizontal bands but these arent related to the vehicle type (all colors appear more or less everywhere), so people [observations] cannot be clustered based on vehicle type. Indeed, this makes sense all kinds of people buy all kinds of cars.

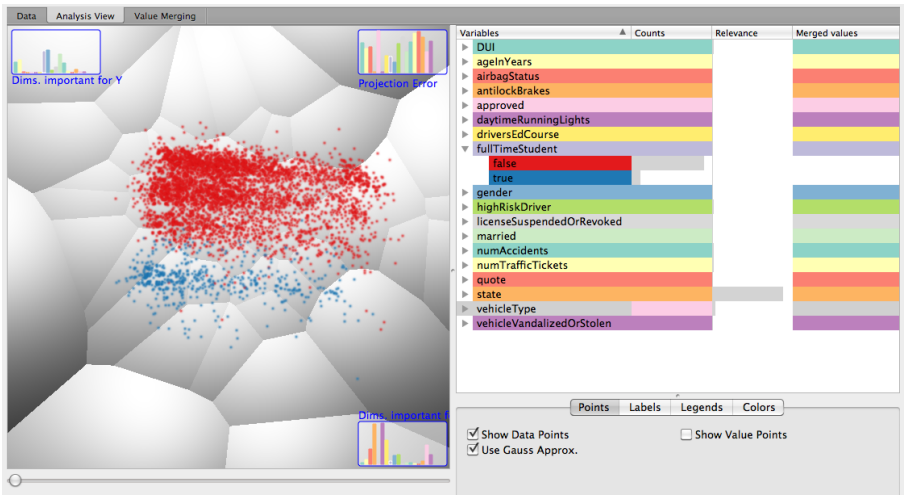


Figure A.4: Apparently, theres only so much I can get from the x axis. Now I selected fullTimeStudent in the y barchart (its a pretty large bar; oh and it is short in the x axis barchart). Wow, clear! I see a cluster of students and one of non-students. Indeed, students are at the bottom, which I knew from Q1. And now I see that people [observations] are denser in the left part of the plot, which may make sense I recall that to the right we had richer people, and those should be less than ordinary people.

Rank the tools functions in terms of ease of learn and use (very high, high, low, very low, not used)

- dimensions view: very high (10 of 13); high (3 of 13)
- attributes plot: high (8 of 13); very high (2 of 13); low (2 of 13); very low (1 of 13)
- observations plot: high (7 of 13); very high (4 of 13); low (2 of 13)
- color linking: high (7 of 13); very high (5 of 13); low (1 of 13)

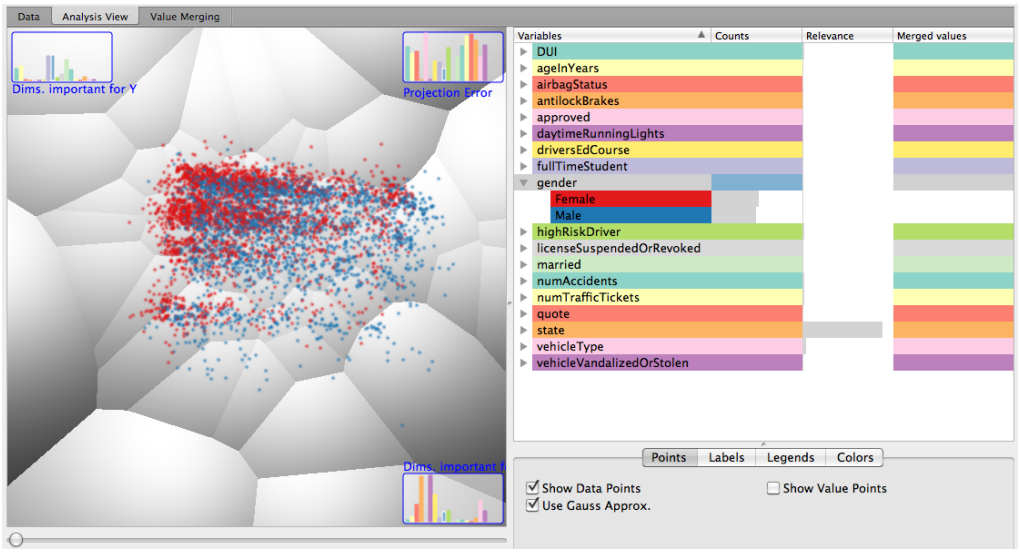


Figure A.5: Now that were at the y barchart, theres a second large bar here for gender. Let me select this one. Hm, I think I see two overlapping clusters, somehow the male are shifted a bit more to the right could this mean that they are richer? Maybe not. In any case, it seems that there are no clear gender-related patterns to be found here.

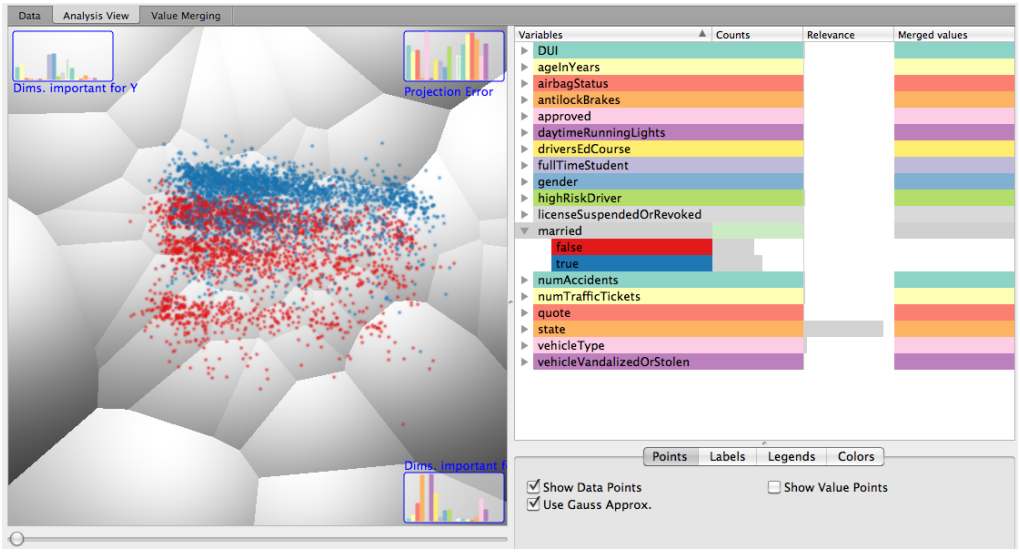


Figure A.6: OK, the next large bar in the y barchart is for married. Looks very much like the pattern for the gender attribute. Yes, but there is some correlation here older people are clearly more often married than younger ones, and students (lower cloud) are mostly not married. Indeed. And just as for vehicleType, they seem to come as both rich and non-rich.

- merging / filtering: high (2 of 13); low (8 of 13); very low (3 of 13)
- x and y barcharts: very high (4 of 13); high (5 of 13); low (4 of 13)
- error barchart: high (3 of 13); low (8 of 13); very low (2 of 13)
- tooltips and selection: high (6 of 13); very high (4 of 13); low (3 of 13)

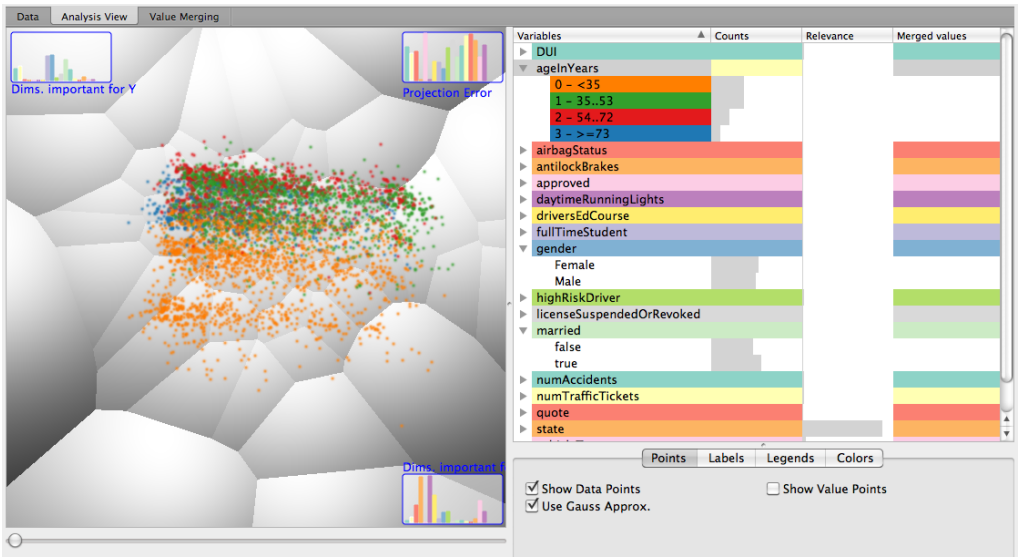


Figure A.7: Now for the last one: I select the third-largest bar in the y barchart, age. OK, I see two clear clusters: young people (orange, at the bottom) and old people (red, at top). The other age categories are fuzzier overlapping clusters in between. Not surprising. Concluding the whole exercise: If I have to name a clear separation, or clear clusters, Id say this is the lower cloud, containing young, studying, and unmarried people. These seem to be quite different from the rest. There also seems to be a cluster of safety conscious, richer, people to the right, and one of less safety conscious, average, people to the left. These are my main findings.

Which suggestions for improvement or other remarks do you have?

- The tool would be much easier (and nice) to use if it were faster!
- I could learn the barcharts quite easily. It is a nice, helpful, idea.
- The observation plot was easier to learn and use than the projections view. The explanation of the latter could be done better, so it presents its data in some more intuitive form.
- It would be useful to somehow add labels to the barchart legends, and allow to sort them on size. Otherwise finding those long bars takes time. Also a zooming mechanism would be good, I want often to focus on the longest bars, and dont really care about the shortest ones.
- It would be nice to provide some more refined mechanisms to help finding clusters faster (something like a wizard with guided steps).
- The tool crashes sometimes when doing the merging.
- The Voronoi plot is a bit slow when it renders again (on my computer, it takes under 1 second, but having something real-time would be less disruptive)
- Overall, the tool is interesting, and fun to use. I did not think you can look at table data in this way. Once you learn the basic mechanisms, it keeps you engaged playing with it to find more facts about the data.
- When merging, what would be nice, is to have some smooth animation from the previous [non-merged] state to the merged state. In this way, I could better see what changed when

I merge. Right now, its still OK since merging works in real-time, so I can play with the distance slider to-and-fro to see what happened, but an animation would be even faster.

- What I liked about the tool, is that I can use it with no configuration or tweaking options, its just plug-and-play once I have a table saved in a text file. Thats easy. Maybe Ill use it on some other tables I have.

A.4.5 Threats to validity

Actual tool usage time: Some students may have actually spent more than 2 hours on working with the tool. This is hard to exactly measure, since, for logistical reasons, we had to distribute the tool and dataset upfront, to make sure that all participants could install and run it with no issues on their laptops. However, knowing our student population (from earlier user experiments of the same kind, and from related practical assignments), this is a very low risk the students have a strong tendency of not working more than asked for, especially if there are no incentives (bonus, exam points) given for this work.

Questions: The questions could have been made more precise. The reason we structured the questions as described, is that we tried to make them in line with our own findings and experiences with the involved dataset, so we could next assess the value/validity of the produced answers. However, some extra refinement would have helped. There is, for instance, a (subtle) overlap between Q1 and Q2 which could have been eliminated.

Merging: We noticed that merging was used very sparingly. This could be because it was also explained the least during the introductory session. A more involved introductory session, with more time dedicated to this option, might have changed the results, possibly in favor of merging.

Participants: There was clearly some amount of correlation between the answers and intellectual level of the students. We noticed that the smarter students (assessed by their overall grades and performance during other activities in our study) did perform overall better, and also found the tool better. As such, the participant population was quite mixed. Another issue involves mixing students of three quite different levels (BSc, MSc, PhD) in the same evaluation. A better strategy would have selected a more uniform group of people.

Scoring: The multiple choice options used to score the ease-of-use and utility of the different techniques could have been refined further

- Split ease-of-use and utility in two separate questions (something can be easy to use, but not useful, and conversely)

- Refine the 5-point scale into (very high, high, not high or not low, low, very low, not applicable). We did not choose the neutral mid-point (not high or not low) as we tried to avoid too many neutral answers. Still, this choice introduces a bias in the evaluation (one has to decide if something is useful/usable or not).

Bibliography

[1] J. LeBlanc, M. O. Ward, and N. Wittels, "Exploring n-dimensional databases," in *Proc. of the IEEE Conf. on Visualization*, ser. VIS '90, 1st. Los Alamitos, CA, USA: IEEE Computer Society Press, 1990, pp. 230–237. [Online]. Available: http://dx.doi.org/10.1109/VISUAL.1990.146386

[2] J.-F. Im, M. McGuffin, and R. Leung, "Gplom: The generalized plot matrix for visualizing multidimensional multivariate data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2606–2614, 2013. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2013.160

[3] R. Amar, J. Eagan, and J. Stasko, "Low-level components of analytic activity in information visualization," in *Proc. of the IEEE Symp. on Information Visualization*, ser. InfoVis '05, 11th, 2005, pp. 111–117. [Online]. Available: http://dx.doi.org/10.1109/INFVIS.2005.1532136

[4] J. Taylor and N. Raden, *Smart Enough Systems*. Prentice Hall, 2004.

[5] F. Frasincar, A. C. Telea, and G.-J. Houben, "Adapting graph visualization techniques for the visualization of rdf data," in *Visualizing the Semantic Web*, V. Geroimenko and C. Chen, Eds. Springer London, 2006, pp. 154–171. [Online]. Available: http://dx.doi.org/10.1007/1-84628-290-X_9

[6] R. Lu and S. Sadiq, "A survey of comparative business process modeling approaches," in *Business Information Systems*, ser. Lecture Notes in Computer Science, W. Abramowicz, Ed. Springer Berlin Heidelberg, 2007, vol. 4439, pp. 82–94. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-72035-5_7

[7] R. Ko, S. Lee, and E. Lee, "Business process management (bpm) standards: a survey," *Business Process Management Journal*, vol. 15, pp. 744–791, 2009. [Online]. Available: http://dx.doi.org/10.1108/14637150910987937

[8] F. Niedermann and H. Schwarz, "Deep business optimization: Making business process optimization theory work in practice," in *Enterprise, Business-Process and Information Systems Modeling*, ser. Lecture Notes in Business Information Processing, T. Halpin, S. Nurcan, J. Krogstie, P. Soffer, E. Proper, R. Schmidt, and I. Bider, Eds. Springer Berlin Heidelberg, 2011, vol. 81, pp. 88–102. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-21759-3_7

[9] (2013) Operational decision manager. IBM. [Online]. Available: http://www-01.ibm.com/software/decision-management/operational-decision-management/websphere-operational-decision-management

[10] (2013) Drools, business logic integration platform. JBoss. [Online]. Available: http://www.jboss.org/drools

[11] (2013) Blaze advisor business rules management. FICO. [Online]. Available: http://www.fico.com/en/Products/DMTools/Pages/FICO-Blaze-Advisor-System.aspx

[12] (2013) Business process model and notation. The Object Management Group. [Online]. Available: http://www.bpmn.org

[13] A. Chniti, S. Dehors, P. Albert, and J. Charlet, "Authoring business rules grounded in owl ontologies," in *Semantic Web Rules*, ser. Lecture Notes in Computer Science, M. Dean, J. Hall, A. Rotolo, and S. Tabet, Eds. Springer Berlin Heidelberg, 2010, vol. 6403, pp. 297–304. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-16289-3_25

[14] N. Fuchs and R. Schwitter, "Specifying logic programs in controlled natural language," *Computing Research Repository*, vol. abs/cmp-lg/9507009, 1995. [Online]. Available: http://arxiv.org/abs/cmp-lg/9507009

[15] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eriksson, N. F. Noy, and S. W. Tu, "The evolution of protégé: an environment for knowledge-based systems development," *Int'l. Journal of Human-Computer Studies*, vol. 58, no. 1, pp. 89–123, 2003. [Online]. Available: http://dx.doi.org/10.1016/S1071-5819(02)00127-1

[16] H. Knublauch, R. W. Fergerson, N. F. Noy, and M. A. Musen, "The protégé owl plugin: An open development environment for semantic web applications," in *The Semantic Web*, ser. ISWC '04, Lecture Notes in Computer Science, S. McIlraith, D. Plexousakis, and F. Harmelen, Eds. Springer Berlin Heidelberg, 2004, vol. 3298, pp. 229–243. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30475-3_17

[17] (2013) Protégé visualization plugins. Stanford Center Biomedical Informatics for Research. [Online]. Available: http://protegewiki.stanford.edu/wiki/Visualization

[18] D. Holten, "Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 741–748, 2006. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2006.147

[19] C. Forgy, "Rete: A fast algorithm for the many pattern/many object pattern match problem," *Artificial Intelligence*, vol. 19, no. 1, pp. 17–37, 1982. [Online]. Available: http://dx.doi.org/10.1016/0004-3702(82)90020-0

[20] D. P. Miranker, "Treat: A better match algorithm for ai production systems; long version," University of Texas at Austin, Tech. Rep. AI TR87-58, 1987.

[21] D. A. Keim, J. Kohlhammer, G. Ellis, and F. Mansmann, Eds., *Mastering The Information Age - Solving Problems with Visual Analytics*. Eurographics, November 2010.

[22] S. Diehl, *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. Berlin: Springer, 2007.

[23] W. S. Cleveland, *Visualizing Data*. Summit, New Jersey, U.S.A.: Hobart Press, 1993.

[24] S. K. Card, J. D. Mackinlay, and B. Shneiderman, *Readings in information visualization: using vision to think*, ser. Interactive Technologies. Morgan Kaufman publ. Incorporated, 1999.

[25] C. Ware, *Information Visualization: Perception for Design*, ser. Interactive Technologies. Elsevier Science, 2004.

[26] C. Chen, *Information Visualization: Beyond the Horizon*. Springer, 2004.

[27] R. Spence, *Information Visualization: Design for Interaction*. Pearon/Prentice Hall, 2007.

[28] S. J. Russell, P. Norvig, J. Candy, J. Malik, and D. Edwards, *Artificial intelligence: a modern approach*. Prentice Hall, 1996.

[29] (2013) Spss. IBM. [Online]. Available: http://www-01.ibm.com/software/analytics/spss/

[30] (2013) R. The R Foundation. [Online]. Available: http://www.r-project.org/

[31] (2013) Stata. StataCorp LP. [Online]. Available: http://www.stata.com

[32] (2013) Matlab. The Mathworks Inc. [Online]. Available: http://www.mathworks.com/products/matlab/

[33] X. Wang, W. Dou, T. Butkiewicz, E. A. Bier, and W. Ribarsky, "A two-stage framework for designing visual analytics system in organizational environments," in *Proc. of the IEEE Conf. on Visual Analytics Science and Technology*, ser. VAST '11, 2011, pp. 251–260. [Online]. Available: http://dx.doi.org/10.1109/VAST.2011.6102463

[34] A. Savikhin, R. Maciejewski, and D. Ebert, "Applied visual analytics for economic decision-making," in *Proc. of the IEEE Symp. on Visual Analytics Science and Technology*, ser. VAST '08, 2008, pp. 107–114. [Online]. Available: http://dx.doi.org/10.1109/VAST.2008.4677363

[35] A. Savikhin, H. C. Lam, B. Fisher, and D. Ebert, "An experimental study of financial portfolio selection with visual analytics for decision support," in *Proc. of the Hawaii Int'l. Conf. on System Sciences*, ser. HICCS '11, 44th, 2011, pp. 1–10. [Online]. Available: http://dx.doi.org/10.1109/HICSS.2011.54

[36] S. Al-Hajj, I. Pike, B. Riecke, and B. Fisher, "Visual analytics for public health: Supporting knowledge construction and decision-making," in *Proc. of the Hawaii Int'l. Conf. on System Sciences*, ser. HICCS '13, 46th, 2013, pp. 2416–2423. [Online]. Available: http://dx.doi.org/10.1109/HICSS.2013.599

[37] A. Wlodyka, R. Mlynarski, G. Ilczuk, E. Pilat, and W. Kargul, "Visualization of decision rules - from the cardiologist's point of view," in *Proc. of the Int'l Conf. on Computers in Cardiology*, 2008, pp. 645–648. [Online]. Available: http://dx.doi.org/10.1109/CIC.2008.4749124

[38] P. C. Wong, P. Whitney, and J. Thomas, "Visualizing association rules for text mining," in *Proc. of the IEEE Symp. on Information Visualization*, ser. InfoVis '99, 5th, 1999, pp. 120–123, 152.

[39] J. Blanchard, F. Guillet, and H. Briand, "A user-driven and quality-oriented visualization for mining association rules," in *Proc. of the IEEE Int'l. Conf. on Data Mining*, ser. ICDM '03, 3rd, 2003, pp. 493–496. [Online]. Available: http://dx.doi.org/10.1109/ICDM.2003.1250960

[40] M. Hahsler and S. Chelluboina, "Visualizing association rules in hierarchical groups," in *Proc. of the Symp. on the Interface: Statistical, Machine Learning, and Visualization Algorithms*, ser. Interface '11, 42nd. The Interface Foundation of North America, June 2011.

[41] S. Afzal, R. Maciejewski, and D. Ebert, "Visual analytics decision support environment for epidemic modeling and response evaluation," in *Proc. of the IEEE Conf. on Visual Analytics Science and Technology*, ser. VAST '11, 2011, pp. 191–200. [Online]. Available: http://dx.doi.org/10.1109/VAST.2011.6102457

[42] M. Migut and M. Worring, "Visual exploration of classification models for risk assessment," in *Proc. of the IEEE Symp. on Visual Analytics Science and Technology*, ser. VAST '10, 2010, pp. 11–18. [Online]. Available: http://dx.doi.org/10.1109/VAST.2010.5652398

[43] M. Friendly, "Mosaic displays for multi-way contingency tables," *Journal of the American Statistical Association*, vol. 89, no. 425, pp. 190–200, 1994. [Online]. Available: http://dx.doi.org/10.1080/01621459.1994.10476460

[44] D. C. Oppen, "Prettyprinting," *ACM Transactions on Programming Languages and Systems*, vol. 2, no. 4, pp. 465–483, Oct. 1980. [Online]. Available: http://doi.acm.org/10.1145/357114.357115

[45] D. E. Knuth, "Literate programming," *The Computer Journal*, vol. 27, no. 2, pp. 97–111, 1984. [Online]. Available: http://dx.doi.org/10.1093/comjnl/27.2.97

[46] ——, *Literate Programming*, ser. CSLI lectures notes number. Center for the Study of Language and Information, Leland Stanford Junior University, 1992.

[47] R. M. Baecker and A. Marcus, *Human factors and typography for more readable programs*. New York, NY, USA: ACM, 1989.

[48] ——, "Printing and publishing c programs," in *Software Visualization–Programming as a Multimedia Experience*. MIT Press, 1998, pp. 45–61.

[49] M. A. Jackson, *Principles of Program Design*. Orlando, FL, USA: Academic Press, Inc., 1975.

[50] H. Goldstine and J. von Neumann, *Planning and Coding of Problems for an Electronic Computing Instrument*, ser. Report on the mathematical and logical aspects of an electronic computing instrument. Institute for Advanced Study, Princeton, N. J., 1947.

[51] I. Nassi and B. Shneiderman, "Flowchart techniques for structured programming," *ACM SIGPLAN Notices*, vol. 8, no. 8, pp. 12–26, Aug. 1973. [Online]. Available: http://dx.doi.org/10.1145/953349.953350

[52] P. Irani and C. Ware, "Diagrams based on structural object perception," in *Proc. of the ACM Working Conf. on Advanced Visual Interfaces*, ser. AVI '00. New York, NY, USA: ACM, 2000, pp. 61–67. [Online]. Available: http://dx.doi.org/10.1145/345513.345254

[53] ——, "The effect of a perceptual syntax on the learnability of novel concepts," in *Proc. of the Int'l. Conf. on Information Visualisation*, ser. IV '04, 8th, 2004, pp. 308–314.

[54] M. Lanza, R. Marinescu, and S. Ducasse, *Object-Oriented Metrics in Practice*. Springer, 2005.

[55] R. J. Walker, G. C. Murphy, B. Freeman-Benson, D. Wright, D. Swanson, and J. Isaak, "Visualizing dynamic software system information through high-level models," *ACM SIGPLAN Notices*, vol. 33, no. 10, pp. 271–283, Oct. 1998. [Online]. Available: http://dx.doi.org/10.1145/286942.286966

[56] T. Systä, K. Koskimies, and H. Müller, "Shimba–an environment for reverse engineering java software systems," *Software: Practice and Experience*, vol. 31, no. 4, pp. 371–394, 2001. [Online]. Available: http://dx.doi.org/10.1002/spe.386

[57] K. Mehner, "Javis: A uml-based visualization and debugging environment for concurrent java programs," in *Software Visualization*, ser. Lecture Notes in Computer Science, S. Diehl, Ed. Springer Berlin Heidelberg, 2002, vol. 2269, pp. 163–175. [Online]. Available: http://dx.doi.org/10.1007/3-540-45875-1_13

[58] J. Grundy and J. Hosking, "Softarch: Tool support for integrated software architecture development," *Int'l. Journal of Software Engineering and Knowledge Engineering*, vol. 13, no. 02, pp. 125–151, 2003. [Online]. Available: http://dx.doi.org/10.1142/S0218194003001238

[59] W. de Pauw, R. Helm, D. Kimelman, and J. Vlissides, "Visualizing the behavior of object-oriented systems," in *Proc. of the ACM Annual Conf. on Object-oriented Programming Systems, Languages, and Applications*, ser. OOPSLA '93, 8th. New York, NY, USA: ACM, 1993, pp. 326–337. [Online]. Available: http://dx.doi.org/10.1145/165854.165919

[60] S. Ducasse, M. Lanza, and R. Bertuli, "High-level polymetric views of condensed run-time information," in *Proc. of the European Conf. on Software Maintenance and Reengineering*, ser. CSMR '04, 8th, 2004, pp. 309–318.

[61] A. Zaidman and S. Demeyer, "Managing trace data volume through a heuristical clustering process based on event execution frequency," in *Proc. of the European Conf. on Software Maintenance and Reengineering*, ser. CSMR '04, 8th, 2004, pp. 329–338. [Online]. Available: http://dx.doi.org/10.1109/CSMR.2004.1281435

[62] A. Kuhn and O. Greevy, "Exploiting the analogy between traces and signal processing," in *Proc. of the IEEE Int'l. Conf. on Software Maintenance*, ser. ICSM '06, 22nd, 2006, pp. 320–329.

[63] B. Cornelissen, D. Holten, A. Zaidman, L. Moonen, J. J. van Wijk, and A. van Deursen, "Understanding execution traces using massive sequence and circular bundle views," in *Proc. of the IEEE Int'l. Conf. on Program Comprehension*, ser. ICPC '07, 15th, 2007, pp. 49–58. [Online]. Available: http://dx.doi.org/10.1109/ICPC.2007.39

[64] D. Holten, B. Cornelissen, and J. J. van Wijk, "Trace visualization using hierarchical edge bundles and massive sequence views," in *Proc. of the IEEE Int'l. Workshop on Visualizing Software for Understanding and Analysis*, ser. VISSOFT '07, 4th, 2007, pp. 47–54. [Online]. Available: http://dx.doi.org/10.1109/VISSOF.2007.4290699

[65] B. Cornelissen, A. Zaidman, D. Holten, L. Moonen, A. van Deursen, and J. J. van Wijk, "Execution trace analysis through massive sequence and circular bundle views," *Journal of Systems and Software*, vol. 81, no. 12, pp. 2252–2268, 2008, ¡ce:title¿Best papers from the 2007 Australian Software Engineering Conf. (ASWEC 2007), Melbourne, Australia, April 10-13, 2007¡/ce:title¿ ¡xocs:full-name¿Australian Software Engineering Conf. 2007¡/xocs:full-name¿. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2008.02.068

[66] S. G. Eick, J. L. Steffen, and J. Sumner, E. E., "Seesoft-a tool for visualizing line oriented software statistics," *IEEE Transactions on Software Engineering*, vol. 18, no. 11, pp. 957–968, 1992.

[67] M. J. Baker and S. G. Eick, "Space-filling software visualization," *Journal of Visual Languages and Computing*, vol. 6, no. 2, pp. 119–133, 1995. [Online]. Available: http://dx.doi.org/10.1006/jvlc.1995.1007

[68] (2013) Graphical interfacs for cvs. [Online]. Available: http://cvsgui.sourceforge.net/

[69] (2013) Graphical interface for subversion. [Online]. Available: http://tortoisesvn.net/

[70] (2013) Graphical interfaces for git. [Online]. Available: http://git-scm.com/downloads/guis

[71] L. Voinea, A. C. Telea, and J. J. van Wijk, "Cvsscan: Visualization of code evolution," in *Proc. of the ACM Symp. on Software Visualization*, ser. SoftVis '05. New York, NY, USA: ACM, 2005, pp. 47–56. [Online]. Available: http://dx.doi.org/10.1145/1056018.1056025

[72] Z. Liu, J. Stasko, and T. Sullivan, "Selltrend: Inter-attribute visual analysis of temporal transaction data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1025–1032, 2009. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2009.180

[73] B. Alsallakh, E. Gröller, S. Miksch, and M. Suntinger, "Contingency wheel: Visual analysis of large contingency tables," in *Proc. of the Int'l. Workshop on Visual Analytics*, ser. EuroVA '11. Eurographics, 2011, pp. 53–56. [Online]. Available: http://dx.doi.org/10.2312/PE/EuroVAST/EuroVA11/053-056

[74] C. Turkay, A. Lundervold, A. J. Lundervold, and H. Hauser, "Representative factor generation for the interactive visual analysis of high-dimensional data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2621–2630, dec. 2012. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2012.256

[75] S. J. Fernstad and J. Johansson, "A task based performance evaluation of visualization approaches for categorical data analysis," in *Proc. of the Int'l. Conf. on Information Visualisation*, ser. IV '11, 15th. Washington, DC, USA: IEEE Computer Society Press, 2011, pp. 80–89. [Online]. Available: http://dx.doi.org/10.1109/IV.2011.92

[76] E. R. Tufte, *Envisioning Information*, 4th ed. Graphics Press, 1990.

[77] J. S. Yi, Y. A. Kang, J. Stasko, and J. Jacko, "Toward a deeper understanding of the role of interaction in information visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1224–1231, Nov. 2007. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2007.70515

[78] J. M. Heer and B. Shneiderman, "Interactive dynamics for visual analysis," *Queue*, vol. 10, no. 2, pp. 30:30–30:55, Feb. 2012. [Online]. Available: http://dx.doi.org/10.1145/2133416.2146416

[79] A. Kerren and F. Schreiber, "Toward the role of interaction in visual analytics," in *Proc. of the Winter Simulation Conf.*, ser. WSC '12, 2012, pp. 1–13. [Online]. Available: http://dx.doi.org/10.1109/WSC.2012.6465208

[80] J. W. Tukey, *Exploratory Data Analysis*. Addison-Wesley, 1977.

[81] J. Bertin, *La graphique et le traitement graphique de l'information*. Flammarion, 1977.

[82] F. van Ham, "Using multilevel call matrices in large software projects," in *Proc. of the IEEE Symp. on Information Visualization*, ser. InfoVis '03, 9th, 2003, pp. 227–232. [Online]. Available: http://dx.doi.org/10.1109/INFVIS.2003.1249030

[83] J. Abello and F. van Ham, "Matrix zoom: A visual interface to semi-external graphs," in *Proc. of the IEEE Symp. on Information Visualization*, ser. InfoVis '04, 10th, 2004, pp. 183–190.

[84] N. Henry, J.-D. Fekete, and M. McGuffin, "Nodetrix: a hybrid visualization of social networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1302–1309, 2007. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2007.70582

[85] N. Henry and J.-D. Fekete, "Matlink: Enhanced matrix visualization for analyzing social networks," in *Human-Computer Interaction*, ser. INTERACT '07, Lecture Notes in Computer Science, C. A. Baranauskas, P. Palanque, J. Abascal, and S. Barbosa, Eds. Springer Berlin Heidelberg, 2007, vol. 4663, pp. 288–302. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-74800-7_24

[86] R. Chang, M. Ghoniem, R. Kosara, W. Ribarsky, J. Yang, E. Suma, C. Ziemkiewicz, D. Kern, and A. Sudjianto, "Wirevis: Visualization of categorical, time-varying data from financial transactions," in *Proc. of the IEEE Symp. on Visual Analytics Science and Technology*, ser. VAST '07, 2007, pp. 155–162. [Online]. Available: http://dx.doi.org/10.1109/VAST.2007.4389009

[87] K. Dinkla, M. A. Westenberg, and J. J. van Wijk, "Compressed adjacency matrices: Untangling gene regulatory networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2457–2466, 2012. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2012.208

[88] M. Ghoniem, J.-D. Fekete, and P. Castagliola, "A comparison of the readability of graphs using node-link and matrix-based representations," in *Proc. of the IEEE Symp. on Information Visualization*, ser. InfoVis '04, 10th. Washington, DC, USA: IEEE Computer Society Press, 2004, pp. 17–24. [Online]. Available: http://dx.doi.org/10.1109/INFOVIS.2004.1

[89] ——, "On the readability of graphs using node-link and matrix-based representations: A controlled experiment and statistical analysis," *Information Visualization*, vol. 4, no. 2, pp. 114–135, Jul. 2005. [Online]. Available: http://dx.doi.org/10.1057/palgrave.ivs.9500092

[90] R. Rao and S. K. Card, "The table lens: Merging graphical and symbolic representations in an interactive focus + context visualization for tabular information," in *Proc. of the ACM SIGCHI Conf. on Human Factors in Computing Systems*, ser. CHI '94. New York, NY, USA: ACM, 1994, pp. 318–322. [Online]. Available: http://dx.doi.org/10.1145/191666.191776

[91] P. Pirolli and R. Rao, "Table lens as a tool for making sense of data," in *Proc. of the ACM Workshop on Advanced Visual Interfaces*, ser. AVI '96. New York, NY, USA: ACM, 1996, pp. 67–80. [Online]. Available: http://dx.doi.org/10.1145/948449.948460

[92] M. Sarkar and M. H. Brown, "Graphical fisheye views of graphs," in *Proc. of the ACM SIGCHI Conf. on Human Factors in Computing Systems*, ser. CHI '92. New York, NY, USA: ACM, 1992, pp. 83–91. [Online]. Available: http://dx.doi.org/10.1145/142750.142763

[93] T. Tenev and R. Rao, "Managing multiple focal levels in table lens," in *Proc. of the IEEE Symp. on Information Visualization*, ser. InfoVis '97, 3rd, 1997, pp. 59–63. [Online]. Available: http://dx.doi.org/10.1109/INFVIS.1997.636787

[94] A. C. Telea, "Combining extended table lens and treemap techniques for visualizing tabular data," in *Proc. of the Joint Eurographics / IEEE VGTC Conf. on Visualization*, ser. EuroVis '06, 8th. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2006, pp. 51–58. [Online]. Available: http://dx.doi.org/10.2312/VisSym/EuroVis06/051-058

[95] S. K. Feiner and C. Beshers, "Worlds within worlds: Metaphors for exploring n-dimensional virtual worlds," in *Proc. of the ACM Annual SIGGRAPH Symp. on User Interface Software and Technology*, ser. UIST '90, 3rd. New York, NY, USA: ACM, 1990, pp. 76–83. [Online]. Available: http://dx.doi.org/10.1145/97924.97933

[96] A. Inselberg, "Multidimensional detective," in *Proc. of the IEEE Symp. on Information Visualization*, ser. InfoVis '97, 3rd, 1997, pp. 100–107. [Online]. Available: http://dx.doi.org/10.1109/INFVIS.1997.636793

[97] J. Yang, W. Peng, M. O. Ward, and E. A. Rundensteiner, "Interactive hierarchical dimension ordering, spacing and filtering for exploration of high dimensional datasets," in *Proc. of the IEEE Symp. on Information Visualization*, ser. InfoVis '03, 9th. Washington, DC, USA: IEEE Computer Society Press, 2003, pp. 105–112. [Online]. Available: http://dx.doi.org/10.1109/INFVIS.2003.1249015

[98] H. Hauser, F. Ledermann, and H. Doleisch, "Angular brushing of extended parallel coordinates," in *Proc. of the IEEE Symp. on Information Visualization*, ser. InfoVis '02, 8th, 2002, pp. 127–130. [Online]. Available: http://dx.doi.org/10.1109/INFVIS.2002.1173157

[99] D. A. Keim and H.-P. Kriegel, "Visdb: database exploration using multidimensional visualization," *IEEE Computer Graphics and Applications*, vol. 14, no. 5, pp. 40–49, 1994. [Online]. Available: http://dx.doi.org/10.1109/38.310723

[100] M. Friendly, "Visualizing categorical data: Data, stories, and pictures," in *Proc. of the SAS User Group Conf.*, 2000.

[101] ——, "Extending mosaic displays: Marginal, conditional, and partial views of categorical data," *Journal of Computational and Graphical Statistics*, vol. 8, pp. 373–395, 1999. [Online]. Available: http://dx.doi.org/10.1080/10618600.1999.10474820

[102] E. Kolatch and B. Weinstein. (2001) Cattrees: Dynamic visualization of categorical data using treemaps. [Online]. Available: http://www.cs.umd.edu/class/spring2001/cmsc838b/Project/Kolatch_Weinstein/index.html

[103] B. Johnson and B. Shneiderman, "Tree-maps: a space-filling approach to the visualization of hierarchical information structures," in *Proc. of the IEEE Conf. on Visualization*, ser. VIS '91, 2nd, 1991, pp. 284–291. [Online]. Available: http://dx.doi.org/10.1109/VISUAL.1991.175815

[104] R. Kosara, F. Bendix, and H. Hauser, "Parallel sets: interactive exploration and visual analysis of categorical data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 4, pp. 558–568, 2006. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2006.76

[105] T. Mihalisin, J. Timlin, and J. Schwegler, "Visualization and analysis of multi-variate data: A technique for all fields," in *Proc. of the IEEE Conf. on Visualization*, ser. VIS '91, 2nd. Los Alamitos, CA, USA: IEEE Computer Society Press, 1991, pp. 171–178. [Online]. Available: http://dx.doi.org/10.1109/VISUAL.1991.175796

[106] C. Stolte, D. Tang, and P. Hanrahan, "Polaris: a system for query, analysis, and visualization of multidimensional relational databases," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 1, pp. 52–65, 2002. [Online]. Available: http://dx.doi.org/10.1109/2945.981851

[107] J. D. Mackinlay, P. Hanrahan, and C. Stolte, "Show me: Automatic presentation for visual analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1137–1144, Nov. 2007. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2007.70594

[108] J. W. Emerson, W. A. Green, B. Schloerke, J. Crowley, D. Cook, H. Hofmann, and H. Wickham, "The generalized pairs plot," *Journal of Computational and Graphical Statistics*, vol. 22, no. 1, pp. 79–91, 2013. [Online]. Available: http://dx.doi.org/10.1080/10618600.2012.694762

[109] P. Joia, F. V. Paulovich, D. Coimbra, J. A. Cuminato, and L. G. Nonato, "Local affine multidimensional projection," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2563–2571, 2011. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2011.220

[110] F. V. Paulovich, L. G. Nonato, R. Minghim, and H. Levkowitz, "Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 3, pp. 564–575, 2008. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2007.70443

[111] W. Cui, Y. Wu, S. Liu, F. Wei, M. Zhou, and H. Qu, "Context preserving dynamic word cloud visualization," in *Proc. of the IEEE Pacific Visualization Symp.*, ser. PacificVis '10, 2010, pp. 121–128. [Online]. Available: http://dx.doi.org/10.1109/PACIFICVIS.2010.5429600

[112] C. Faloutsos and K.-I. Lin, "Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets," *ACM SIGMOD Record*, vol. 24, no. 2, pp. 163–174, may 1995. [Online]. Available: http://dx.doi.org/10.1145/568271.223812

[113] F. V. Paulovich, C. T. Silva, and L. G. Nonato, "Two-phase mapping for projecting massive data sets," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1281–1290, 2010. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2010.207

[114] J. Poco, D. M. Eler, F. V. Paulovich, and R. Minghim, "Employing 2d projections for fast visual exploration of large fiber tracking data," *Computer Graphics Forum*, vol. 31, no. 3pt2, pp. 1075–1084, 2012. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2012.03100.x

[115] A. Cauchy, "Sur l'equation a l'aide de laquelle on determine les inegalites seculaires des mouvements des planetes," *Oeuvres Completes (2eme serie)*, vol. 4, pp. 174–195, 1829.

[116] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *Philosophical Magazine Series 6*, vol. 2, no. 11, pp. 559–572, 1901. [Online]. Available: http://dx.doi.org/10.1080/14786440109462720

[117] H. Hotteling, "Analysis of a complex of statistical variables into principal components," *Journal of Educational Psychology*, vol. 24, no. 6, pp. 417–441, 1933. [Online]. Available: http://dx.doi.org/10.1037/h0071325

[118] H. O. Hirschfeld, "A connection between correlation and contingency," *Mathematical Proc. of the Cambridge Philosophical Society*, vol. 31, pp. 520–524, 10 1935. [Online]. Available: http://dx.doi.org/10.1017/S0305004100013517

[119] J. P. Benzecri and L. Bellier, *L'analyse des donnees*, 2nd ed. Paris, France: Dunod, 1976.

[120] K. R. Gabriel, "The biplot graphic display of matrices with application to principal component analysis," *Biometrika*, vol. 58, no. 3, pp. 453–467, 1971. [Online]. Available: http://biomet.oxfordjournals.org/content/58/3/453

[121] M. J. Greenacre, *Biplots in Practice*. Fundación BBVA, 2010.

[122] J. C. Gower, S. G. Lubbe, and N. J. L. Roux, *Understanding Biplots*. Wiley, 2011.

[123] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000. [Online]. Available: http://dx.doi.org/10.1126/science.290.5500.2323

[124] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000. [Online]. Available: http://dx.doi.org/10.1126/science.290.5500.2319

[125] V. de Silva and J. B. Tenenbaum, "Global versus local methods in nonlinear dimensionality reduction," in *NIPS*, S. Becker, S. Thrun, and K. Obermayer, Eds. MIT Press, 2002, pp. 705–712.

[126] S. Lee and S. Choi, "Landmark {MDS} ensemble," *Pattern Recognition*, vol. 42, no. 9, pp. 2045–2053, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2008.11.039

[127] U. Brandes and C. Pich, "Eigensolver methods for progressive multidimensional scaling of large data," in *Graph Drawing*, ser. Lecture Notes in Computer Science, M. Kaufmann and D. Wagner, Eds. Springer Berlin Heidelberg, 2007, vol. 4372, pp. 42–53. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-70904-6_6

[128] E. R. Gansner, Y. Koren, and S. North, "Graph drawing by stress majorization," in *Graph Drawing*, ser. Lecture Notes in Computer Science, J. Pach, Ed. Springer Berlin Heidelberg, 2005, vol. 3383, pp. 239–250. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-31843-9_25

[129] J. W. Sammon, "A nonlinear mapping for data structure analysis," *IEEE Transactions on Computers*, vol. C-18, no. 5, pp. 401–409, 1969.

[130] M. M. Bronstein, A. M. Bronstein, R. Kimmel, and I. Yavneh, "Multigrid multidimensional scaling," *Numerical Linear Algebra with Applications*, vol. 13, pp. 149–171, 2006. [Online]. Available: http://dx.doi.org/10.1002/nla.475

[131] E. Pekalska, D. de Ridder, R. P. W. Duin, and M. A. Kraaijveld, "A new method of generalizing sammon mapping with application to algorithm speed-up," in *Proc. of the Annual Conf. of the Advanced School for Computing and Imaging*, ser. ASCI '99, 5th, vol. 99, 1999, pp. 221–228.

[132] P. A. Eades, "A heuristic for graph drawing," in *Congressus Numerantium*, vol. 42, 1984, pp. 149–160.

[133] M. Chalmers, "A linear iteration time layout algorithm for visualising high-dimensional data," in *Proc. of the IEEE Conf. on Visualization*, ser. VIS '96, 7th, 1996, pp. 127–131. [Online]. Available: http://dx.doi.org/10.1109/VISUAL.1996.567787

[134] Y. Frishman and A. Tal, "Multi-level graph layout on the gpu," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1310–1319, 2007. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2007.70580

[135] S. Ingram, T. Munzner, and M. Olano, "Glimmer: Multilevel mds on the gpu," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 2, pp. 249–261, 2009.

[136] F. Jourdan and G. Melançon, "Multiscale hybrid mds," in *Proc. of the Int'l. Conf. on Information Visualisation*, ser. IV '04, 8th, 2004, pp. 388–393.

[137] E. Tejada, R. Minghim, and L. G. Nonato, "On improved projection techniques to support visual exploration of multi-dimensional data sets," *Information Visualization*, vol. 2, no. 4, pp. 218–231, 2003. [Online]. Available: http://dx.doi.org/10.1057/palgrave.ivs.9500054

[138] A. Frank and A. Asuncion. (2010) UCI machine learning repository. [Online]. Available: http://archive.ics.uci.edu/ml

[139] T. Baudel, "Browsing through an information visualization design space," in *Extended Abstracts of the ACM SIGCHI Conf. on Human Factors in Computing Systems*, ser. CHI EA '04. New York, NY, USA: ACM, 2004, pp. 765–766. [Online]. Available: http://dx.doi.org/10.1145/985921.985925

[140] ——, "Visualizing business rule management system artifacts." IBM, IBM Center for Advanced Studies, Paris, France, Tech. Rep., 2009.

[141] B. Shneiderman. (2009) Treemaps for space-constrained visualization of hierarchies. [Online]. Available: http://www.cs.umd.edu/hcil/treemap-history/

[142] (1998) Map of the market. SmartMoney.com. [Online]. Available: http://smartmoney.com/marketmap

[143] (2010) Superpower: Visualising the internet. BBC. [Online]. Available: http://news.bbc.co.uk/2/hi/technology/8562801.stm

[144] (2007) Health of the car, van, suv, and truck markets. The New York Times. [Online]. Available: http://www.nytimes.com/imagepages/2007/02/25/business/20070225_CHRYSLER_GRAPHIC.html

[145] M. Bruls, K. Huizing, and J. J. van Wijk, "Squarified treemaps," in *Data Visualization*, ser. Eurographics, W. Leeuw and R. Liere, Eds. Springer Vienna, 2000, pp. 33–42. [Online]. Available: http://dx.doi.org/10.1007/978-3-7091-6783-0_4

[146] W. Buxton, "Chunking and phrasing and the design of human-computer dialogues," in *Proc. of the IFIP World Computer Congress*. North Holland Publishers, 1986, pp. 475–480.

[147] N. Kong, J. M. Heer, and M. Agrawala, "Perceptual guidelines for creating rectangular treemaps," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 990–998, 2010. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2010.186

[148] B. Otjacques, M. Cornil, M. Noirhomme, and F. Feltz, "Cgd – a new algorithm to optimize space occupation in ellimaps," in *Human-Computer Interaction*, ser. INTERACT '09, Lecture Notes in Computer Science, T. Gross, J. Gulliksen, P. Kotzé, L. Oestreicher, P. Palanque, R. Prates, and M. Winckler, Eds., vol. 5727. Springer Berlin Heidelberg, 2009, pp. 805–818. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-03658-3_84

[149] M. Balzer, O. Deussen, and C. Lewerentz, "Voronoi treemaps for the visualization of software metrics," in *Proc. of the ACM Symp. on Software Visualization*, ser. SoftVis '05. New York, NY, USA: ACM, 2005, pp. 165–172. [Online]. Available: http://dx.doi.org/10.1145/1056018.1056041

[150] J. M. Heer and M. Agrawala, "Software design patterns for information visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 853–860, 2006. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2006.178

[151] T. Baudel, "Visualisations compactes: Une approche déclarative pour la visualisation d'information," in *Proc. of the French-speaking Conf. on Human-computer Interaction (Conférence Francophone Sur L'Interaction Homme-Machine)*, ser. IHM '02, 14th. New York, NY, USA: ACM, 2002, pp. 161–168. [Online]. Available: http://dx.doi.org/10.1145/777005.777027

[152] M. Friendly, "A brief history of the mosaic display," *Journal of Computational and Graphical Statistics*, vol. 11, pp. 89–107, mar 2002. [Online]. Available: http://dx.doi.org/10.1198/106186002317375631

[153] D. A. Keim, M. C. Hao, and U. Dayal, "Hierarchical pixel bar charts," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 3, pp. 255–269, 2002. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2002.1021578

[154] M. Zizi and M. Beaudouin-Lafon, "Accessing hyperdocuments through interactive dynamic maps," in *Proc. of the ACM European Conf. on Hypermedia Technology*, ser. ECHT '94. New York, NY, USA: ACM, 1994, pp. 126–135. [Online]. Available: http://dx.doi.org/10.1145/192757.192786

[155] B. B. Bederson, B. Shneiderman, and M. Wattenberg, "Ordered and quantum treemaps: Making effective use of 2d space to display hierarchies," *ACM Transactions on Graphics*, vol. 21, no. 4, pp. 833–854, Oct. 2002. [Online]. Available: http://dx.doi.org/10.1145/571647.571649

[156] J. Wood and J. Dykes, "Spatially ordered treemaps," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1348–1355, 2008. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2008.165

[157] M. de Berg, K. Onak, and A. Sidiropoulos, "Fat polygonal partitions with applications to visualization and embeddings," *Computing Research Repository*, vol. abs/1009.1866, 2010. [Online]. Available: http://arxiv.org/abs/1009.1866

[158] K. Onak and A. Sidiropoulos, "Circular partitions with applications to visualization and embeddings," in *Proc. of the ACM Annual Symp. on Computational Geometry*, ser. SCG '08, 24th. New York, NY, USA: ACM, 2008, pp. 28–37. [Online]. Available: http://dx.doi.org/10.1145/1377676.1377683

[159] H.-J. Schulz, S. Hadlak, and H. Schumann, "The design space of implicit hierarchy visualization: A survey," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 4, pp. 393–411, 2011. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2010.79

[160] R. Vliegen, J. J. van Wijk, and E.-J. van der Linden, "Visualizing business data with generalized treemaps," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 789–796, 2006. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2006.200

[161] E. H. Chi and J. Riedl, "An operator interaction framework for visualization systems," in *Proc. of the IEEE Symp. on Information Visualization*, ser. InfoVis '98, 4th, 1998, pp. 63–70. [Online]. Available: http://dx.doi.org/10.1109/INFVIS.1998.729560

[162] J. Bertin, *Semiology of graphics*. University of Wisconsin Press, 1983.

[163] L. Wilkinson, *The Grammar of Graphics (Statistics and Computing)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[164] H. Wickham and H. Hofmann, "Product plots," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2223–2230, 2011. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2011.227

[165] A. Slingsby, J. Dykes, and J. Wood, "Configuring hierarchical layouts to address research questions," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 977–984, 2009. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2009.128

[166] H.-J. Schulz, Z. Akbar, and F. Maurer, "A generative layout approach for rooted tree drawings," in *Proc. of the IEEE Pacific Visualization Symp.*, ser. PacificVis '13. IEEE, Feb. 2013, pp. 225–232. [Online]. Available: http://dx.doi.org/10.1109/PacificVis.2013.6596149

[167] J. Larus, "Efficient program tracing," *Computer*, vol. 26, no. 5, pp. 52–61, 1993.

[168] R. Smith and B. Korel, "Slicing event traces of large software systems," *arXiv preprint cs/0101005*, 2001. [Online]. Available: http://arxiv.org/abs/cs/0101005

[169] A. Zaidman, "Scalability solutions for program comprehension through dynamic analysis," in *Proc. of the European Conf. on Software Maintenance and Reengineering*, ser. CSMR '06, 10th, 2006, pp. 4 pp.–330. [Online]. Available: http://dx.doi.org/10.1109/CSMR.2006.46

[170] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. J. van Wijk, J.-D. Fekete, and D. Fellner, "Visual analysis of large graphs: State-of-the-art and future research challenges," *Computer Graphics Forum*, vol. 30, no. 6, pp. 1719–1749, 2011. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2011.01898.x

[171] A. Chniti, P. Albert, and J. Charlet, "Mdrontology: An ontology for managing ontology changes impacts on business rules," in *CEUR Workshop proceedings. Joint Workshop on Knowledge Evolution and Ontology Dynamics. In conjunction with Int'l. Semantic Web Conf. (ISWC).*, 2012.

[172] D. Reniers, L. Voinea, and A. C. Telea, "Visual exploration of program structure, dependencies and metrics with solidsx," in *Proc. of the IEEE Int'l. Workshop on Visualizing Software for Understanding and Analysis*, ser. VISSOFT '11, 6th, 2011, pp. 1–4. [Online]. Available: http://dx.doi.org/10.1109/VISSOF.2011.6069461

[173] B. Broeksema and A. Telea, "Visual support for porting large code bases," in *Proc. of the IEEE Int'l. Workshop on Visualizing Software for Understanding and Analysis*, ser. VISSOFT '11, 6th, 2011, pp. 1–8. [Online]. Available: http://dx.doi.org/10.1109/VISSOF.2011.6069450

[174] T. Baudel and F. van Ham, "Rule correlation to rules input attributes according to disparate distribution analysis," US Patent US20 130 103 636 A1, 04 25, 2013.

[175] ——, "Contextual feedback of rules proximity based upon co-occurence history in a collaborative rule editing system," US Patent US20 130 073 512 A1, 03 21, 2013.

[176] H. Abdi and D. Valentin, *Encyclopedia of Measurement and Statistics*. Thousand Oaks (CA), 2007, ch. Multiple Correspondence Analysis, pp. 651–657.

[177] M. Tenenhaus and F. W. Young, "An analysis and synthesis of multiple correspondence analysis, optimal scaling, dual scaling, homogeneity analysis and other methods for quantifying categorical multivariate data," *Psychometrika*, vol. 50, no. 1, pp. 91–119, 1985. [Online]. Available: http://dx.doi.org/10.1007/BF02294151

[178] P. Bourdieu, *La distinction: Critique sociale du jugement*, ser. Collection "Le Sens commun". Éditions de Minuit, jan 1979.

[179] H. Abdi and L. J. Williams, *Encyclopedia of Research Design*. Thousand Oaks, 2010, ch. Correspondence Analysis, pp. 267–278.

[180] H. Abdi, *Encyclopedia of Measurement and Statistics*. Thousand Oaks, 2010, ch. Singular Value Decomposition and Generalized Singular Value Decomposition, pp. 907–912.

[181] S. Johansson, M. Jern, and J. Johansson, "Interactive quantification of categorical variables in mixed data sets," in *Proc. of the Int'l. Conf. on Information Visualisation*, ser. IV '08, 12th, 2008, pp. 3–10. [Online]. Available: http://dx.doi.org/10.1109/IV.2008.33

[182] C. A. Brewer and M. Harrower. (2011) Color Brewer 2.0. http://colorbrewer2.org.

[183] A. C. Telea and J. J. van Wijk, "Visualization of Generalized Voronoi Diagrams," in *Data Visualization*, ser. Eurographics, D. Ebert, J. Favre, and R. Peikert, Eds. Springer Vienna, 2001, pp. 165–174. [Online]. Available: http://dx.doi.org/10.1007/978-3-7091-6215-6_18

[184] N. H. Riche, B. Lee, and C. Plaisant, "Understanding interactive legends: a comparative evaluation with standard widgets," *Computer Graphics Forum*, vol. 29, no. 3, pp. 1193–1202, 2010. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2009.01678.x

[185] S. Oeltze, H. Doleisch, H. Hauser, P. Muigg, and B. Preim, "Interactive visual analysis of perfusion data," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 13, no. 6, pp. 1392–1399, 2007. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2007.70569

[186] B. Shneiderman, "The eyes have it: a task by data type taxonomy for information visualizations," in *Proc. of the IEEE Symp. on Visual Languages*, 1996, pp. 336–343. [Online]. Available: http://dx.doi.org/10.1109/VL.1996.545307

[187] D. A. Keim, F. Mansmann, J. Schneidewind, and H. Ziegler, "Challenges in visual data analysis," in *Proc. of the Int'l. Conf. on Information Visualisation*, ser. IV '06, 10th, 2006, pp. 9–16. [Online]. Available: http://dx.doi.org/10.1109/IV.2006.31

[188] G. H. Golub and C. F. van Loan, *Matrix computations (3rd ed.)*. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[189] E. Bertini and G. Santucci, "By chance is not enough: preserving relative density through nonuniform sampling," in *Proc. of the Int'l. Conf. on Information Visualisation*, ser. IV '04, 8th, 2004, pp. 622–629.

[190] R. van Liere and W. de Leeuw, "Graphsplatting: visualizing graphs as continuous fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 2, pp. 206–212, 2003.

[191] (2013) Apache solr. The Apache Software Foundation. [Online]. Available: lucene.apache.org/solr

[192] B. Broeksema, A. C. Telea, and T. Baudel, "Visual analysis of multi-dimensional categorical data sets," *Computer Graphics Forum*, vol. 32, no. 8, pp. 158–169, 2013. [Online]. Available: http://dx.doi.org/10.1111/cgf.12194

[193] D. L. Post and F. A. Greene, "Color-name boundaries for equally bright stimuli on a crt: Phase i," *SID Digest*, vol. 86, pp. 70–73, 1986.

[194] C. Healey, "Choosing effective colours for data visualization," in *Proc. of the IEEE Conf. on Visualization*, ser. VIS '96, 7th, 1996, pp. 263–270. [Online]. Available: http://dx.doi.org/10.1109/VISUAL.1996.568118

[195] M. A. Harrower and C. A. Brewer, "Colorbrewer.org: An online tool for selecting color schemes for maps," *Cartographic Journal*, vol. 40, pp. 27–37, 2003.

[196] C. Ware and J. C. Beatty, "Using color dimensions to display data dimensions," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 30, no. 2, pp. 127–142, 1988. [Online]. Available: http://dx.doi.org/10.1177/001872088803000201

[197] J.-D. Fekete and C. Plaisant, "Excentric labeling: Dynamic neighborhood labeling for data visualization," in *Proc. of the ACM SIGCHI Conf. on Human Factors in Computing Systems*, ser. CHI '99. New York, NY, USA: ACM, 1999, pp. 512–519. [Online]. Available: http://dx.doi.org/10.1145/302979.303148

[198] J. Christensen, J. Marks, and S. Shieber, "An empirical study of algorithms for point-feature label placement," *ACM Transactions on Graphics*, vol. 14, no. 3, pp. 203–232, jul 1995. [Online]. Available: http://dx.doi.org/10.1145/212332.212334

[199] K. Been, E. Daiches, and C. Yap, "Dynamic map labeling," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 773–780, 2006. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2006.136

[200] M. Fink, J.-H. Haunert, A. Schulz, J. Spoerhase, and A. Wolff, "Algorithms for labeling focus regions," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2583–2592, 2012. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2012.193

[201] B. Shneiderman and H. Kang, "Direct annotation: a drag-and-drop strategy for labeling photos," in *Proc. of the Int'l. Conf. on Information Visualisation*, ser. IV '00, 4th, 2000, pp. 88–95. [Online]. Available: http://dx.doi.org/10.1109/IV.2000.859742

[202] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, Mar. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944968

[203] C. Schmid, *Statistical Graphics: Design Principles and Practices*, ser. A Wiley-Interscience publication. John Wiley & Sons Australia, Limited, 1983.

[204] W. S. Cleveland, *The elements of graphing data*. Belmont, CA, USA: Wadsworth Publ. Co., 1985.

[205] E. R. Tufte, *The visual display of quantitative information*. Cheshire, CT, USA: Graphics Press, 1986.

[206] S. Kosslyn, *Graph Design for the Eye and Mind*. Oxford University Press, USA, 2006.

[207] W. Tobler, "Thirty five years of computer cartograms," *Annals of the Association of American Geographers*, vol. 94, pp. 58–73, 2004.

[208] P. Isenberg, A. Bezerianos, P. Dragicevic, and J.-D. Fekete, "A study on dual-scale data charts," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2469–2478, 2011. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2011.160

[209] G. W. Furnas, "Generalized fisheye views," in *Proc. of the ACM SIGCHI Conf. on Human Factors in Computing Systems*, ser. CHI '86. New York, NY, USA: ACM, 1986, pp. 16–23. [Online]. Available: http://dx.doi.org/10.1145/22627.22342

[210] E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. D. DeRose, "Toolglass and magic lenses: The see-through interface," in *Proc. of the ACM Annual Conf. on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '93, 20th. New York, NY, USA: ACM, 1993, pp. 73–80. [Online]. Available: http://doi.acm.org/10.1145/166117.166126

[211] A. Martin and M. Ward, "High dimensional brushing for interactive exploration of multivariate data," in *Proc. of the IEEE Conf. on Visualization*, ser. VIS '95, 6th, 1995, pp. 271–. [Online]. Available: http://dx.doi.org/10.1109/VISUAL.1995.485139

[212] W. S. Cleveland, "Robust locally weighted regression and smoothing scatterplots," *Journal of the American Statistical Association*, vol. 74, pp. 829–836, 1979. [Online]. Available: http://dx.doi.org/10.1080/01621459.1979.10481038

[213] (2013) Dojo toolkit. The Dojo Foundation. [Online]. Available: dojotoolkit.org

[214] M. Bostock. (2013) D3.js data-driven documents. [Online]. Available: www.d3js.org

[215] E. Barse, H. Kvarnstrom, and E. Jonsson, "Synthesizing test data for fraud detection systems," in *Proc. of the Annual Computer Security Applications Conf.*, ser. ACSA '03, 19th, 2003, pp. 384–394. [Online]. Available: http://dx.doi.org/10.1109/CSAC.2003.1254343

[216] D. R. Jeske, B. Samadi, P. J. Lin, L. Ye, S. Cox, R. Xiao, T. Younglove, M. Ly, D. Holt, and R. Rich, "Generation of synthetic data sets for evaluating the accuracy of knowledge discovery systems," in *Proc. of the ACM Int'l. Conf. on Knowledge discovery in data mining*, ser. KDD '05, 11th. New York, NY, USA: ACM, 2005, pp. 756–762.

[217] P. J. Lin, B. Samadi, A. Cipolone, D. R. Jeske, S. Cox, C. Rendon, D. Holt, and R. Xiao, "Development of a synthetic data set generator for building and testing information discovery systems," in *Proc. of the Int'l. Conf. on Information Technology: New Generations*, ser. ITNG '06, 3rd. Washington, DC, USA: IEEE Computer Society Press, 2006, pp. 707–712. [Online]. Available: http://dx.doi.org/10.1109/ITNG.2006.51

[218] K. Houkjær, K. Torp, and R. Wind, "Simple and realistic data generation," in *Proc. of the Int'l. Conf. on Very Large Data Bases*, ser. VLDB '06, 32nd. VLDB Endowment, 2006, pp. 1243–1246. [Online]. Available: http://dl.acm.org/citation.cfm?id=1182635.1164254

[219] M. A. Whiting, W. Cowley, J. Haack, D. Love, S. Tratz, C. Varley, and K. Wiessner, "Threat stream data generator: Creating the known unknowns for test and evaluation of visual analytics tools," in *Proc. of the ACM AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*, ser. BELIV '06. New York, NY, USA: ACM, 2006, pp. 1–3. [Online]. Available: http://dx.doi.org/10.1145/1168149.1168166

[220] J. Heer and M. Agrawala, "Design considerations for collaborative visual analytics," *Information Visualization*, vol. 7, no. 1, pp. 49–62, 2008. [Online]. Available: http://dx.doi.org/10.1057/palgrave.ivs.9500167

[221] J. M. Heer, "Supporting asynchronous collaboration for interactive visualization," Ph.D. dissertation, University of California at Berkeley, Berkeley, CA, USA, 2008, aAI3353319.

[222] B. Lin, W.-S. Ho, B. Kao, and C.-K. Chui, "Adaptive frequency counting over bursty data streams," in *Proc. of the IEEE Symp. on Computational Intelligence and Data Mining*, ser. CIDM '07, 2007, pp. 516–523. [Online]. Available: http://dx.doi.org/10.1109/CIDM.2007.368918

[223] Q. He, K. Chang, and E.-P. Lim, "Using burstiness to improve clustering of topics in news streams," in *Proc. of the IEEE Int'l. Conf. on Data Mining*, ser. ICDM '07, 7th, 2007, pp. 493–498. [Online]. Available: http://dx.doi.org/10.1109/ICDM.2007.17

[224] Y. Li, X. Lv, and H. Wang, "A dynamic burst detection model over data streams," in *Proc. of the Int'l. Conf. on Mechanic Automation and Control Engineering*, ser. MACE '11, 2nd, 2011, pp. 7100–7103. [Online]. Available: http://dx.doi.org/10.1109/MACE.2011.5988686