FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

A Scalable Visualization of Set-Typed Data

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Martin Wortschack
Matrikelnummer 0627573

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Mag. Dr. Silvia Miksch
Mitwirkung: Dr. Dipl. Ing. Bilal Alsallakh

Wien, 31. März 2016

Martin Wortschack

Silvia Miksch

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

A Scalable Visualization of Set-Typed Data

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Business Informatics

by

Martin Wortschack
Registration Number 0627573

to the Faculty of Informatics

at the Vienna University of Technology

Advisor: Univ.Prof. Mag. Dr. Silvia Miksch
Assistance: Dr. Dipl. Ing. Bilal Alsallakh

Vienna, 31st March, 2016

Martin Wortschack

Silvia Miksch

Erklärung zur Verfassung der Arbeit

Martin Wortschack
Zieglergasse 14/2/1, 1070 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 31. März 2016

Martin Wortschack

Acknowledgements

First of all, I would like to thank my parents for their support during my studies. Their advice and encouragement helped me a lot to successfully finish my thesis.

Furthermore I want to thank my advisors Silvia Miksch and Bilal Alsallakh for their great assistance during my thesis. They not only supported me by providing scientific input and by proof reading the thesis but also contributed with their ideas in the process of developing a functional prototype.

Finally, special thanks go to my girlfriend Lena Richter for proof reading the thesis and for participating in the evaluation and to Lukas Pacha and Mathias Frey who volunteered for testing and evaluating the prototype and provided constructive feedback.

Kurzfassung

Mengenbasierte Daten spielen in der Informationsvisualisierung eine wichtige Rolle. Sie werden hauptsächlich genutzt, um Beziehungen zwischen Sets und Elementen zu repräsentieren, wie zum Beispiel welche Länder (Sets) ein bestimmtes Produkt (Element) exportieren, oder welchen Genres (Sets) ein Film (Element) angehört. Mengenbasierte Daten kommen in unterschiedlicher Form vor und dienen als Datenmodell in verschiedenen Analyse-Szenarien.
Skalierbarkeit ist eine der größten Herausforderungen im Zusammenhang mit mengenbasierten Daten. Euler- und Venn-Diagramme zählen zu den bekanntesten Visualisierungen in diesem Zusammenhang, da sie auf einfache Weise die grundlegenden Konzepte der Mengenlehre abbilden. Trotz ihrer Beliebtheit und der weiten Verbreitung in unterschiedlichen wissenschaftlichen Feldern, sind diese Diagramme nicht in der Lage, mehr als drei Sets darzustellen, ohne dabei enorm an Komplexität zuzulegen. Dadurch eignen sie sich im Allgemeinen nicht, um Daten zu analysieren und zu visualisieren, die hunderte Sets beinhalten. Dies trifft jedoch auf eine Vielzahl von Daten aus der realen Welt zu. Neben Euler- und Venn-Diagrammen wurden in der Vergangenheit etliche Visualisierungen für mengenbasierte Daten entwickelt. Die meisten dieser Visualisierungen skalieren allerdings entweder mit der Anzahl der Elemente oder mit der Anzahl der Sets.
Ziel dieser Arbeit ist die Entwicklung eines funktionellen Prototyps, der Skalierbarkeit sowohl in der Anzahl der Elemente, als auch in der Anzahl der Sets bietet. Die vorgestellte Visualisierung, auch `Scets`[1] genannt, nutzt verschiedene Methoden zur Aggregation und stellt die aggregierten Daten in einer übersichtlichen Matrix dar. Weiters ermöglicht diese Visualisierung Nutzern die Daten interaktiv zu untersuchen. Der entwickelte Prototyp setzt auf moderne Web-Technologien. Mit Hilfe eines serverseitigen Backends können große Datenmenge verarbeitet werden und einer Menge an Benutzern über eine webbasierte Oberfläche zugänglich gemacht werden. Zwei Anwendungsfällen veranschaulichen, wie die Visualisierung genutzt werden kann, um Echtdaten zu untersuchen und inwiefern der Nutzer dabei unterstützt wird, neue Erkenntnisse aus den visualisierten Daten zu gewinnen.

[1] Der Prototyp ist verfügbar unter: `http://scets.sybdev.com`

Abstract

In information visualization set-typed data refers to datasets that represent element-set memberships, such as which countries (sets) produce a certain product (element), or which genres (sets) a movie (element) belongs to. Set-typed data appears in various forms and can serve as a data models in various data analysis scenarios.
One of the main challenges in the context of set-typed data visualization is scalability. Traditionally, Euler and Venn diagrams count as two of the most popular set visualizations that depict the concepts from set theory. However, despite the widespread usage of these diagrams across several scientific fields they lack the ability of visualizing more than three sets without becoming too complex. This limits their applicability to data analysis scenarios that involve hundreds of sets such as the world countries. Besides Euler and Venn diagrams a variety of visualization techniques for set-typed data has been developed over the past. Typically, existing techniques scale well with either an increasing number of elements or an increasing number of sets.
The goal of this thesis is to develop a set visualization technique which offers high scalability in both the number of sets and elements. The proposed technique, called `Scets`[2], employs different aggregations of set-type data, and uses a matrix layout to visualize the aggregated information. Furthermore, it allows users to explore the aggregated information interactively. The implemented prototype uses modern web technologies which make the visualization both able to handle a large amount of data using server-side backend, and accessible to a wide range of users using web-based frontend. Two different use cases demonstrate how the proposed visualization technique helps to investigate real-world data and enables users in an intuitive way to reveal several patterns which could not be easily detected by other visualization techniques.

[2]The prototype is available at `http://scets.sybdev.com`

Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

1.1 Motivation and Problem Definition

Approximately 90% of all the data in the world has been generated over the last two years [Sci]. As a result, the amount of information which is available to business analysts and decision makers has increased tremendously during this period of time. However, the ability to process and interpret this vast amount of data in a meaningful way has only increased little. Reliable data is the prerequisite for making good decisions. But without the ability to make sense of the available information even the most accurate and reliable data is useless. Data visualization offers powerful means to make sense of large amounts of data. The majority of human sense receptors reside in the eye making vision a powerful input channel from the surrounding world. Colin Ware, one of the leading experts in visual perception describes the importance of data visualization in the following way [War13, p. xvi]:

> "Why should we be interested in visualization? Because the human visual system is a pattern seeker of enormous power and subtlety. The eye and the visual cortex of the brain form a massively parallel processor that provides the highest-bandwidth channel into human cognitive centers. At higher levels of processing, perception and cognition are closely interrelated, which is why the words *understanding* and *seeing* are synonymous. We know that the visual system has its own rules. We can easily see patterns presented in certain ways, but if they are presented in other ways, they become invisible... The more general point is that when data is presented in certain ways, the patterns can be readily perceived. If we can understand how perception works, our knowledge can be translated into rules for displaying information. Following perception-based rules, we can present our data in such a way that

> the important and informative patterns stand out. If we disobey the rules, our data will be incomprehensible or misleading."

Using appropriate visual representation of data allows data analysts to gain an overview of the data and to recognize patterns and correlations within the data that were unknown before. Information visualization (InfoVis) is a sub-domain of data visualization that focuses mainly on abstract data that has no inherent spatiality. Stuart Card defines information visualization as "... a set of technologies that use visual computing to amplify human cognition with abstract information." [Jac12, p. 546] Both data visualization and in particular information visualization have become an important step in the data analysis process. It is much easier to extract valuable information from visualized data than from looking at the raw data. In addition, visualization gives answers faster and helps to identify and understand cause-effect relationships. To better illustrate these arguments, let's take a look at the data in Table 1.1:

Skill	**Lisa**	**Barbara**	**Thomas**
Project Management	1	0	1
Programming	1	1	0
Communication	0	1	1
Graphic Design	1	0	0
Administrative	1	1	0

Table 1.1: Example data containing skills of three employees

By looking at the raw data, one can tell that the table shows a skill matrix of three persons Lisa, Barbara and Thomas. But what about answering specific questions such as "Which is the most skilled employee, i.e., which person possesses the most skills?" or "How many skills do Lisa and Barbara have in common?" In the example above there are only five data rows which allows us to find answers to these questions in a fairly short period of time. But when we have to deal with large data tables containing hundreds or even thousands of records we cannot simply process the huge amount of information by simply scanning the raw data. This is where visualization comes in handy.

Now let's look at the visualized data. Both diagrams in Figure 1.1 and Figure 1.2 visualizes the sample data from Table 1.1 in a more readable form. They present data in different ways allowing us to answer different questions. The first diagram (see Figure 1.1, called a Venn diagram, depicts the three persons as circles and the skills as dots inside these circles. The Venn diagram also reveals information which was not obvious in the table: Lisa is the only person who can work as a Graphic Designer. The second diagram (see Figure 1.2), a histogram, gives some indication of the frequency distribution of the skills among the three persons. It shows that Lisa is the most skilled person in the group, possessing four skills in total.

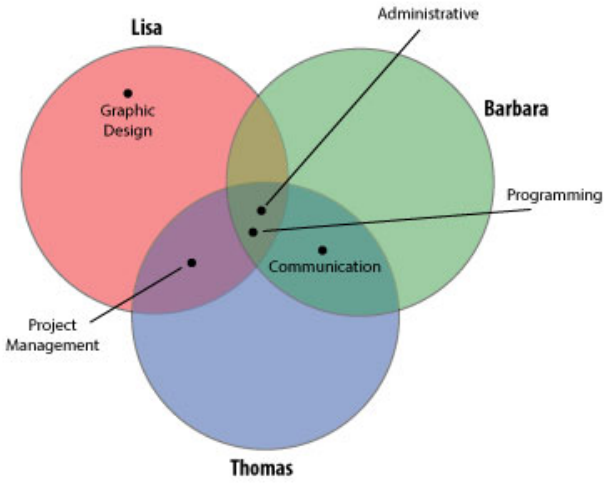


Figure 1.1: A Venn diagram visualizing set memberships for elements (i.e., the relationship between employees (sets) and skills (elements).)

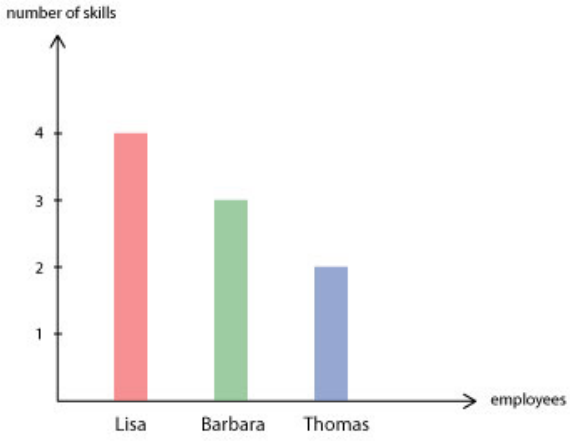


Figure 1.2: A histogram showing the distribution of skills for three employees.

Data may be available in different form and often has to be pre-processed to be visualized. In the context of this thesis I will focus on set-typed data. The concept of sets is well known from mathematics. A set is defined as a collection of distinct elements. Set-typed data is mainly used to represent element-set memberships. Sets are also frequently used in a variety of scientific fields such as biology and computer science.

Let us take another look at the sample data in Figure 1.1. Employees are represented as a columns (i.e., sets) and skills (i.e., attributes resp. elements) are represented as rows resembling an adjacency matrix. In this example Boolean attributes are used to specify which elements belong to a certain set. For example, Lisa possesses 'Project Management' skills but she does not have 'Graphic Design' skills. Several techniques for visualizing set-typed data have been developed. Examples include Euler and Venn diagrams, Set O'Grams [FMH08] and Radial Sets [AAMH13]. A recent State of the Art report by Alsallakh et al. [AMA+14] provides a comprehensive list of proposed set visualization techniques. Most tools perform well as long as only a handful of sets has to be visualized but they lack the ability of handling a large number of sets properly. In the majority of cases the visualization becomes either too complex or there is too little space to fit all relevant data. The Venn diagram shown in Figure 1.1 enables us to quickly figure out the number of skills Lisa and Barbara have in common. As long as only up to three sets have to be visualized, a Venn diagram is a good choice. But as soon as four or more sets are involved, we have to use different shapes other than circles to represent the involved sets. As a result of the diagram's inherent limitation in terms of the number of sets it can handle, the visualization soon gets very complex. Scalability has become a key requirement for visualization tools when addressing real-world problems. Thus, I developed a novel visualization technique, called `Scets`, which is capable of handling a large number of sets. This technique focuses on scalability by means of data abstraction and aggregation and scales well both in the number of sets and in the number of elements. In order to address a large group of users, it is developed as a web application.

1.2 Research Questions

The purpose of my research is to investigate new visualizations of set-typed data that are scalable both with the number of sets and with the number of elements. According to a recent survey on set visualization [AMA+14], aggregation-based visualization techniques are suited to address scalability with the number of elements. The survey also proposes that matrix-based techniques are suited to address scalability with the number of sets.

My research hypothesis is that aggregation-based and matrix-based techniques can be combined to address the scalability with both sets and elements. In particular, I aim to answer the following research questions.

- **RQ1**: How to represent aggregates of set elements using a matrix layout that scales with the number of sets? Answering this question involves investigating

different alternatives for the visual design, and making use of interaction to aid the understanding of visual representation and support set-based queries over the elements.

- **RQ2**: Which scalability advantages with the number of sets do combined aggregation-based and matrix-based representations have? Can they handle 100 sets, 500 sets, 1000 sets?
- **RQ3**: How intuitive and usable is a new visual representation of set-typed data, and which real-world analysis scenarios can profit from it?

1.3 Methodological Approach

The methodological approach process comprises the following steps:

1. **Research and Literature Review**
   - Analyzing the properties of existing solutions based on literature reviews and usage examples. (State of the Art research)
   - The main focus of this research are relevant visualization techniques for set-typed data and the challenge of scalability in this regard.
2. **Design**
   - Evaluation of available tools/technologies for prototype implementation.
   - User interface mockups will be created during an iterative design process.
   - The proposed design will focus on scalability by means of data abstraction and aggregation.
3. **Prototype Development**
   - A prototype will be implemented that solves the problem of scalability better than existing comparable solutions.
   - The prototype will be developed as a client-side web application built with modern web technologies such as HTML5, CSS3 and Javascript.
   - A high level of accessibility will be obtained as the visualization will be available to a large number of users who can easily run and test the prototype.
   - The web-based implementation will also allow for covering several usage scenarios by uploading various data via the web browser.
4. **Evaluation**
   - The implemented prototype will be used as the main evaluation method. Experiments will be performed with the prototype to discuss feasibility and optimality of the proposed solution.

- The evaluation will be based on usage scenarios and qualitative user feedback in terms of usability and intuitiveness.
- Potential side-effects such as performance issues will also be examined in this step.

1.4 Structure of the Thesis

The thesis is structured in as follows:

- **Chapter 2 Related Work**: This chapter provides an overview of work that is related to this thesis. Important techniques within the context of this thesis will be categorized into two categories and relevant examples for each category will be explained in detail. Moreover the presented techniques will be analyzed in terms of scalability.
- **Chapter 3 The Scets Visualization Technique**: This chapter starts off with motivational aspects of set-typed data visualization and an explanation of the design requirements of the implemented visualization. Following this, the final user interface design and its components are introduced. Finally, the most important features and implementation details of the prototype, such as the binning algorithm, the matrix layout in combination with its aggregation feature and the main interaction possibilities will be introduced and explained in detail.
- **Chapter 4 Implementation**: At the beginning of chapter 4 the technologies and frameworks used to build the prototype will be described briefly. Following this, a detailed description about the project structure and the setup process is given. This chapter is concluded with a technical documentation of the implemented visualization prototype.
- **Chapter 5 Use Cases**: In this chapter I will describe some data sets that have been used during the implementation phase to constantly test the current state of the prototype. In addition, I will give some examples of patterns revealed in the data.
- **Chapter 6 Evaluation**: This chapter deals with the evaluation of the prototype's usability and intuitiveness. Furthermore, I will reveal potential limitations of the visualization and assess user feedback.
- **Chapter 7 Discussion and Future Work**: In this chapter I provide a summary of the achieved results and discuss them in detail. Finally I will summarize some ideas for further improvements and additional features which have been collected during the implementation and evaluation phase of my thesis.
- **Chapter 8 Summary and Conclusion**: The thesis ends with a summary and conclusion of the work and the obtained results.

CHAPTER 2

Related Work

The following chapter gives a compact overview of selected relevant techniques for set visualization. I adopted the classification of techniques from a recent state of the art report by Alsallakh et al. [AMA+14]. At the beginning of this chapter Euler and Venn diagrams will be introduced as they provide the basis for most recent visualization techniques. Within the context of this thesis I mainly focus on two visual categories: aggregation-based and matrix-based visualization techniques. These categories are most relevant for the implementation of a prototype and several techniques will be presented and analyzed with regards to their scaling abilities in the following subsections. Other techniques will be briefly described under the Subsection 2.4 at the end of this chapter. The chapter is concluded with a summary.

2.1 Euler and Venn Diagrams

Euler and Venn diagrams are numbered among the most popular visualizations for set-typed data. They provide a simple way to depict concepts from set theory, such as union, intersection, subset, etc. In Euler diagrams sets are represented by geometric shapes (e.g., circles, ellipses or any other closed path) and set intersections are represented by overlapping shapes. Some Euler-based visualization techniques create areas whose size is proportional to the number of containing elements, whereas other techniques simply illustrate the intersections. Both, Euler and Venn diagrams look similar but there is one major difference, as illustrated in Figure 3.13: Euler diagrams only show a region if it contains elements, whereas Venn diagrams show regions for all possible combinations no matter if the region contains elements or not. Thus, all Venn diagrams are also Euler diagrams but not the other way around.

Euler and Venn diagrams are extensively used in biology (see Figure 2.2 as an example), economics and computer science to name a few application domains. Both diagrams cannot handle a large number of sets as the number of potential subsets and overlaps

(a) (b)

Figure 2.1: Difference between Euler and Venn diagrams: (a) an Euler diagram shows only the relationships that exist, (b) a Venn diagram shows all possible relationships, even if some areas don't contain any elements (areas labelled with ø)

grows exponentially with the number of sets which causes the visualization to become very complex and confusing. Considering a system of m sets, the maximum number of overlaps will be 2^m. Thus, Venn diagrams typically represent two or three sets. Several variants for Venn and Euler diagrams exist which try to overcome this limitation. The number of representable sets varies and heavily depends on the curve shape. For example, Venn diagrams that deploy ellipses instead of circles can visualize up to five overlapping sets as shown in Figure 2.2.

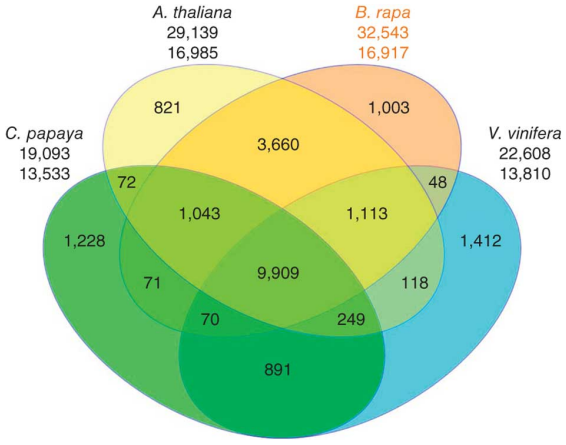


Figure 2.2: Venn diagram showing unique and shared gene families between and among four sequenced dicotyledonous species (B. rapa, A. thaliana, C. papaya and V. vinifera) [WWW+11]

2.2 Aggregation-based set visualization techniques

With an increasing number of elements it becomes more difficult to visualize set membership of individual elements. The limited number of representable sets is one of the major problems of Euler-based techniques as described in Section 2.1. Thus, these diagrams are generally not suited for visualizing a large number of sets.

Aggregation-based visualization comprises techniques that try to obtain scalability by omitting individual elements. Many techniques use abstraction in the sense that they don't show individual elements but rather express information in a summary form, i.e., through frequency representations of the data. Thereby multiple data elements are combined into a single visual element and reduce the total number of data being displayed. Detailed information on element level can be obtained through interaction. This makes aggregation-based visualizatoin techniques highly scalable in the number of elements and supports data analysis as it facilitates the investigation of data.

2.2.1 Mosaic Plots

Mosaic plots [Hof00] enable powerful visualization of multivariate categorical data. Whereas bar charts are mainly used to display univariate data, mosaic plots enable users to examine relationships between multiple categorical variables. Typically a mosaic plot starts with a single rectangle which is divided into two horizontal bars to represent the probabilities of the first categorical variable. Then each bar is split horizontally into sections based on the second variable. This partitioning process is repeated for each categorical variable. Optional spaces between sections are conventional an increase the graph's readability. Figure 2.3 shows a mosaic plot for three categorical variables.

The more categorical variables have to be displayed, the more complex mosaic plots become. Besides, there are several other problems that exist in mosaic plots. For example, it is very difficult to compare the size of rectangles when their aspect ratios vary extremely. Another common problem is that labels often overlap or cannot be clearly associated with rectangles. The reason is that sections are both horizontally and vertically subdivided.

2.2.2 Set O'Gram

A *Set O'Gram* is a type of interactive bar graph that looks similar to a histogram at a first glance. Each bar represents a set and is divided into sections that correspond to elements of different degree [FMH08]. An element's degree is defined as the the number of sets it belongs to. That means, starting from the bottom, the first block depicts the amount of items having only one element. The second block represents data items appearing in combination with exactly one other element (not more or less), and so on. A block's height is proportional to the number of elements within this section. Blocks within the

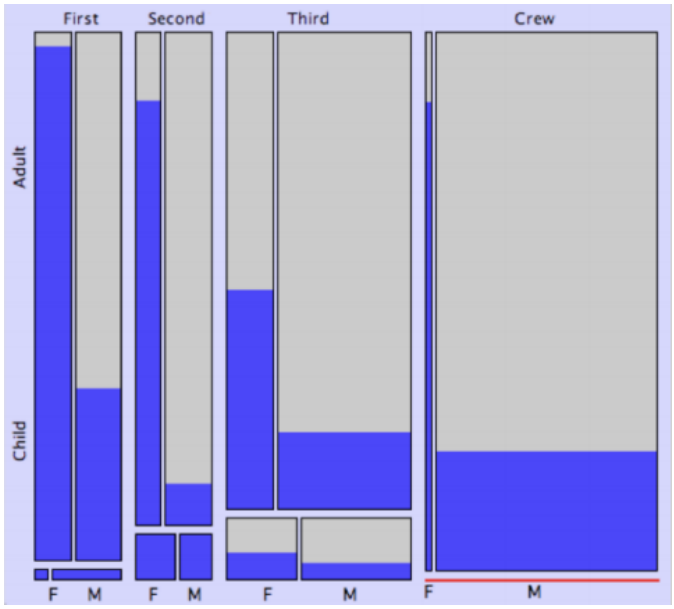


Figure 2.3: A mosaic plot that visualizes mortality rates aboard the Titanic influenced by three categorical variables: age, sex and passenger class.

same bar can be distinguished from each other by varying width. The bottommost block of each bar consumes the full width. Each of the stacked blocks decreases in width by a certain value. A striped bar background supports the comparison of block widths. This already gives valuable information about the data. Wide blocks indicate elements of lower degree, whereas narrow blocks indicate elements of higher degree, i.e., elements that occur in combination with several other elements. Through user interaction set overlaps become visible. By moving the mouse over a block not only the respective block is highlighted but all corresponding blocks in other bars are highlighted too. This enables the identification of the number of elements that belong exclusivley to one set and the number of elements that occur in k other sets.

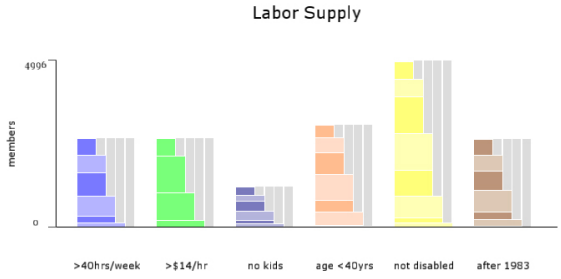


Figure 2.4: Set O'Gram [FMH08] of labor supply showing 6 categories

2.2.3 Radial Sets

Radial Sets [AAMH13] is an aggregation-based visualization technique for analyzing large overlapping sets that extends the basic idea of Set O'grams. In the main view sets are arranged in a circular layout where each segment of the circle corresponds to a particular set from a given input file. Elements in each set are divided into groups according to their degree. This means that the outermost group within a set contains elements that do not occur in any other set except for the given set itself (degree 1). Going from the outside to the inside, the second group in a set contains elements that exist in one other set (degree 2). The groups within a set are visualized as histograms, i.e., the size of a given group is proportional to the number of elements within the given group. (Figure 2.5) This makes radial sets highly scalable in the number of elements.

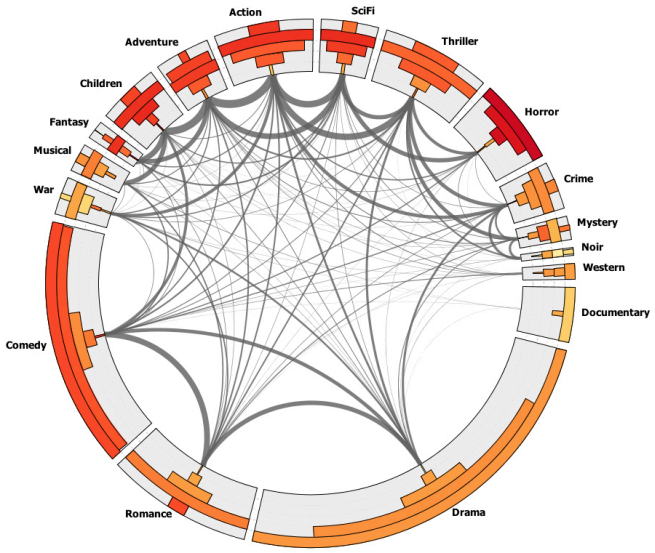


Figure 2.5: Radial Sets [AAMH13] arranges sets in a radial layout and represents elements as histograms within the respective circular segment.

Overlaps between two sets are displayed as links connecting the involved circular segments. Overlaps of higher degree are visualized with bubbles combined with arrow heads forming hyperedges. Thick connecting arcs indicate large overlaps, whereas thin links indicate small overlaps. The histogram bars can be colored according to an arbitrary element attribute. In addition to the main view, summary views show detailed information about sets, elements and overlaps. The summary views deploy further visualization elements to enable task-driven analysis. Examples include a bar chart that depicts the size of sets and a histogram of elements binned by degree. Through interaction elements can be selected based on set memberships and attributes. Selected elements will be highlighted both in the main and the summary views. Hovering the mouse over a histogram or a connection arc displays the number of elements in the corresponding histogram

resp. overlap. As noted earlier, Radial Sets scale very well in the number of elements. However, Radial Sets is limited to a small number of sets (20-30 [AMA+14]). For a larger number of sets this technique does not scale well as the visualization becomes very complex.

2.2.4 UpSet

UpSet [LGS+14] is an interactive tool for visualizing and analyzing set-typed data. In fact, UpSet is not a pure aggregation-based visualization as it combines various concepts from several visualization techniques. I'd rather prefer to classify UpSet as a hybrid between aggregation-based and matrix-based technique. Yet it is listed under this section just for the sake of convenience. UpSet's main focus lies on the relationship between multiple sets and on the general properties of a set. The layout is comprised of two separate views: the set view and the element view. In the set view typical set operations, such as intersections, are visualized in a matrix layout where each column represents a set and each row represents a set combination. Hence UpSet can also be classified as a matrix-based visualization technique. Rows can be compared to overlapping areas in a Venn diagram. Sets involved in an intersection are represented as filled dots in the corresponding cell.

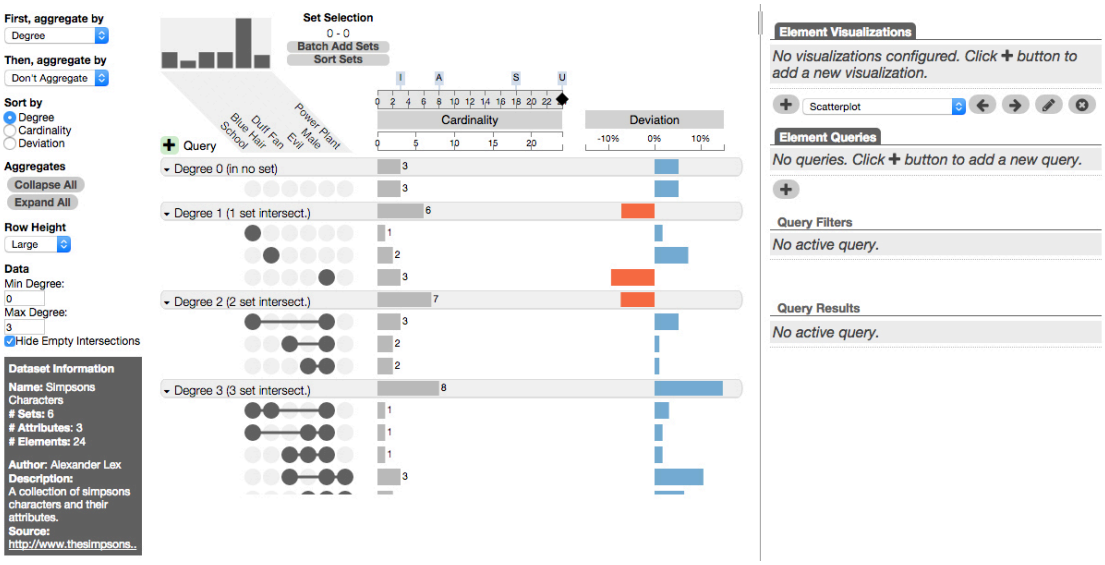


Figure 2.6: UpSet [LGS+14] showing the relationship between sets. The layout is split into set and element view.

Additional columns in the matrix layout provide more detailed information about the aggregates. One example are bars which are used to encode the cardinality of an intersection. As seen in Figure 2.7(b) there are four elements which are fruits and are also delicious. The length of a bar is proportional to the number of elements the intersection

contains. Another example are box plots which are used to show summaries of element attributes. This allows analysts to address several attribute-related tasks.

(a) (b)

Figure 2.7: UpSet [LGS+14] uses bar charts to visualize cardinality of intersections

UpSet's performance depends on a number of factors, such as the maximum non-empty degree of the intersections and the number of elements and sets. [LGS+14] Several performance optimization techniques have already been implemented to avoid memory leaks. While it scales very well in the number of elements (up to 50.000 elements [LGS+14]), its main scalability issue remains a large number of sets. As sets are arranged side by side in a tabular layout the available display space is limited. Figure 2.8 shows 42 selected sets which requires the user to horizontally scroll in order to investigate the remaining columns.

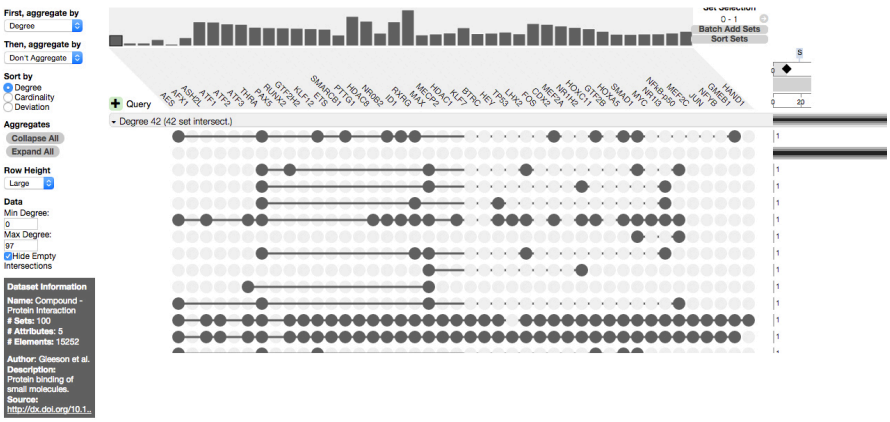


Figure 2.8: UpSet's [LGS+14] element view with 42 sets selected

2.3 Matrix-based set visualization techniques

Matrix-based set visualization methods are typically used to show relationships between sets and elements with the help of a flexible grid layout. Rows and columns represent items and values in each cell and show relationships. Such approaches allow decent scalability in both the number of elements and sets. However, these approaches are often

limited in the number of presentable set relations as stated in the survey by Alsallakh et al. [AMA+14]

2.3.1 ConSet

ConSet [KLS07] is an interactive visualization tool that supports users in exploring relationships among multiple sets. ConSet uses a permutation matrix and map sets and elements to rows and columns respectively. A gray-filled cell indicates a set-element membership. Each set is identified by a unique color. Due to the limited number of distinctive colors only 32 sets can be visualized. Besides reordering of rows and columns, the matrix view also facilitates aggregation of elements or sets. Aggregated elements are visualized in form of little bar charts in a separate row. In addition to the matrix view, two detail views (Dynamic Control view and Diagram Ordering view) exist where set and element information is displayed and where users can investigate relationships among up to three selected sets. Set relationships are visualized both as a Venn diagram and as a Fan diagram which is a novel diagram that enables comparison of two or three sets without any inconsistencies that may exist in Venn diagrams.

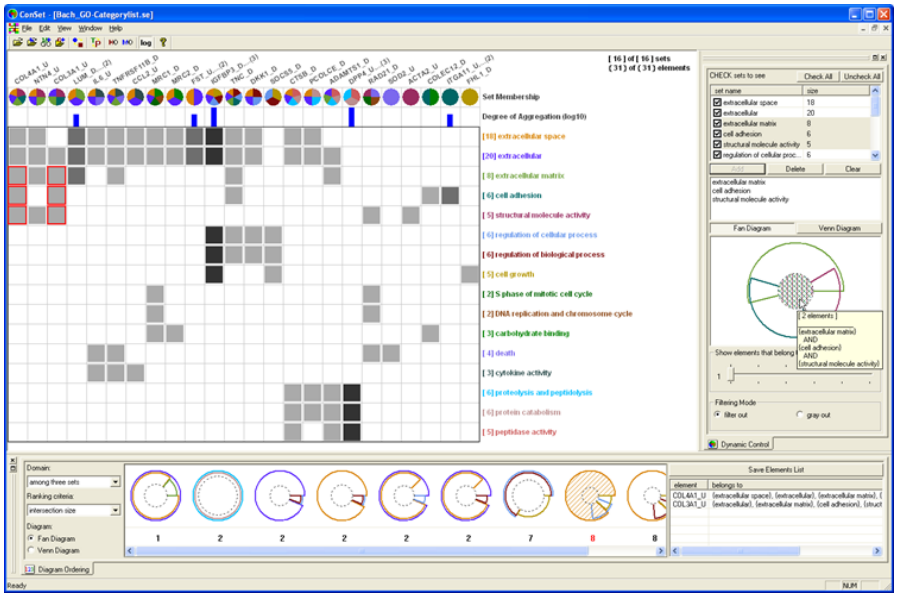


Figure 2.9: ConSet is composed of three views: (a) Permutation Matrix (b) Dynamic Control view enables the filtering of sets and elements (c) Diagram Ordering view shows the top 10 ranked diagrams of two or three sets by a selected ranking criterion [KLS07]

2.3.2 OnSet

OnSet [SMDS14] is a visualization technique for large scale binary set data. OnSet represents each set as a large rectangle. Elements of a given set are shown as pixels on the grid inside the rectangle. Each set is represented by the total number of elements, not

just by the elements it contains. An elements position is unique across all sets. Assuming there are three sets each of which contains eight unique elements which only appear in one set. This makes a total of 24 distinct elements. Each set would be composed of 24 pixels forming a grid. Elements which don't occur in a set are depicted as blank pixels. Through interaction element-set memberships can be easily identified. By hovering over an element its name will be displayed and its position will be highlighted in every set it belongs to as shown in Figure 2.10.

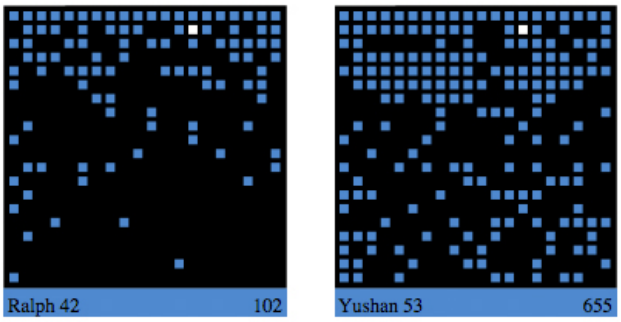
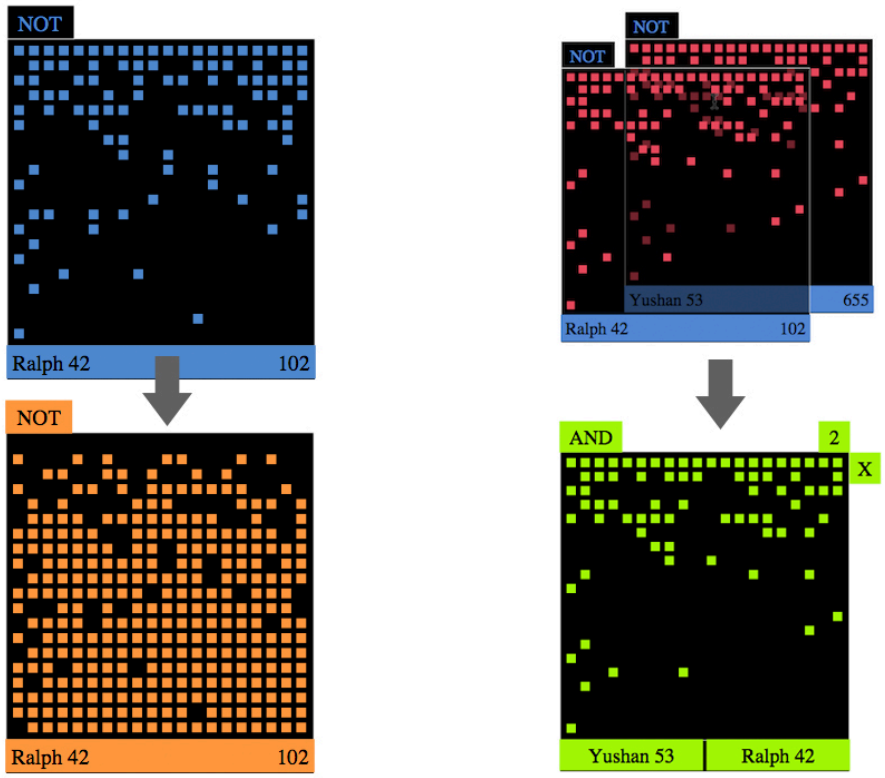


Figure 2.10: Elements get highlighted in every set by hovering over a pixel.

OnSet supports three basic set operations: complement (logical NOT), union (logical AND) and intersection (logical OR). By clicking on the "NOT" label on the upper edge of a set, the corresponding set's complement will be shown (see Figure 2.11(a)). Clicking the label again will display the basic set again. Set operations that involve multiple sets can be simulated via drag and drop operations. When a set is directly moved on top of another set, a new set will be generated, called a MultiLayer. By default a logical AND operation is applied resulting in the union of the two involved sets as shown in Figure 2.11(b). In addition a logical OR operation is supported which enables users to create intersections of multiple sets. The applied operation is labelled on top of the created MultiLayer. Users can switch between the two operations (AND, OR) by clicking on the corresponding label. Besides, the number and names of the involved sets are shown at the bottom of the MultiLayer.



(a) (b)

Figure 2.11: Set operations in OnSet [SMDS14]: (a) complement of a set, (b) intersection of 2 sets

Links between sets, so-called similarity bands, can be displayed to represent set overlaps. The thickness of the links is proportional to the similarity metric. Hovering over such a link will highlight all elements which occur in both sets. Sets which are not connected by the band fade out. OnSet can easily process and visualize sets with hundreds of elements. Since no aggregation is used to reduce the number of visible elements, scalability is affected when both the number of sets and elements increases.

2.3.3 AggreSet

AggreSet [YEB16] is a data exploration technique for set-typed data. It mainly focuses on scalable set exploration. Like UpSet [LGS+14], it can be classified as a hybrid visualization technique. AggreSet uses a matrix-based visualization for set relations in order to improve scalability. Moreover, it employs aggregation to achieve scalability in the number of elements. It is designed as a multi-view visualization technique which uses linked visualizations to support rich, contextual data exploration. Therefore, aggregations for each data dimension are created, such as sets, set-degrees and set-pair intersections.

The element count per aggregate for set-pair intersections is visualized in a scrollable and zoomable set-matrix which enables users to explore relations between different sets through interaction. Histogram views visualize the element count per aggregate for set lists, set-degrees and other attributes. AggreSet provides various opportunities for interaction to support users in exploring data. One of its key features, is the so-called 'result-preview', a linked brushing feature. By hovering over an aggregate glyph in one view, it is highlighted in every other aggregate view. Besides, the visualization technique allows users to filter elements of a selected aggregate by clicking. Three different filter modes are available in the set list: union (logical OR), intersection (logical AND) and complement (logical NOT). Furthermore, users can compare selections by locking an aggregate and moving the mouse over different aggregations. AggreSet inserts black 'compare-lines' on top of the 'results-preview' in order to support exploration.

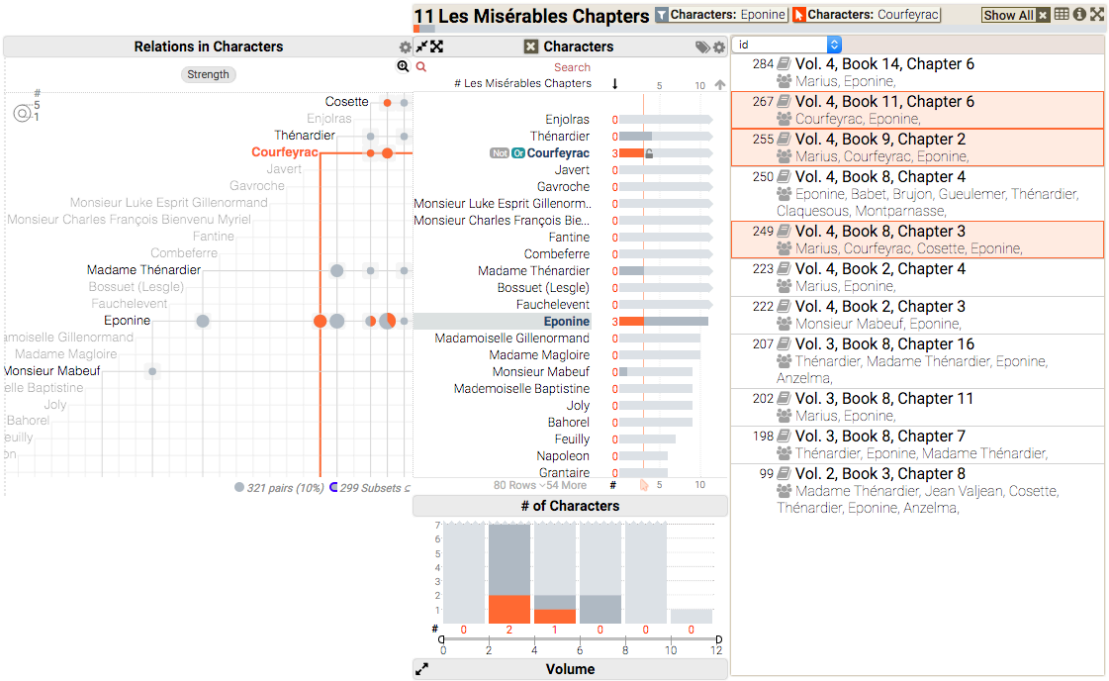


Figure 2.12: AggreSet [YEB16] is composed of multiple linked views (including an aggregate matrix and histograms) which visualize aggregates in various ways.

As for scalability, AggreSet scales well in the number of elements and is able to handle 100.000+ aggregated elements. [YEB16] However, it is limited to 50+ [YEB16] sets which can be attributed to the space-consuming matrix-view. Even though the matrix view supports zooming out and panning which allows showing more data and exploring areas outside the viewport, scaling to hundreds of sets is not feasible.

2.4 Other Techniques

Several additional techniques for visualizing set-typed data are mentioned in the comprehensive State-of-the-Art Report by Bilal Alsallakh, et al. [AMA+14] This section will give a brief overview of selected techniques which are less relevant in the context of this thesis however.

2.4.1 Overlays

Overlays focus on depicting set memberships by extending existing visualizations that provide specific context information. *Region-based* overlay techniques such as *Bubble Sets* [CPC09] draw colored areas for each set containing all elements that belong to it. These transparent areas are placed on top of an existing visualization. Figure-2.13(a) shows countries represented as circles arranged on a scatterplot. Countries that belong to the same continent are connected by hyperedges of different color. This way outliers can be easily identified. *Line-based* overlay techniques enable better distinction between the overlay visualization and the underlying base visualization by using lines instead of colored areas to represent set memberships. *Glyphs* and icons provide another option to represent set memberships in terms of overlays. Glyphs represent one or more variables of a data set through symbols or icons. Consequently they are often used to visualize multivariate data. Usually glyphs are geometric objects and the variables that have to represented are encoded by different properties of the glyph, such as color or size.

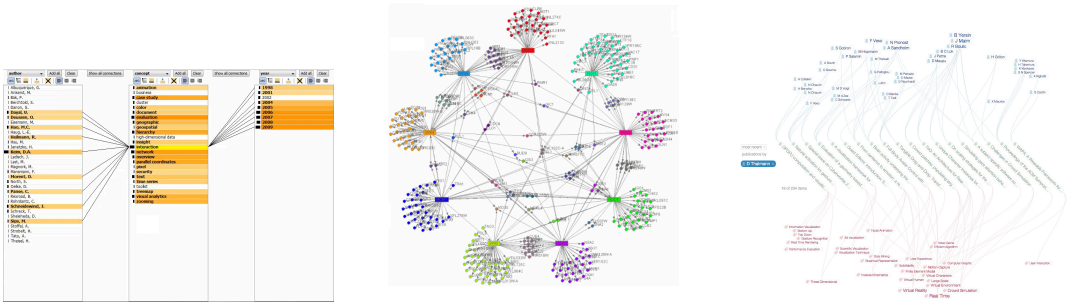
(a) (b) (c)

Figure 2.13: Overlay techniques: (a) Region-based overlay with Bubble Sets [CPC09], (b) Line-based overlay with LineSets [ARRC11], (c) color-coded glyphs [zon]

2.4.2 Node-link diagrams

The group of node-link diagrams includes several techniques to visualize the relations between elements and sets. These relations can be modelled as edges of a bipartite graph

[AMA+14]. Examples include *Jigsaw* [SGL08] *Anchored Maps* [Mis06] and *PivotPaths* [DRRD12].



(a) (b) (c)

Figure 2.14: Node-link diagrams: (a) Jigsaw [SGL08], (b) Anchored Maps [Mis06], (c) PivotPaths [DRRD12]

2.5 Summary

In this chapter selected relevant techniques for set visualization have been described in detail. First, Venn and Euler diagrams were introduced. Then existing techniques have been categorized into aggregation-based and matrix-based visualization techniques as proposed by Alsallakh et al. [AMA+14].

Venn and Euler diagrams are based on the set theory and are commonly used to depict relationships between sets. Typically circles or ovals are used in both diagrams to represent inclusion exclusion and intersections, although other shapes such as polygons can be used as well. As for the number of elements, Euler-based diagrams are capable of visualizing up to hundreds of elements. The number of representable sets varies between 3 (for circles) and up to 10+ (for polygons) depending on the shape. In general, such diagrams don't scale well to a large number of elements. Overlapping sets eventually lead to drawability issues and to visual complexity of such diagrams. As a result, distinguishing overlapping sets becomes a very difficult task.

Aggregation-based visualizations obtain scalability by combining multiple data elements into a single visual element. Typically, techniques that utilize aggregation of elements can handle a large number (thousands) of elements. However, usually such techniques lack the ability of scaling with an increasing number of sets. As an example, Mosaic plots [Hof00] only scale up to about 4 sets. Radial Sets [AAMH13] (20-30 sets) and Set O'Grams [FMH08] (about 50 sets) perform slightly better but still don't scale well to a large number of sets due to their visual complexity. UpSet [LGS+14], which supports tens of thousands of elements, lacks the ability to visualize more than about 20 sets

without requiring the user to scroll horizontally.

Matrix-based visualization methods allow scalability in the number of sets by arranging sets in a flexible grid. These approaches are often limited in the number of presentable set relations [AMA+14].
In ConSet [KLS07] set-element memberships are visualized in a permutation matrix where each element is represented in a column and each set is represented in a row and encoded by a unique color. Since elements are not aggregated, the matrix view is not scalable by the number of elements. Thus, approximately up to 100 elements can be visualized. Due to the limited number of distinctive colors the number of representable sets is limited to 32.
OnSet [SMDS14] represents each set as a so-called PixelLayer and only scales to a small number of sets (approximately 20). This is largely because the size of a PixelLayer is determined by the number of elements and as a result the size of each PixelLayer increases with an increasing number of elements.
AggreSet is able to handle 100.000+ aggregated elements [YEB16]. Nevertheless, due to it's space-consuming matrix view, AggreSet's scalability in the number of sets is limited to 50-100 sets.

It can be concluded, that existing techniques don't provide high scalability in both the number of sets and the number of elements. This is attributable to various causes, such as visual complexity, limited space, etc. In order to overcome these limitations, this work proposes a novel visualization technique that scales up to thousands of elements and to hundreds of sets by combining several benefits from aggregation-based and matrix-based techniques.

CHAPTER 3

The Scets[1] Visualization Technique

The primary goal of data visualization is to convey information to users efficiently with the help of graphical representation. Visualizations are intended to make complex data understandable. However, this is not always the case. Often times visualizations are rather confusing and misleading than helpful. Moreover, most existing visualization techniques for set-typed data lack the ability to scale well with an increasing number of sets and elements respectively as mentioned earlier in Chapter 1.1. Typically, the result is either a remarkable performance loss or the visualization gets very complex which makes it hard for humans to reveal patterns and answer specific questions about the visualized data. Thus, one of the key requirements of `Scets` is the possibility to process and visualize data that involves hundreds of sets and thousands of elements. `Scets` is an interactive, hybrid visualization technique that employs features from both aggregation-based and matrix-based visualization techniques in order to improve scalability. Besides scalability in both the number of sets and the number of elements, additional design requirements include the following:

- Interactivity: The prototype has to be developed as a dynamic visualization that empowers users to explore the visualized data in different ways.
- Intuitiveness: The user interface and moreover the visualization itself as well as the used graphical metaphors have to be intuitive. This implies that the visualization should not be misleading and should convey the information in the most intuitive way.

[1] The acronym is formed from SCalable sET viSualization. An online version is available for testing: `http://scets.sybdev.com`

- Accessibility: A high level of accessibility should be obtained. This is achieved as the visualization will be running in any modern web browser irrespectively of the installed operating system making the visualization available to a large number of users who can easily run and test the prototype.
- Performance: Even with large amounts of visualized data, the prototype has to achieve a short response time to avoid affecting the usability negatively.

3.1 Visual Design

The 'Design Triangle' (see Figure 3.1) proposed by Miksch and Aigner [MA14] formed the basis for the iterative design process which eventually led to the final design mockup as shown in Figure 3.2. This framework helped to identify three main questions: First, what kind of data has to be visualized, secondly which type of users are working with the visualization and finally what kind of tasks do they typically perform. Consequently, the following requirements could be identified:

- Data: set-typed, binary data in any application domains (e.g. biology or economics). For example, the data could be a skill matrix of employees. In general, `Scets` is especially designed to handle data comprising hundreds of sets, defined over tens of thousands of elements.
- Users: domain exports in any field working with set-typed data, such as HR managers who analyze employees skill data.
- Tasks: The analysis tasks listed below are should be supported.
  - T1: Analyze the distribution of elements per set according to their degrees, i.e., the number of sets they belong to.
  - T2: Identify elements in a set which are unique to it or are shared with as many sets as possible.
  - T3: Compare sets in terms of their distinctiveness and find the most and least distinctive set. A set's distinctiveness is defined in Section 3.2.7.
  - T4: Analyze which elements of a set are shared between other sets and detect the portion of elements appearing in other sets.
  - T5: Find all sets which contain a specific element.

Based on the above specification of data, users and tasks, suitable visual representations and interaction methods could be developed. The main focus of the created mockup was on the design of a combination of aggregation-based and matrix-based visualization in order to represent set-typed data and achieve the desired scalability. Based on the identified requirements above a mockup of the user interface was created. The final user interface is composed of three main components:

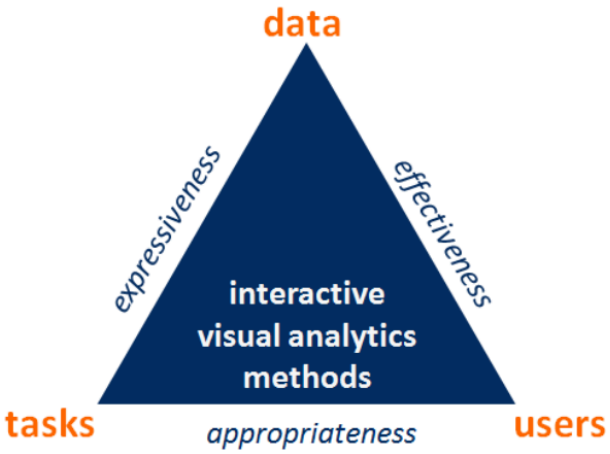


Figure 3.1: The Design Triangle framework: Data - Users - Tasks

1. Control Panel: The control panel is located on the top side of the user interface (UI) and contains control elements for sorting sets, opening the binning view, expanding and collapsing rows and removing the active selection.

2. Matrix View: The matrix view is the visualization's key component. It is where the data is depicted by using the visual metaphor. Elements are aggregated by degree and represented as colored circles arranged in a grid.

3. Element View: Next to the matrix view, the element view is situated. It consists of a table that displays detailed information about elements included in the current selection. This includes the name and the degree of each element as well list of sets it belongs to.

In the following I will introduce the basic idea behind `Scets` and furthermore describe how the design evolved during several iterations.

3.1.1 Visualizing the Sets: The Matrix View

The visual design of `Scets` is based on a simple idea: Sets and element degrees are visualized in a matrix layout where rows represent degrees and columns represent sets. Single sets are visualized as rectangles as shown in Figure 3.3 (1). Each rectangle is eighteen pixels wide in total, with a one pixel outline to distinguish neighboring sets. Sets are arranged next to each other and are separated by a two pixel space. The entire available screen width is used to fit as many sets as possible. As soon as a set no longer fits into a single row a new so called 'set group' is created in a new row, as can be seen in Figure 3.3 (4). Set groups contain multiple sets and are positioned one below the other. Whereas a set's width is defined by a fixed value, a set's height is determined by

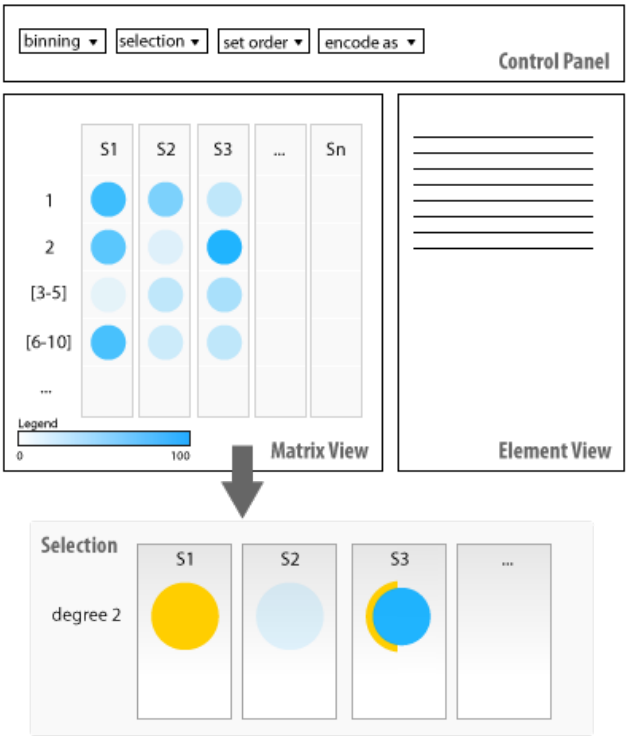


Figure 3.2: Conceptual design of the user interface

the number of visible elements within its content area. This visual metaphor allows for intuitive element-set membership visualization. Labels above each set display its name as it appears in the raw data set.

One of the main characteristics of effective information visualization systems is the ability to display results in a meaningful, not overwhelming way. Therefore, techniques that group data and visualize condensed results are necessary. This is where data aggregation comes into play. In general, data aggregation can be any process which displays a summarized form of priorly gathered information. Data aggregation is an essential part of `Scets`' visualization technique as it is the key prerequisite and enabling factor for scalability.

Like Radial Sets [AAMH13] and Set O'Grams [FMH08], `Scets` uses basic aggregation in order to group elements based on their degree. Once again, an element's degree denotes the number of sets it belongs to. For example, two elements X and Y appear in exactly three sets. Consequently, both elements are of degree three and will be grouped. It is important to note, that elements X and Y don't necessarily have to appear in the same sets. The only thing that matters in this regard is the total number of sets where they

are featured. Elements grouped by degree are represented as colored circle glyphs within a set. In the context of this thesis these circle glyphs are called subsets. Subsets are arranged in rows placed one above the other within a set, i.e., starting with degree one on top, followed by degree two right below and so on. Empty subsets are not displayed, leaving a blank space between other subsets. The bottommost subset represents the highest degree.
A subset's quantity is color-encoded. For this reason, a color scale is used to map each quantity value to a predefined range of possible colors. This means, that darkened circles indicate a high number of elements, whereas light colors denote a small number of elements. To support users in identifying a subset's total number of elements, a color legend is displayed above the matrix view.

`Scets` pursues the goal of high scalability in both the number of sets and the number of elements. Thus, it is designed for a large number of sets and elements. This implies that with an increasing number of sets there will be a high possibility of large degrees, i.e., in the order of 50 or 100. By visualizing each element degree in a separate row, the visualization will soon become very complex, especially if there is a need for multiple set groups due to the large number of sets. With hundreds of sets the number of degrees can be large which results in many rows. Moreover, the probability of degrees having zero elements is very high. Hence, there was a strong need for improving the basic approach by introducing advanced 2nd-level data aggregation which is explained in the upcoming section.

3.1.2 Advanced Aggregation

Grouping elements by degree reduces the number of displayed elements effectively but this method does not necessarily lead to the desired scalability. In order to ensure better scalability and solve the visual problem of increasing set height due to a large number of visible rows as explained in the previous section, data binning is used as an additional aggregation method. The term binning refers to grouping N data values into less than N groups. Data binning is especially well known from fields such as statistics, image processing, etc. A common example of binning in the context of information visualization is a histogram. As an example, to construct a histogram from people's ages, individual data items get arranged into intervals first. Subsequently, those groups, also called bins, can be visualized instead of each individual age. In the context of this work binning describes the procedure of combining a group of degrees, i.e., a group of succeeding rows in the matrix, into a single row, denoted as bin. Thus, the overall number of rows displayed at a time is reduced and a higher abstraction level is achieved. Broadly speaking, binning is defined as the mapping from an input, i.e., different levels of degrees, to an output, i.e., aggregated degrees.

In general, one has to differentiate between primitive *Equal-width binning* and data-driven *Equal-frequency binning* algorithms. Both, the *Equal-width binning* and the

Equal-frequency binning are unsupervised methods as they don't take the class information into account. Besides, both methods require that the number of total bins is specified manually. Determining the number of desired bins k often results from a trade-off between abstracting to a very low level for a small number of bins and ending up with empty bins in case of a high number of bins. Hence, `Scets` offers users the possibility to define the number of bins and the bin ranges manually according to what the data represents and which bins are useful. This feature is described in greater detail in Section 3.2.6.

An *Equal-width binning* algorithm simply divides the data into k intervals of equal size without taking the distribution of data into account. Thus, the bin width is computed by $(max - min)/k$ where max is the maximum value and min is the minimum value in a given dataset. As an example, bins of a histogram generally have same width, irrespective of the number of elements contained. Often the *Equal-width binning* works fairly well but in some cases most of the data is placed within just a few bins whereas the remaining bins contain hardly any data. This holds especially true for data distributions which are far away from a uniform distribution. To avoid such unequal distribution among bins *Equal-frequency binning* can be used which divides the data into k groups, each of which contains approximately the same number of values. Consider the sample data $[0, 4, 12, 16, 17, 18, 24, 26, 28]$. Table 3.1 and Table 3.2 compare the bin values and intervals computed by the two algorithms.

	Values	**Interval**
Bin 1	0, 4	[-, 10)
Bin 2	12, 16, 17, 18	[10, 20)
Bin 3	24, 26, 28	[20, +)

Table 3.1: Equal-width binning

	Values	**Interval**
Bin 1	0, 4, 12	[-, 14)
Bin 2	16, 17, 18	[14, 21)
Bin 3	24, 26, 28	[21, +)

Table 3.2: Equal-frequency binning

`Scets` utilizes a data-driven binning algorithm. Generally, this approach follows the idea of a histogram equalization, a signal-processing technique which aims to obtain equal distribution of elements among a given number of bins k. First, a histogram H is computed that shows the number of elements per degree. Then the number of elements placed in each bin is determined by dividing the total number of elements by the number of desired bins. Following this, the lower and upper bound for each of the k bins are computed. Thereby the algorithm ensures that each bin contains roughly the same amount of elements by trying to iteratively fit elements of the successive degree into the current bin. If the current bin has reached a certain threshold, a new bin is created and filled with elements from the degree. This procedure is repeated until all elements from the histogram H are allocated to the k bins. Since the binning algorithm strongly depends on the distribution of the data the computed bins might need further adjustments. Thus, users can change the binning settings manually at any time in the

'Binning Settings' menu. However, bin customization is not part of this section but is covered in Section 3.2.6.

Algorithm 3.1 shows an implementation of the binning algorithm in pseudocode with the following variables:

- H: a histogram of degrees
- k: the number of desired bins ($k = 5$ by default)
- n: the total number of elements
- s: the number of elements per bin which is defined as $s = n/k$
- *binSize*: the total number of elements of the current bin

Algorithm 3.1: Binning Algorithm

```
1  ind ⟵ 0;
2  leftElements ⟵ n;
3  for bin ← 0 to bin < k do
4  |   start[bin] ← ind;
5  |   binSize ← H[ind];
6  |   s ← leftElements/(k − bin);
8  |   while ind < n − 1&&(binSize + H[ind + 1]) ≤ s do
9  |   |   ind ← ind + 1;
10 |   |   binSize ← binSize + H[ind];
11 |   end
12 |   end[bin] ← ind;
13 |   leftElements ← leftElements − binSize;
14 |   ind ← ind + 1;
15 end
```

To account for the fact that each element in bin k is present k times in the visualization (i.e., in k different sets), the size of bin k is multiplied with its degree k. For this reason, there are two histograms computed in `Scets`. The first histogram is used in the Binning View (see Section 3.2.6) to simply show the number of elements per degree and to support users in making meaningful adjustments to the bin settings. The second histogram is used as an input parameter H for the binning algorithm (see Algorithm 3.1). In this case the k multiplier is applied to each bin, that is to say the total number of elements in the second degree is multiplied by two, the total number of elements in the third degree is multiplied by three, and so on. This ensures that `Scets` will be more balanced in the distribution of dark colors. Without applying the multiplier, the color distribution in the

visualization would be distorted as density of color might appear mostly in high-degree rows. For example, an item of degree 100 counts one time in the algorithm and in the binning view histogram, but shows up 100 times in the matrix view.

3.1.3 The Final Design

With the introduction of advanced aggregation (see Section 3.1.2) which groups degrees into bins, further enhancements to the basic design apply. These additional features are described in the following.

In the matrix view bins are represented as rows which can comprise multiple degrees. As shown in Figure 3.3 (3) bins are marked with a range shown in square brackets indicating the first and last degree included. Elements that belong to the same bin are aggregated. The so called 'aggregates' are represented as colored circles placed inside the dedicated set. Aggregates look like subsets at first glance. Nevertheless, aggregates can be distinguished from subsets based on their color which is explained in greater detail later in this section. Typically a set holds multiple aggregates, more precisely one aggregate for each bin. Aggregates at the same level in different sets belong to the same bin.
Right next to a bin's label a little toggle icon is placed which enables users to expand resp. collapse the corresponding bin (see Figure 3.3 (3)). Each set group comprises all bins. This allows expanding resp. collapsing a particular bin in every set group when clicking on toggle icon. Aggregates can be selected by clicking on them. This expands the related bin and displays all subsets of the clicked aggregate. More details on selection possibilities and expanding bins can be found in Section 3.2.

Usually an aggregate comprises several subsets, strictly speaking one for each degree included in the corresponding bin. As an example, the aggregate of bin [3-4] in the set u_12 shown in Figure 3.3 (2) contains two subsets, one for degree 3 and one for degree 4. They are shown when the respective bin is expanded. Just like aggregates, subsets can also be selected. A selected subset is marked blue. The portion of elements which is included in other subsets than the selected one, is visualized as a blue arc around the corresponding subset. Elements which are included in the selected subset are displayed in a table in the element view. Section 3.2.1 describes this interaction method extensively.

Aggregate quantities differ from subset quantities for the simple reason that the sum is bigger than the individual values. To allow better differentiation between aggregates and subsets in the UI, two different color scales are employed as shown in Figure 3.4. Without using separate color scales, the lowest color grade of the main color scale would be assigned to most inner bins. As a result, the user interface shows separate legends for aggregates and subsets respectively. Whereas the aggregate legend is always visible, the subset legend only appears when at least one bin is expanded. As soon as all bins are collapsed, the subset legend is hidden. Moreover, `Scets` provides the possibility of

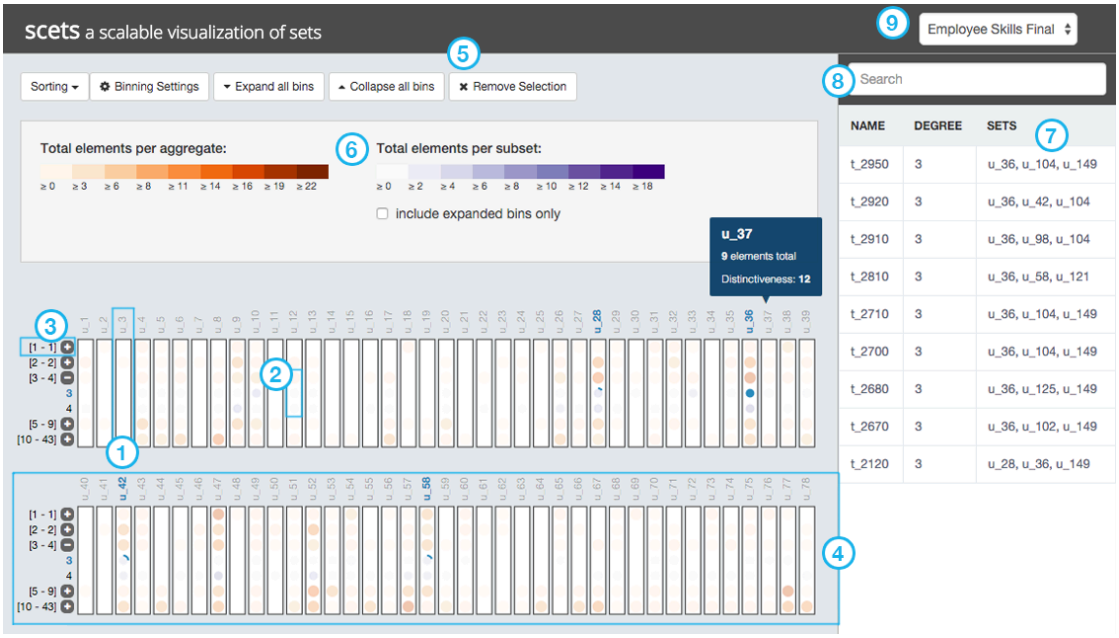
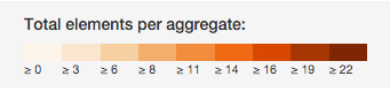
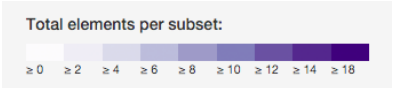


Figure 3.3: The implemented user interface and its main components: (1) a set with its name label, (2) aggregate [3-4] and subsets for degrees 3 and 4, (3) bin label and toggle button to expand/collapse the bin, (4) the second set group including sets from 'u_36' up to 'u_70', (5) the control panel, (6) the legend container including one legend for aggregates and one subsets, (7) the data table which shows detailed information about selected elements, (8) the search input field to perform dynamic queries, (9) the data navigator allows to switch back and forth between uploaded data sets

altering the color encoding of subsets depending on whether subset quantities of collapsed bins should be considered or not. This feature is described in detail in Section 3.2.8.



(a)



(b)

Figure 3.4: Legends support users in identifying the quantities for (a) aggregates and (b) subsets.

3.2 Interaction Design

Dix et al. [DE98] have given a very general definition of the term 'interaction' by describing it as 'The communication between user and the system'. According to Few [Few09] the effectiveness of information visualization depends on both, accurate visual representation and the ability of interaction. In general, interactive visualizations are more effective for solving complex tasks than static images. Fixed images are a good choice when there is no need for multiple views showing different perspectives on the same information.

Many dynamic visualizations have been developed under Ben Shneiderman's *Visual Information-Seeking Mantra*: 'overview first, zoom and filter, details on demand' [Shn96]. Applications which have been designed and developed in accordance with these guidelines, typically provide an overview of the data as well as the possibility to drill down into details. This way, users are given the control of changing the perspective and adjusting the visualization according to their needs. "An interactive visualization that offers an overview of the data alongside tools for 'drilling down' into the details may successfully fulfill many roles at once, addressing the different concerns of different audiences, from those new to the subject matter to those already deeply familiar with the data." [Mur13, p. 3]

Interaction is a user-centered process and as such, it is important to understand user behavior and user intents. Therefore Yi et al. [YKSJ07] have categorized interaction based on user intents and introduced the following seven categories of user interaction as a result of their study:

- *Select*: mark something as interesting
- *Explore*: show me something else
- *Reconfigure*: show me a different arrangement
- *Encode*: show me a different representation
- *Abstract/Elaborate*: show me more or less details
- *Filter*: show me something conditionally
- *Connect*: show me related items

Additionally, the category *Undo/Redo* is not only mentioned in the study, but is also relevant in the context of this thesis. Techniques falling within this category enable users to go to a pre-existing system state.

`Scets` applies several established interaction techniques in order to support users in exploring and analyzing data. In this section the main interaction concepts used in `Scets` are described in detail. At the end of this section the applied techniques are classified according to the seven interaction categories as proposed by Yi et al.

3.2.1 Selection possibilities

Selection techniques enable users to mark items of interest. These techniques are often referred to as 'Brushing'. Voigt [Voi02] defined brushing as "...selecting a subset of the data items with an input device (mouse)". This way, users can visually distinguish items of interest from other parts and keep track of selected items even in large data sets. A typical use case for brushing in `Scets` is the selection of aggregates and subsets in expanded bins. By clicking on an aggregate resp. subset in the visualization, this item is marked. Items which are not part of the selection will be displayed with reduced opacity. Thereby relevant parts of the data are highlighted and can be easily isolated. The 'Remove selection' button in the control panel (see Figure 3.6) enables users to undo a selection and to go back to the default state where no items are selected. Closely linked to selection techniques are so-called 'Connect' interactions which are mainly used to highlight relationships between data items. In `Scets` this technique is used to visualize the portion of elements appearing in other subsets of the same bin by showing a blue circular arc that surrounds the related subset. The combined use of selection techniques and connect interactions enables users to reveal patterns and associations in the data.

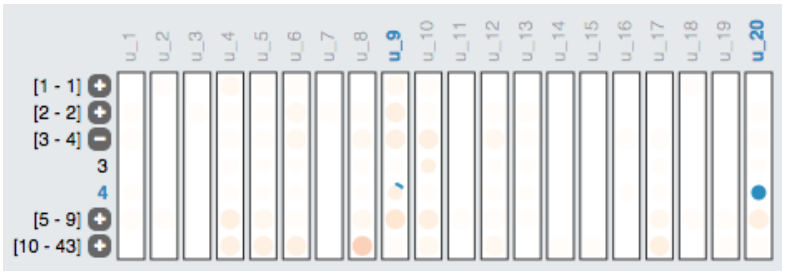
Consider the following example: The subset with degree 4 in set 'u_20' (referred to as A) contains one element, called 't_1350'. Table 3.3 displays all elements included in set 'u_20' along with a list of sets each of those elements is included in. It can be noted, that one of the listed sets in Table 3.3 is set 'u_9' (also referred to as B). Elements belonging to B are listed in Table 3.4. By comparing both tables, one can identify, that B contains six elements, out of which one (namely 't_1350') is also included in A. Thus, the intersection of A and B is defined as $A \cap B = \{t\_1350\}$. Assuming that the user has selected A, then the ratio of B which belongs to the selection is therefore equal to the formula $|A \cap B|/|A| = 1/6$. Hence, the colored circular arc framing 'u_9' is 1/6 of the subset's perimeter. By moving the mouse over B while A is still selected, the tooltip displays the percentage of the selected elements included in B (see Figure 3.7(d)).

name	**degree**	**sets**
t_1350	4	u_9, u_20, u_120, u_121

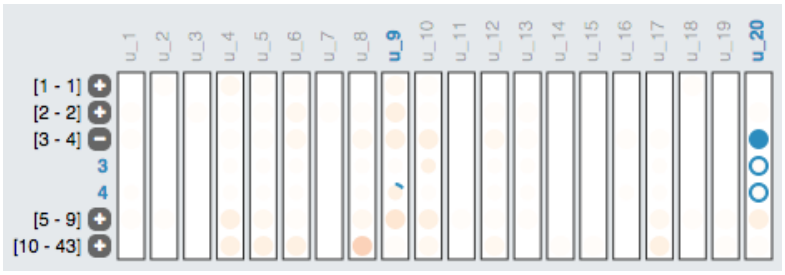
Table 3.3: List of elements included in subset u_20 of degree 4

name	degree	sets
t_1410	4	u_9, u_54, u_79, u_113
t_1350	4	u_9, u_20, u_120, u_121
t_1290	4	u_9, u_51, u_74, u_147
t_1270	4	u_9, u_38, u_74, u_138
t_1190	4	u_9, u_32, u_52, u_74
t_221	4	u_9, u_31, u_50, u_147

Table 3.4: List of elements included in subset u_9 of degree 4



(a)



(b)

Figure 3.5: Selection possibilities: (a) a selected subset of degree 4 in set 'u_52', (b) a selected aggregate ([3-4]) in set 'u_52'. Selected items are displayed as blue circles. Circular arcs depict the portion of elements shared between a particular subset and the selected item.

3.2.2 Expand and collapse bins

By providing the possibility to expand and collapse bins, the level of detail can be adjusted. Yi et al. categorize such techniques as 'Abstract/Elaborate' [YKSJ07] which give users the ability to change the level of abstraction. Techniques from this category are aligned to the 'Details on Demand' technique which is part of Ben Shneiderman's Visual Information Seeking Mantra. In `Scets` users are allowed to change the representation from a compact overview down to details on degree level. A little toggle button is placed next to each bin. Small icons inside the buttons indicate the current state of the corresponding bin. Each bin can be in one of the following two states:

- *collapsed*: The bin is currently in a collapsed state and the toggle button displays a (+) icon. Clicking the button will result in expanding the bin. As for the labels, only the bin label is displayed, showing start and end degree of the given bin. As for visualized items, only aggregates are displayed inside sets.
- *expanded*: The bin is currently expanded and the toggle button displays a (-) icon. A button click will trigger the collapsing of the bin. Below the bin label, each degree level of the expanded bin is labelled. Also, for each degree level a subset is displayed right below the related aggregate inside every set.

By default, all bins are in a collapsed state. As the user clicks on the toggle button the bin's state instantly changes. This is visualized by a smooth animation which shifts degree labels and subsets up (resp. down) depending on whether bins get expanded or collapsed. It is important to note, that not only the bin of a particular set group is expanded (resp. collapsed), but also bins across all set groups are affected by a state change caused through a click event. Figure 3.3 (2) shows an example of an expanded aggregate and the subsets included. Additionally, the buttons 'Expand all bins' and 'Collapse all bins' located in the control panel (see Figure 3.6) can be used to conveniently trigger a state change for all existing bins. This is especially useful when the user wants to jump back and forth between the abstracted overview showing only collapsed bins and the detail view where all bins are expanded.

3.2.3 Control Panel

As described in Section 3.1 the control panel is a major component of the user interface. It serves as a common container for several control elements (see Figure 3.6) which are part of interaction operations introduced in this section.

3.2.4 Tooltips

Tooltips are another typical technique from the category of 'Details on Demand' operations. As such, tooltips provide additional information without changing the view. Due to limited screen space it is often impossible to visualize every detail. Consequently, the visualization provides a basic overview of the data and enables users to request more

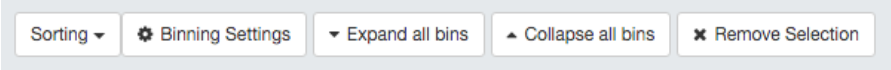
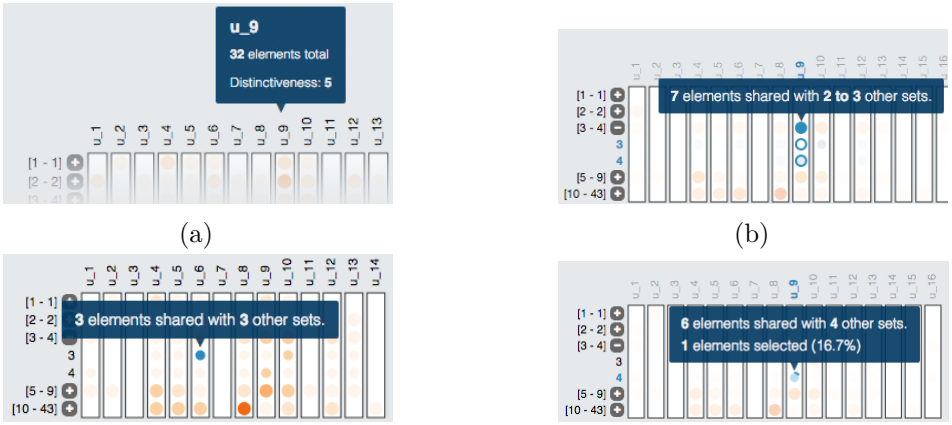


Figure 3.6: The control panel consists of five buttons (from left to right): change the sort methods for sets, open the binning settings view, expand all bins, collapse all bins, remove an active selection

details through interaction, such as a mouse-over. In `Scets` tooltips are frequently used, e.g., when the user moves the mouse cursor over an aggregate or a subset details about the component are brought up. This way additional information is displayed without changing the representational context. Figure 3.7 shows several components and the tooltips as a user hovers over the item. Once the user moves the cursor away from the object, the tooltip disappears immediately.



(a) (b) (c) (d)

Figure 3.7: Tooltips for several components: (a) hovering over a set label displays the set's name along with the number of elements included as well as its distinctiveness, (b) a tooltip gives indication about the total number of elements included in an aggregate, including all subsets that belong to this aggregate, (c) another tooltip that displays the number of elements included in a particular subset, (d) a tooltip for a selection that shows both the total amount and the ratio of selected elements

3.2.5 Filter

Filtering allows users to separate out irrelevant items. Yi et al. define 'Filtering' as a basic interaction technique that enables users "...to change the set of data items being presented based on some specific conditions. In this type of interaction, users specify a range or condition, so that only data items meeting those criteria are presented. Data items outside of the range or not satisfying the condition are hidden from the display or shown differently, but the actual data usually remain unchanged so that whenever users

reset the criteria, the hidden or differently shown data items can be recovered.” [YKSJ07]

`Scets` employs an interactive search through dynamic queries. This implies that queries in the form of user inputs continuously update the data which is visualized. By means of instantaneous visual updates, users receive direct feedback about how the system responds to their actions. Users can perform a search query by typing a desired term into the search bar located right above the data table. As the user types, the query is executed almost in real time and possible matches are highlighted in the matrix view as shown in Figure 3.8. This so-called ‘incremental search’ is a well-established user interaction pattern in computer science. In order to prevent the matrix view from constantly getting updated while the user is still typing, the update is performed with a delay of 500 milliseconds. Thereby not only possible flickering due to repeatedly updating the results can be avoided but also the computational overhead can be reduced by avoiding expensive operations.

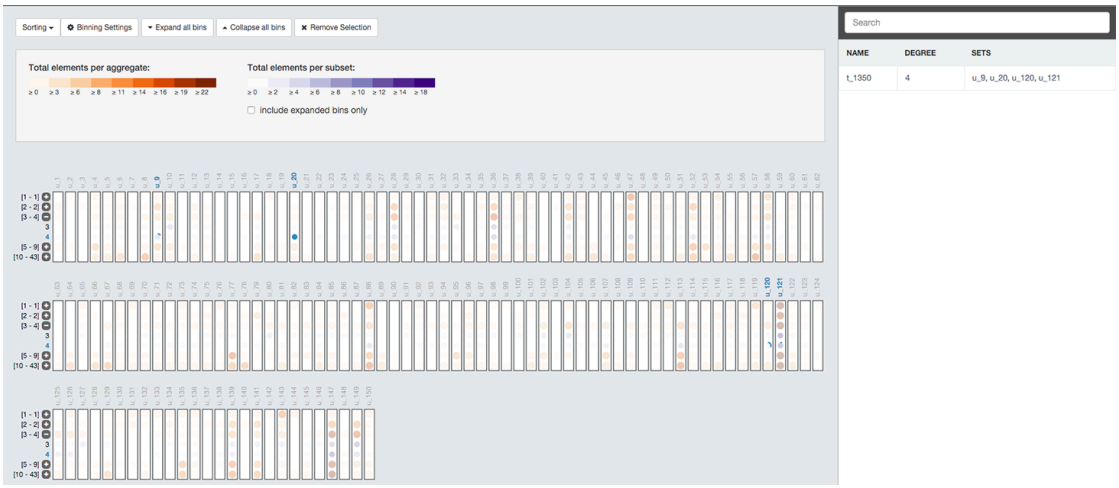


Figure 3.8: Interactive search: Items matching the search term ‘t_1350’ get instantly highlighted in the matrix view.

3.2.6 Binning View

As opposed to the binning algorithm described in Section 3.1.2, which determines the default bin settings, the ‘Binning View’ allows users to manually configure bins. By clicking on the button ‘Binning Settings’ located in the control panel (see Figure 3.6) a new modal window appears on top of the main view.

The binning view is separated into two sections. On the left-hand side a histogram of elements binned by their degree is displayed. Bins created by the binning algorithm are displayed on the right-hand side of the histogram. The purpose of the histogram is to

support users in finding answers to questions such as *What should be the total number of bins?* or *What are the individual bin ranges?* The possibility to change binning settings enables users to change the grouping of elements and provides a different perspective onto the data. Thus, this technique falls under the category of 'Reconfiguration'. [YKSJ07] Both the total number of bins and the range of each bin can be manually changed by the user through interaction. When the number of bins is changed, the range input fields for individual bins are updated automatically showing the computed bin ranges. Once the user confirms her changes by clicking the 'Save Changes' button, basic validation is performed to ensure that bins are not overlapping each other. As soon as validation is finished and no errors occur, the modal window disappears and the matrix view is re-rendered with the updated bins. Figure 3.10 shows how the visualization changes when the total number of bins is changed from five to three.

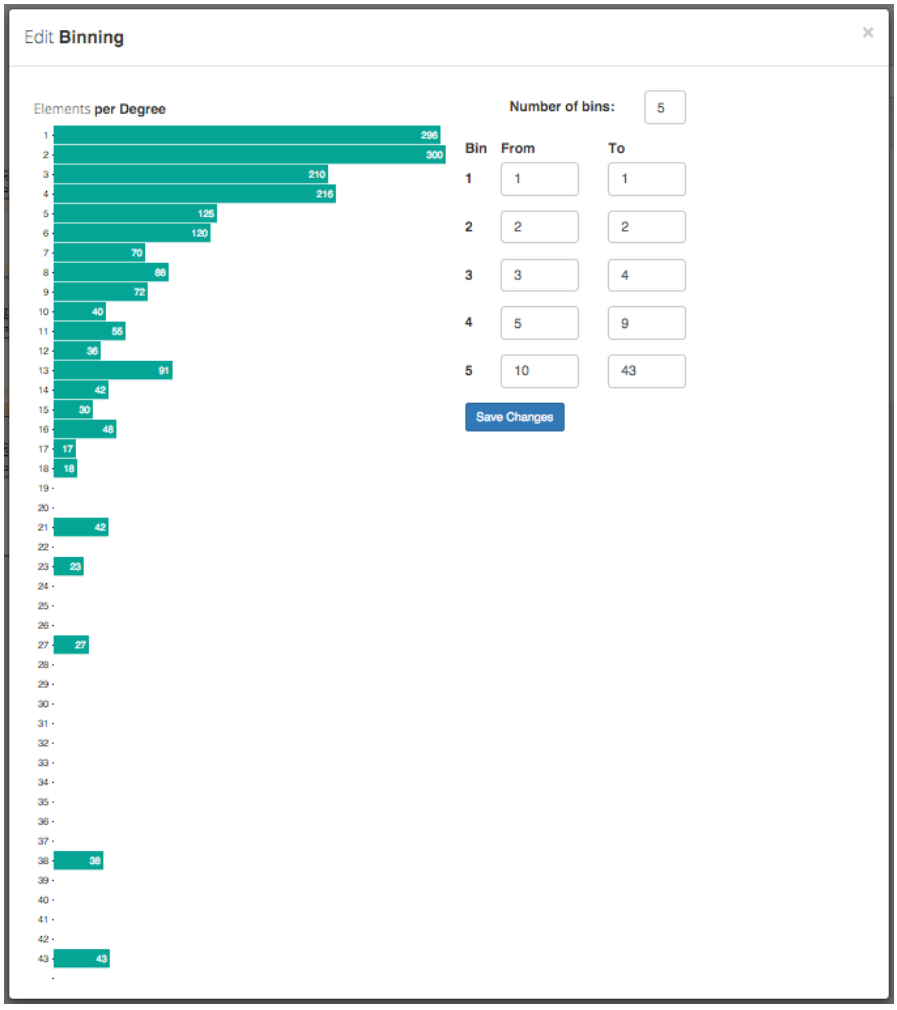
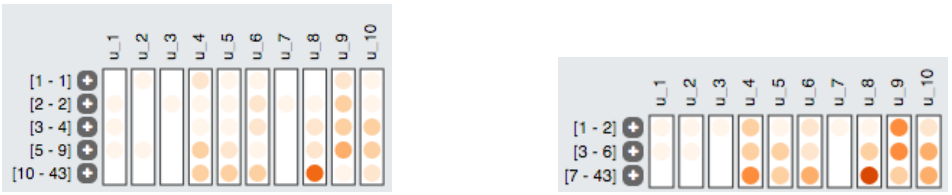


Figure 3.9: The Binning View: It is composed of a degree histogram and control elements to modify bins



(a) (b)

Figure 3.10: Employee data visualized with (a) five bins and with (b) three bins.

3.2.7 Sorting

By default sets are displayed in the order as they appear in the uploaded data set. In some cases users might want to rearrange sets based on various criteria to get a different perspective on the data. Therefore the order of sets can be changed by the user. Like changing the binning settings (see Section 3.2.6), sorting of sets can be categorized as 'Reconfiguration' [YKSJ07]. The sort dropdown menu is located in the control panel (see Figure 3.6) and enables users to switch between one of the following options:

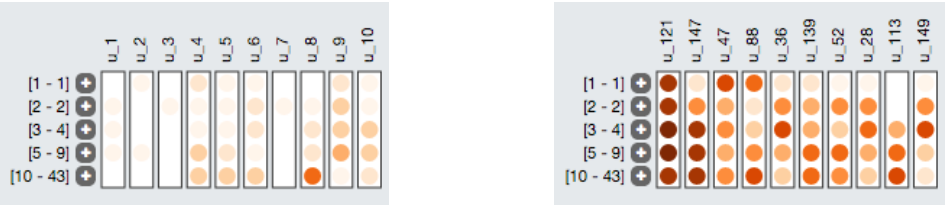
- *Default*: This is the default sort method, i.e., sets are sorted according to the order in the CSV file.
- *Name*: Sets are sorted alphabetically by name.
- *Quantity*: Sets are sorted by the total number of elements in descending order.
- *Distinctiveness*: Sets are sorted by their distinctiveness in descending order.

A set's distinctiveness is computed based on the average degree of the set elements. Algorithm 3.2 shows an implementation of the algorithm for one particular set in pseudocode. High distinctiveness indicates that a given set contains elements of low degree on average, i.e., elements which have few or no memberships with other sets. By contrast, a low distinctiveness value implies that the average element degree in this set is high.

Algorithm 3.2: Compute Average Set Degree Algorithm

```
1 sum ⟵ 0;
2 for i ← 0 to i < aggregates.length do
3 |   subsets ← aggregates[i];
5 |   for j ← 0 to j < subsets.length do
6 |   |   sum ← sum + subset.count ∗ subset.degree;
7 |   end
8 end
9 returnsum/set.size;
```

Once the sort method is changed by the user, the view is re-rendered and depicts sets in updated order. Figure 3.11 shows how the visualization changes when the sort method is switched from default sorting to sorting by quantity.



(a) (b)

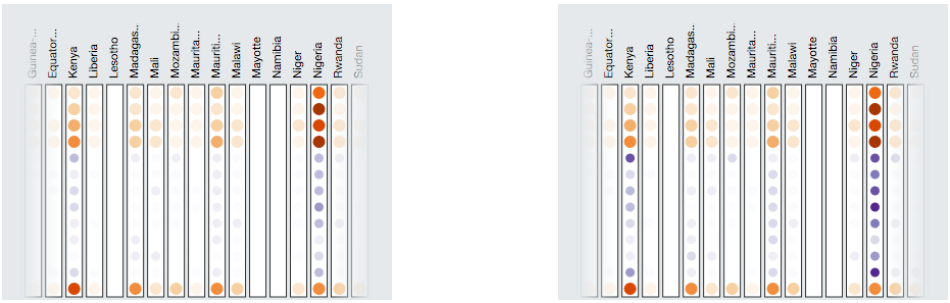
Figure 3.11: Ten sets from the employee data set sorted (a) by default and (b) by quantity

3.2.8 Subset Color Encoding

By enabling users to alter the color encoding of subsets, `Scets` offers an intuitive way of changing the visual representation of the data. Yi et al. [YKSJ07] classify this technique as 'Encode'. "Since color encoding is changed instantly and dynamically, users can experiment with various color encoding schemes to find the most suitable one." [YKSJ07] As already described earlier, `Scets` employs two different color scales to encode quantity: one for aggregates and one for subsets. By default, the subset legend includes the entire range of possible values (i.e., the total numbers of elements per subset). This implies that even quantities of subsets which are located in collapsed bins are included. Thus, a checkbox labelled 'include expanded bins only' is displayed right below the subset legend which enables users to alter the color encoding of subsets. By activating the checkbox the color scale will be computed based on subset quantities of expanded bins only. This is very useful as the information conveyed by color "...are directly related to how users understand relationships and distributions of the data items." [YKSJ07] In this case, it supports users in identifying the most populous subsets by considering expanded bins only rather than taking the entire range of possible subset quantities into account. Figure 3.12 illustrates the difference in the color encoding: With an unselected checkbox (Figure 3.12(a)) subsets in 'Nigeria' are not as prominent as if the checkbox was ticked (Figure 3.12(b)).

3.2.9 Classification of techniques

Table 3.5 shows a matrix of interaction techniques and categories introduced by Yi et al. Interaction techniques are represented as rows, categories are displayed as columns. A checkmark in a cell indicates that a given technique falls into a certain category. Two things are important to note: First, `Scets` makes use of all types of tasks in Yi et al.'s taxonomy. Second, as shown in the table, most techniques belong to multiple categories. The reason is, that most of the techniques used in `Scets` simply cannot be sorted into



(a) (b)

Figure 3.12: Color encoding of subsets: (a) all bins are considered, (b) only expanded bins are included

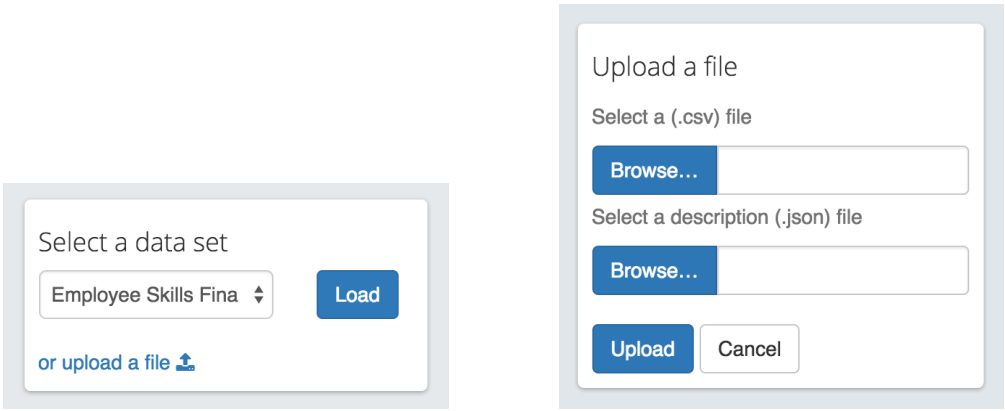
Techniques	Select	Explore	Reconfigure	Encode	Abstract/ Elaborate	Filter	Connect	Undo/Redo
	Categories							
Selection Possibilities	✓						✓	✓
Expand & Collapse Bins		✓			✓			✓
Tooltips					✓		✓	
Filter		✓				✓	✓	
Binning Settings			✓					
Sorting			✓					
Subset Color Encoding			✓	✓				

Table 3.5: Implemented techniques and their classification

a separate category as they usually apply operations from several interaction categories. As an example, selecting subsets resp. aggregates in `Scets` falls on one hand under the 'Select' category but also under the 'Connect' category since relationships between data items are revealed. On the other hand removing a selection using the related button in the control panel is a typical example for an operation from the 'Undo/Redo' category.

3.3 Selecting and uploading data samples

When starting the application, first of all a dialog is shown providing two options: Users can either pick a sample file from a list of existing data sets located on the server or upload a new data set. The component that acts as the application's entry point and provides the interface between user and server is called the 'Data Navigator'.



(a) (b)

Figure 3.13: The 'Data Navigator' component enables users to either (a) select a sample file or (b) to upload a new file.

In the default view a list of sample files (see Figure 3.13(a)) is displayed. This list includes both predefined test data as well as uploaded files by the user. By clicking on the link *'or upload a file'* the upload window appears. The following form requires users to select two files from the local file system: A data file in CSV format and a description file in JSON format. The description file contains meta information about the related data set, such as the file's author and title, the separator (can be any character, e.g., comma or space), etc. An additional 'set' property is used to define a lower and upper bound and thereby limit the range of the binary data. This allows to exactly specify which parts of the data file have to be visualized and which parts are additional information.

skill	**u_1**	**u_2**	**u_3**	**u_4**	**u_5**	**u_6**	**u_7**	**u_8**	**u_9**	**u_10**
t_1	1	0	0	1	0	1	0	0	0	1
t_2	1	1	0	0	0	1	1	1	1	1
t_3	1	1	1	0	1	0	1	0	0	1
t_4	1	1	0	1	1	0	1	1	1	1
t_5	1	1	1	0	0	1	1	1	1	0
t_6	0	0	1	1	1	1	1	1	0	1
t_7	0	1	0	0	0	0	0	1	0	0
t_8	1	0	1	1	1	1	1	1	0	1
t_9	1	0	0	1	0	0	0	1	1	1

Table 3.6: A CSV sample in correct format. Employees (u_1, u_2, ..., u_10) are encoded as columns. Skills (t_1, t_2, ..., t_9) are represented as rows.

Currently, an uploaded file has to strictly follow a given convention: Sets have to be

represented as columns, whereas elements resp. attributes need to be indicated as rows. Table 3.6 shows a sample CSV file. Files formatted in a different way will lead to an error during the upload process since the parser only supports the named convention.

3.4 Task Support

To conclude this chapter, I will briefly describe how `Scets` can be used to support users in solving the tasks specified in Section 3.1.

Users can analyze the distribution of elements per set according to their degrees (task **T1**) by exploring expanded bins in the matrix view. A subset's color gives indication about its quantity. Interaction is required to determine the exact number of elements per subset and the number of sets they belong to. Hovering over the subset displays the information in a tooltip, clicking on the subset shows the included elements along with a list of sets they belong to in the element view.

Task 2 (**T2**) can be achieved by investigating a set's first resp. last degree in the matrix view. Tooltips display the number of elements shared with X-1 other elements as the user moves the mouse over a subset of degree X.

Sets can be compared by distinctiveness in the matrix view (task **T3**). Therefore sets can be sorted by distinctiveness via the sort menu. This way the most competitive country will be in first place and the least competitive country will be in last place. Hovering over a set's name label, a tooltip is displayed showing it's distinctiveness along with additional information. Moreover, a set's color gradient gives indication about its distinctiveness, i.e., a gradient from dark to light suggests high distinctiveness, whereas light to dark indicates low distinctiveness.

By selecting an aggregate or subset in the matrix view, all elements included in the selection are listed in the element view. In addition, the portion of elements appearing in other sets is displayed in the matrix view as blue arcs. This way users can analyze which elements of the corresponding set are shared between other sets (task **T4**).

Finding all sets which contain a specific element (task **T5**) can be achieved by using the search input. As users type an element's name, the matrix view is updated immediately and all subsets which contain the given element are highlighted. Furthermore, the element view lists all elements which match the search criteria.

CHAPTER 4

Implementation

4.1 Technology Fundamentals

The combination of Javascript, CSS and SVG establishes a very powerful toolset for prototype development. Javascript is one of today's most popular languages that runs on every modern web browser. Hence Javascript applications typically don't require a complicated setup process to be executed and therefore can be accessed by a large number of users. This section gives a brief introduction to the main concepts and technologies that are used to build a web-based prototype.

4.1.1 HTML

The Hypertext Markup Language (HTML) is the most relevant markup language for describing the structure and semantic of web pages. HTML is a set of elements each of which describes different document content. It enables the specification of a semantic structure and gives meaning to the content. However, it does not represent the document's structure visually. Therefore CSS is being used (see Chapter 4.1.2). Most HTML elements consist of an opening and an end tag with arbitrary content in between. Element attributes provide additional information about an element and are defined as key/value pairs in the opening tag. Nested elements can be used to create hierarchies within the document. Figure 4.1 shows a sample HTML document.

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta charset="UTF-8">
    <title>A simple example</title>
</head>
<body>
    <h1>Our big fat headline</h1>
    <p>Just a simple paragraph...</p>
    <div class="container">
        Here goes even more stuff.
    </div>
</body>
</html>
```

Figure 4.1: A very simple HTML document

4.1.2 CSS

Cascading Style Sheets (CSS) is a stylesheet language which represents the document structure visually. It describes how HTML elements have to be displayed on the screen. A CSS rule is made up of a selector and a declaration block (see Figure 4.2).

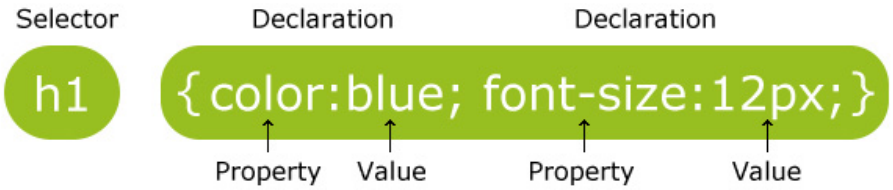


Figure 4.2: A CSS rule consists of a selector and a declaration block.

Source: `http://www.w3schools.com/css/css_syntax.asp`

The selector refers to either a single HTML element or a set of elements on which to operate on. One or more declarations constitute the declaration block. Each declaration is given by a property name and a value separated by a colon. There are different ways to apply CSS rules to a HTML document:

- Attaching inline styles
- Embedding CSS in the HTML document by using the *<style>* tag
- Referencing an external CSS stylesheet from the HTML document with the *<link>* tag

```
<svg width="50" height="50">
    <circle cx="25" cy="25" r="22"
     fill="blue" stroke="gray" stroke-width="2"/>
</svg>
```

Figure 4.3: A blue SVG circle element with a 2 pixel gray border

By creating an external file that contains the style definitions, one can easily change the look of a document by just replacing that file. Following this approach, one also achieves separation of concerns, which is a popular design principle in computer science. This implies the separation of the document structure, i.e., the HTML code, from the definition of content presentation style, i.e., CSS code.

4.1.3 SVG

Scalable Vector Graphics (SVG) is a text-based image format that uses the XML (Extensible Markup Language) format to define graphics. Thus, all elements must have a closing tag. SVG is mainly used to define two-dimensional vector-based graphics for the Web. Compared to a bitmap, a vector-based image does not consist of single pixels but rather of objects that can be modified in size without any loss of quality. This characteristic makes vector graphics so special. SVG code can be included directly in any HTML document or it can be dynamically created and injected into the Document Object Model (DOM) by using Javascript. SVG provides some pre-defined shapes, such as rectangles, circles, polygons, lines, etc. An example of a SVG circle element can be seen in Figure 4.3.

4.1.4 Javascript & d3.js

JavaScript is a cross-platform, object-oriented scripting language and is often referred to as the “language of the web”. Whereas HTML and CSS are used to define the content resp. specify the layout of HTML documents, Javascript enables the implementation of interactive web applications. This section will neither be an introduction into Javascript nor describe language-specific features as this would go beyond the scope of the thesis. However, as the prototype is written in Javascript it is worth mentioning the basic characteristics of the language at this point. Javascript enables user interaction and is used in many fields of application, such as dynamic manipulation of the DOM or client-side form validation just to name a few. Similar to CSS, JavaScript can be included in a HTML document in different ways: Either include a script directly in HTML between the *<script>* tag (see Figure 4.4) or reference an external script from the *<head>* of a document.

D3 (Data Driven Documents) [BOH11] is a Javascript library for creating web-based data visualizations. It was created by Mike Bostock, Vadim Ogievetsky and Jeff Heer in

```
<body>
    <script type="text/javascript">
        alert("Hello, world!");
    </script>
</body>
```

Figure 4.4: A Javascript code snippet included directly in HTML

2011 as the successor of the Protovis [BH09] library. D3 heavily relies on the three core technologies HTML, CSS and SVG which were introduced earlier in this chapter.

> "A huge benefit of how D3 exposes the designer directly to the web page is that the existing technology in the browser can be leveraged without having to create a whole new plotting language. This appears both when selecting elements, which is performed using CSS selectors (users of JQuery will find many of the idioms underlying D3 very familiar), and when styling elements, which is performed using normal CSS. This allows the designer to use the existing tools that have been developed for web design – most notably Firefox's Firebug and Chrome's Developer Tools." [Dew12, p. 2]

The D3 library allows users to bind data to the any kind of DOM elements and apply transformation to the document based on the data. In detail, d3 performs the following four steps: [Mur13, p. 7]

1. Loading data from a source (e.g., from a CSV or JSON file) into the browser
2. Binding data to elements, by creating new ones and updating or removing existing ones
3. Transforming these elements depending on the bound data
4. Performing smooth transitions on elements

D3 covers a wide range of usage scenarios, starting from creating static data tables up to building very sophisticated interactive visualizations. For example, D3 can be used to load a large dataset and create an animated bar chart where each bar is represented as an SVG rectangle and a bar's height being scaled to map from the corresponding value in the data set into pixels. The most important feature is reusability in the sense that D3 is not a typical charting library that comes with pre-build charts but rather provides the necessary tools for creating almost any kind of desired visualization. However, in some cases there is no need for the specificity offered by D3 and other libraries resp. frameworks prove to be more useful. Many alternatives to D3 exist, including Google Charts, Highcharts, Processing.js and Raphaël for example.

4.1.5 Node.js

> "Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices." [nod]

In simple terms, Node.js enables developers to build scalable network applications written in Javascript on the server-side. It was created in 2009 by Ryan Dahl as an Open Source command line tool. One of Node's key features is its event-loop via Javascript callbacks that is used to implement the non-blocking I/O. Typically Node.js is particularly suitable for creating streaming based real-time services, chat applications and in situations where client-side code should also be reused on the server-side and vice versa. In the context of this thesis Node.js serves as the main backend technology for implementing a HTTP server.

4.2 Project Structure

`Scets` is a web application written entirely in Javascript. The source code can be found on the Github repository `https://github.com/mawo87/scets/`. There is also a web demo of the most recent version which is available under `http://scets.sybdev.com/index.html`. This is simplest and most comfortable way to start the application as it requires no local setup.

In general, the application is composed of the following two main components:

- **Backend** which serves existing sample files, parses uploaded data and delivers parsed data as JSON (Javascript Object Notation) to the client.
- **Frontend** that visualizes the parsed data in the web browser and enables user interaction.

The project is structured as follows:

/: The project root including configuration files and the *index.html*

/backend: Contains the server.js file which implements the Node.js server and a subfolder for backend modules

/data: All example files along with their description files are located here

/dist: The distribution folder which contains the Javascript files and the compiled CSS files. This folder should be copied to the local web server.

/doc: The documentation folder which contains documentation files generated by JSDoc and apiDoc.

/js: The Javascript source files which will be copied to the distribution folder later during the build process.

/node_modules: A place for external libraries

/sass: All SASS files are located under this directory.

4.3 Setup & Build Process

`Scets` can also be installed locally. This requires a few simple steps which will be described in the following. Several dependencies are installed as packages via npm, the Node Package Manager [npm]. Therefore Node.js is required which can easily be installed by following the instructions on `https://nodejs.org/`. Once Node.js is installed, the best way to install all the dependencies, is by running the command *npm install* from the project's root directory. This way all the required packages will be installed in the *node_modules* folder.
To get the application up and running locally one has to setup two simple HTTP servers. The first server will run the application's backend code written in Node.js. The second server will be a static file server that serves up all of the necessary resources to the browser, such as HTML, CSS and Javascript files. For this purpose any web server, e.g. Apache, can be used. Once setup is completed, the *dist* folder along with the *index.html* file has to be copied from the repository to the root directory of the static web server. The Node.js server can be started by running *node server.js* from the *backend* directory. Additionally, one can edit the *config.js* located under *backend/modules* in order to make setting adjustments such as changing the server port of the Node.js server or setting the address of the local web server. By entering the address of the just set up server (default *http://localhost:8005/*) in the browser, the application should start up.
Another possibility of setting up a local web server is by using gulp.js[gul]. Gulp is a build system built on Node.js that can be used to automate certain development tasks, such as compiling SASS to CSS, minifying files, etc. Bringing gulp into use not only allows a straightforward setup process of a web server, but also facilitates the development process a lot. The project comes with a *gulpfile.js* where several tasks are already defined and can be used by running **gulp TASKNAME** from within the project folder where TASKNAME has to be replaced by one of the following tasks:

- **connect**: Starts a webserver on port 8005. It also supports live reload, i.e., file changes will be reflected in the browser immediately.
- **sass**: Compiles SASS to CSS files and copies the output to a distribution folder
- **js**: Copies Javascript files from a source folder to a destination folder

- **default**: Runs the tasks 'connect', 'watch', 'sass' and 'js' in sequence
- **jsdoc**: Runs JSDoc from the command line and generates the documentation for the frontend code. Configuration options are specified in *jsdoc* located in the projects root directory.
- **apidoc**: Generates the API documentation. The configuration file *apidoc* in the projects root directory includes several configuration options.

4.4 Server & API

As already noted earlier in this chapter, the parsing and data processing is done on the application's server-side whereas the frontend component is mainly responsible for visualizing the data and for providing an interface that allows users to interact with the visualization. The server implements the following API calls:

GET /examples: Requests a list of existing samples, including both predefined and uploaded samples. An array of files is returned.

GET /example?file=filePath: Requests a particular sample file given by the *file* parameter. The server responds with the parsed file content.

POST /upload: Uploads a sample file together with its description file to the server. The file path is returned.

The API is also deployed to Heroko [her], which is a platform as a service (PaaS) that enables developers to build and run their applications in the cloud. In this case Heroku is configured in such a way that deployments happen automatically whenever something changes in the git repository. Furthermore, the API is running on `https://scets.herokuapp.com/api/` and as such it is publicly available.
Detailed documentation of the API is available under *doc/backend*. The tool *ApiDoc* [api] is used to generate the documentation from annotations in the backend source code.

4.5 Frontend

4.5.1 The Module Pattern

`Scets` makes use of Javascript's module pattern which is a commonly used design pattern that helps avoiding the pollution of the global namespace and supports readability and maintainability of the application's code. As a result, a new namespace called *scets* is defined on the window object. Both, classes, methods and shared data are added to this namespace. Figure 4.5 shows an example of how modules are organized in `Scets`. In this example 'methodC' stays private whereas the 'methodA' and 'methodB' will be exposed as public methods and hence will be available within the *scets* namespace under 'scets.myPublicMethodA' resp. 'scets.myPublicMethodB'. By adding new methods to the application, the module pattern should be used in order to avoid collisions with other objects or variables in the global namespace. This is especially important when dealing with third party libraries.

```
var scats = (function(vis) {

  //private properties
  var tmp = "some string",
      someBoolean = true;

  //private methods
  function methodA () {

  }

  function methodB () {

  }

  function methodC () {

  }

  //expose public methods
  return $.extend(vis, {
    myPublicMethodA: methodA
    myPublicMethodB: methodB
  });

})(scats || {});
```

Figure 4.5: An example of a module in `Scets` with two private properties, one private method and two public methods.

4.5.2 Components

The UML class diagram in Figure 4.6 shows all existing classes of the frontend and lists their attributes and methods. In addition to these classes a utility module exists that implements helper methods used by several other components. In this section the inter-

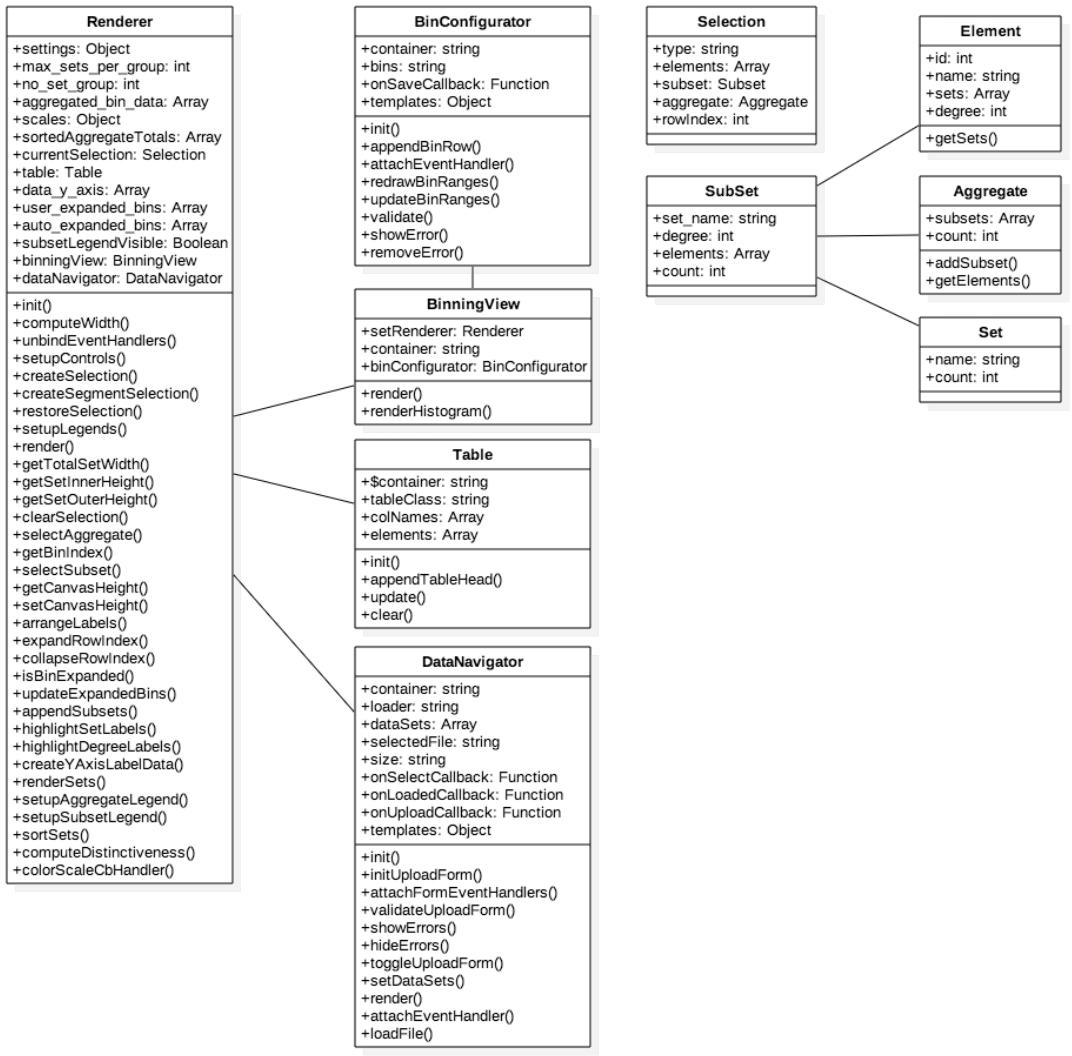


Figure 4.6: UML diagram showing the main components of the `Scets`' frontend

action of components depicted in the UML diagram will be described briefly.

There is only one HTML file called 'index.html' which includes all relevant Javascript and CSS files. The file app.js is the application's entry point. On the one hand it sets up the data object on the *scets* namespace. On the other hand it waits until the document has finished loading (i.e., the DOM is fully loaded) and then creates a new instance of the **Data Navigator** which initially requests a list of available sample files from the `Scets` API and then renders the result in a dropdown list. Now either a sample file can be selected or a new file can be uploaded. In any case the server parses the CSV

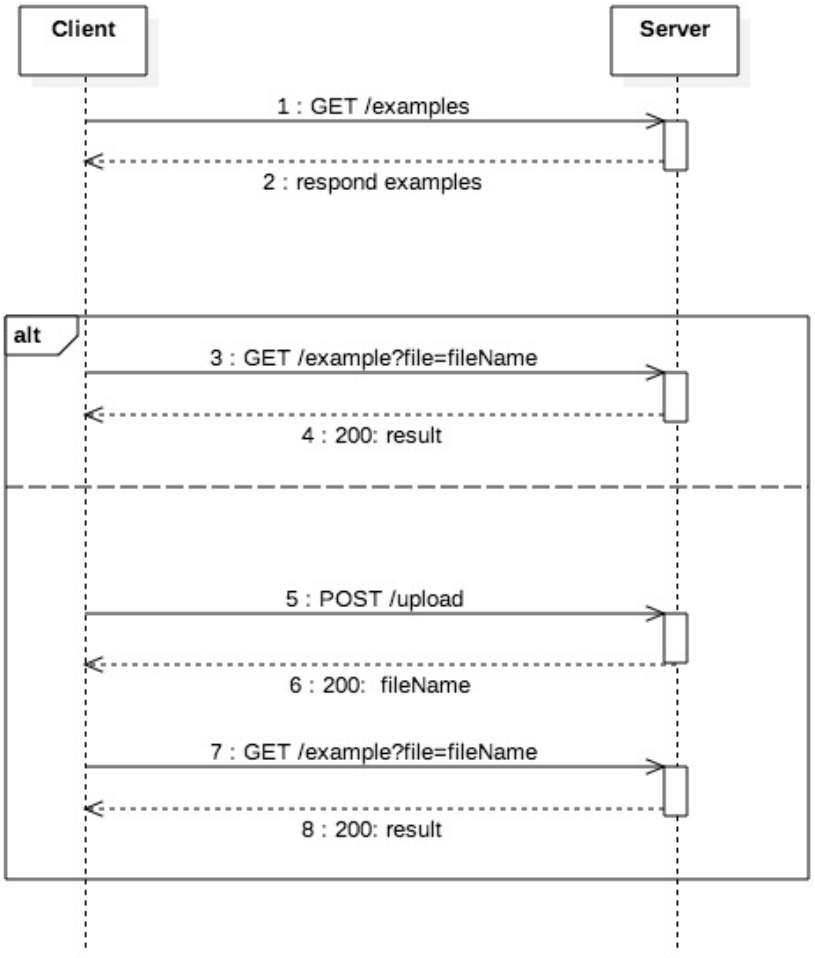


Figure 4.7: The sequence diagram of the client/server communication

file which is either a file selected from the dropdown list or an uploaded file and then returns the parsed file in JSON format. (see Figure 4.7)

On the client-side the Data Navigator's *onLoadedCallback* method gets executed. At this point the server's response gets processed and all custom data objects (i.e., **Sets** and **Elements**) will be created and added to the global namespace. Additionally the default bins get initialized with the help of the utility methods *initBins* and *classifyBinData*. From this point sets, elements and bins are accessible from within application's namespace.

The **Renderer** is the most essential class as it connects all other components. As soon as the client receives the response from the API a new instance of the Renderer

is created. First, several initialization tasks are performed such as setting up the UI controls and the legend. At the same time the aggregated data per bin is also generated. Next the *render* method is called. This the place where the data driven document is created with the help of d3. At first the aggregated data hast to be divided into smaller chunks each of which belongs to one set group. For each set group a SVG group element is created and the related data chunk gets bound via d3's *data()* method. Subsequently sets and aggregates are created and event handlers are attached. Finally labels for both sets and bins are created for every rendered set group.

Furthermore the Renderer creates instances for the **Binning View** and the **Table**. The 'Binning View' opens in a modal window when the user clicks on the button 'Binning Settings'. At this point the **Bin Configurator** gets initialized with the bins from the global data object. Whenever the user modifies the number of bins through the UI, the *initBins* method is called once again and the binning algorithm computes the bins and their data.

Once the user clicks on an aggregate or a subset in the UI, a new **Selection** is created with the Renderer's *createSelection* method. The Selection instance holds an array of selected elements and the selection type, which can be either 'subset', 'aggregate' or 'search'. If an aggregate is selected, the aggregate as well as its row index will be saved additionally. If a subset is selected, the subset will be set on the Selection instance. The Selection is primarily used as a global storage which can be referenced from anywhere else in the source code. Besides creating a new Selection instance, the *createSelection* also takes care of updating the content displayed in the 'Table'. The Table shows information about the current selection, e.g, the name and degree of elements included in the selection. This information gets updated automatically by the Table's *update* method when the selection is changed or removed.

Further documentation of frontend components is created with JSDoc [jsd] and is located in the *doc/frontend* folder on Github. This folder contains a separate HTML file for each class and describes its methods in detail.

4.5.3 Third Party Libraries

Several third party libraries are used to facilitate the development of the web application. As already noted in Section 4.3 all dependencies are located in the *node_modules* folder after being installed via npm. In the following a comprehensive list of libraries in place is given.

- **Bootstrap** [boo]: Bootstrap is a responsive CSS and Javascript framework. It simplifies the implementation of a grid layout and helps to arrange UI elements.
- **colorbrewer** [col]: This small library provides predefined color schemes based on `http://colorbrewer2.org`.
- **csv-parse** [csv]: A parser for converting CSV text input into arrays or objects
- **d3.js**: d3.js provides the basis for the visualization. More information about the library is provided in Section 4.1.4.
- **d3-tip** [d3t]: A very lightweight tooltip library built on d3.js
- **Express** [exp]: A framework for building web applications in Node.js
- **fontawesome** [fon]: A popular toolkit for scalable vector icons
- **JQuery** [jqua]: JQuery is mainly used for DOM manipulation and for selecting elements.
- **JQuery Form** [jqub]: A JQuery plugin that allows to perform AJAX requests on submitting HTML forms. It is used to build the upload form and avoid a page refresh.
- **jsdoc** [jsd]: An API documentation generator for Javascript
- **multer** [mul]: A Node.js middleware that handles file uploading
- **Underscore.js** [und]: A utility library that provides a lot of useful helper functions
- **Velocity.js** [vel]: A JQuery plugin for high performance animations

In addition, several gulp plugins are installed as modules which are mainly used for the build process and for running a local web server.

CHAPTER 5

Use Cases

"The best data visualizations are ones that expose something new about the underlying patterns and relationships contained within the data. Understanding those relationships - and so being able to observe them is key to good decision-making" [ACD12, p. 65]

In this chapter, two case studies are described which evaluate the effectiveness of the `Scets` technique when employed on large data sets. These case studies use data sets from two different domains and illustrate how the novel visualization technique supports data exploration and can be used to gain in-depth insights. In the first case study a movie data set is explored. In the second, exports from several countries are examined.

5.1 Movie Genres

The dataset in this use case represents a demo sample from the MovieLens website collected by GroupLens Research [Gro]. It comprises various multi-value (set-typed) attributes such as name, release date, genre and rating. These attributes describe set-element memberships, such as movies per genre. Overall, the dataset comprises 18 genres (sets) and 3883 movies (elements). By employing the `Scets` technique to visualize the data, some interesting findings were gained.

As an example, by looking at the bin ranges (in fact the range of the last bin) one will discover that the maximum degree is five. Given that and the default number of bins (which is also five), each bin includes exactly one subset. This implies that there are no movies which belong to more than five genres. Also, as the number of default bins coincidentally equals to the maximum degree, there is no need to expand bins in this use case since the information content of aggregates and subsets will be the same.
In general, across all genres only a small number of movies belongs to five genres. This

can be determined by comparing an aggregate's color with the color scale for aggregates. Hovering over the last aggregate circle in each set gives information about the number of elements comprised. By comparing these numbers one will find out that among all genres the 'Action' genre includes most movies (seven in total) with degree five. These insights are displayed in Figure 5.1.

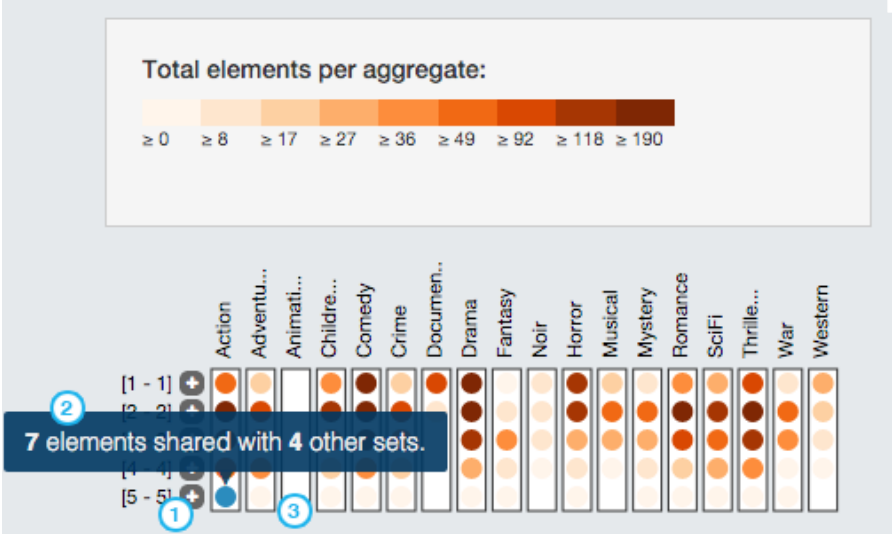


Figure 5.1: Visualization of 18 movie genres with 3883 movies. (1) The upper range of the last bin indicates the maximum degree. (2) 7 'Action' movies belong to four other genres (five genres in total including 'Action' itself). (3) There are no 'Animation' movies.

In general, tooltips can expose a lot of useful information. For example, by moving the mouse over various genres, they can be compared with regard to their total number of movies. Thereby the genre having the most movies can be identified: As shown in Figure 5.2, 'Drama' includes 1603 movies in total of which 843 elements belong to the subset of degree 1, i.e., they don't appear in any other genre.

Another interesting finding which does not require any user interaction and can be immediately discovered by looking at the visualized data, is that the 'Animation' genre does not include any movies. This genre is displayed as an empty set (see Figure 5.1 (3)), i.e., it does neither contain aggregates nor subsets.

The case study also revealed interesting data characteristics which were not visible in the raw data. Documentaries rarely belong to another genre and if they do, then at most to one other genre. This characteristics is reflected in the visualization in form of empty space in the set as shown in Figure 5.3 (1). By investigating the 'Documentary' genre, one will find two aggregate circles, one for the first bin, and one for the second bin. The topmost circle is darkened and depicts movies which are exclusive to the 'Documentary' genre. The aggregate below represents elements which belong to exactly one other

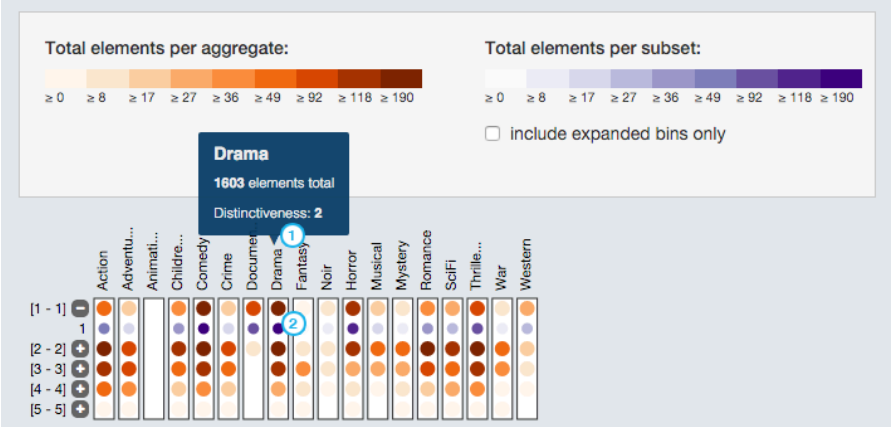


Figure 5.2: (1) The 'Drama' genre includes with 1603 elements the most movies. (2) The subset for degree 1 represents 843 elements which is about 50% of the total number of elements in this genre.

genre. Compared to the color of the first aggregate, this circle is lighter, meaning that the aggregate contains less elements than the first aggregate. There are no more colored circles from the second aggregate onwards leaving some blank space. Thus, it can be concluded that 'Documentary' movies don't belong to more than one other genre.

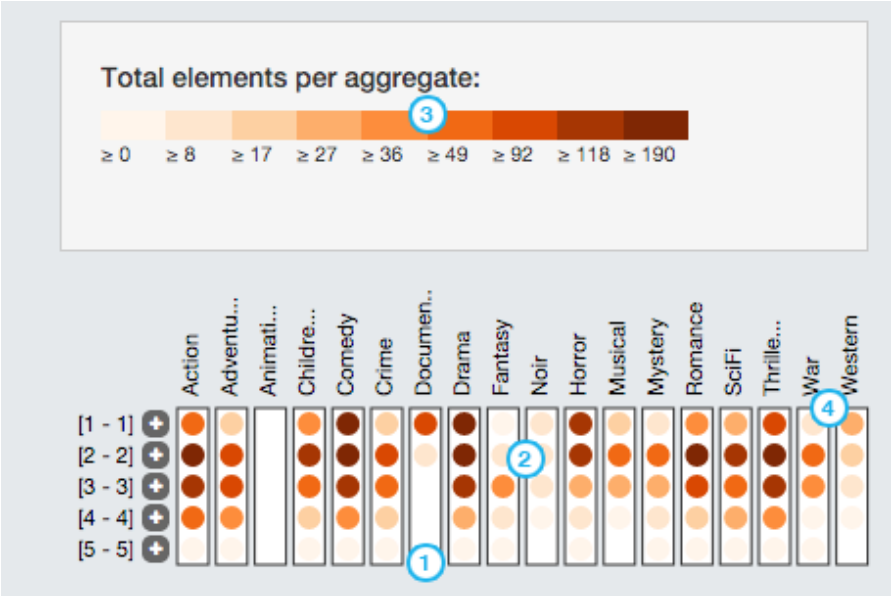


Figure 5.3: (1) 'Documentaries' belong at most to one other genre. (2) 'Fantasy' movies typically belong to two additional genres. (3) The legend helps to distinguish aggregates in terms of the total number of elements. (4) 'War' movies mostly belong to one or two other genres.

Colored aggregates along with the legend support users in identifying bins which contain either a very large number of elements or little elements. For instance, the 'Fantasy' genre mostly comes as addition to two other genres, and rarely alone. This is because aggregate circles in the 'Fantasy' set are mainly brightly colored except for the third aggregate from top (see Figure 5.3 (2)). This one is much darker and indicates a larger number of elements included. By comparing the aggregate's color with the legend one will notice that the number of comprised elements is somewhere between 36 and 49 (see Figure 5.3 (3)).
In this context, it may be also noted that the 'War' genre rarely comes alone. Figure 5.3 (4) shows that 'War' movies mostly appear in one or two other genres.

Furthermore, sorting sets by quantity does not only confirm the previous finding that the 'Drama' set includes the most elements but also highlights another interesting characteristics: The genres 'Drama', 'Comedy', 'Action' and 'Thriller' are likely to come alone and along with one or two other genres. This is illustrated in Figure 5.4 (1) and (2).

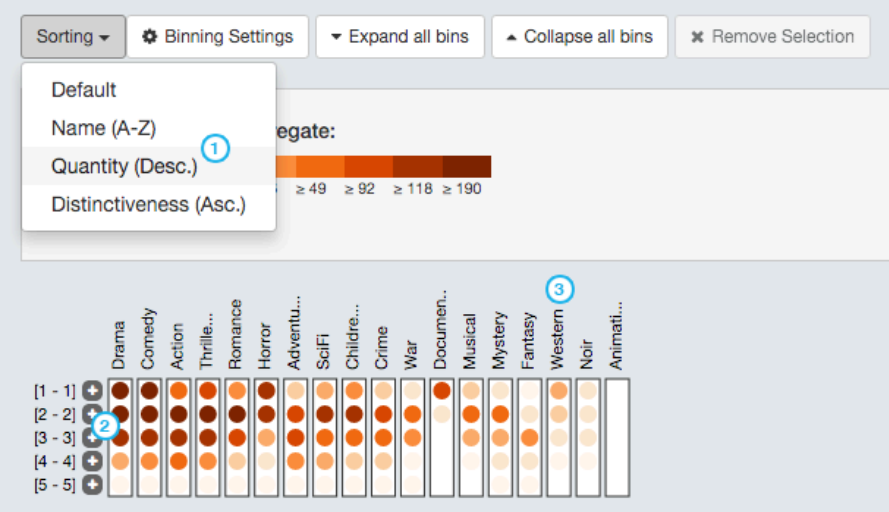


Figure 5.4: (1) Sorting sets by quantity reveals unknown patterns, such as that genres like 'Drama', 'Comedy', 'Action' and 'Thriller' mostly appear with one or two other genres (2). Aggregate's color decreases in intensity in the 'Western' genre from bin to bin (3).

It is also noticeable, that the 'Western' genre has decreasing numbers of elements per bin, i.e., most elements are placed in the first bin whereas the fewest elements are contained in the last bin. Figure 5.4 (3) shows how the the color intensity decreases from the aggregate representing the first bin to the one depicting the last bin.

5.2 Countries Exports

In this use case economic data is investigated by visualizing international trade statistics on exports and imports by countries from the year 2010 [Com]. The dataset comprises 194 countries (sets) and 1354 different export goods (elements). By comparing these numbers with the use case from Section 5.1 one will notice that now about ten times more sets are used. However, the total number of elements in this dataset corresponds to approximately one-third of the total number of elements from the first use case. This difference is also reflected in the visualization: Sets don't fit into one single row anymore. As a result set groups are automatically generated (see Figure 5.5).

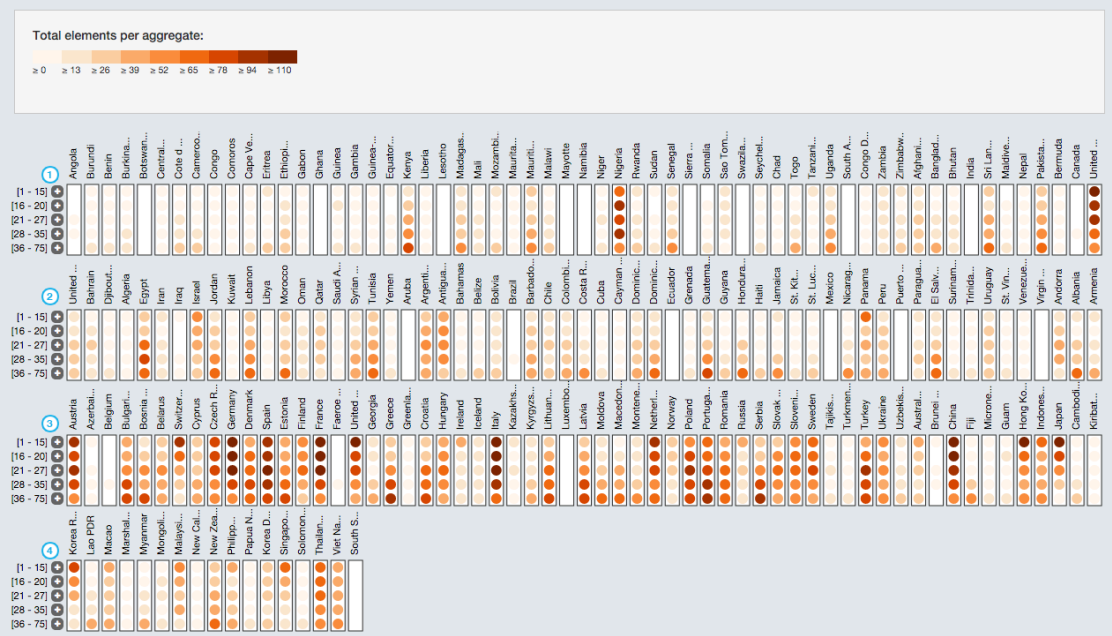


Figure 5.5: 194 countries visualized in 4 set groups (1) - (4).

Since the original dataset did not fulfill the criteria for the supported file structure mentioned in Chapter 3.3, it had to be preprocessed beforehand. In this step, rows and columns were transposed because sets wer represented as rows and elements as columns in the original file. Furthermore, a uniform separator had to be defined and invalid separators were recursively replaced throughout the entire file. Whereas most of these tasks can be automated with the help of certain algorithms, in some cases manual adjustment is still necessary.

One of the first findings users can gain about the data is the quantity of exports per country. The visualization reveals at a glance how many elements a set comprises compared to others based on the color encoding. In this particular case, Germany, Spain and Italy for example export a larger number of elements in contrast to Congo or Comoros.

It also shows that there are 18 countries which don't export any items at all, i.e., they are shown as empty sets. These include Botswana, Ghana, Lesotho, Mayotte, Namibia, Sierra Leone, Swaziland, South Africa, India, Aruba, Ecuador, Mexico, Puerto Rico, Virgin Islands U.S., Luxembourg and South Sudan. The result is shown in Figure 5.6 (3). In this context, two other findings can be gained, namely that Italy exports the most products (578) and Austria is ranked 7th in terms of total exports as shown in Figure 5.6 (1) and (2).

The results listed above could also be detected from a simple bar chart that shows the total number of products per country. If only this information has to be visualized a bar chart might be even more suitable since it uses a bar's length to visually encode the quantity which is known to be significantly more accurate than color encoding which is used in `Scets`. However, the main advantage of `Scets` over a simple visualizations is its detailed overview which enables users to both perform basic tasks and to gain deeper insights. This implies, that on the one hand users can still identify the largest set and empty sets aided by the ranking feature and on the other hand they can use the same view to dig deeper and inspect the data more precisely which is a unique feature basic visualizations don't have.

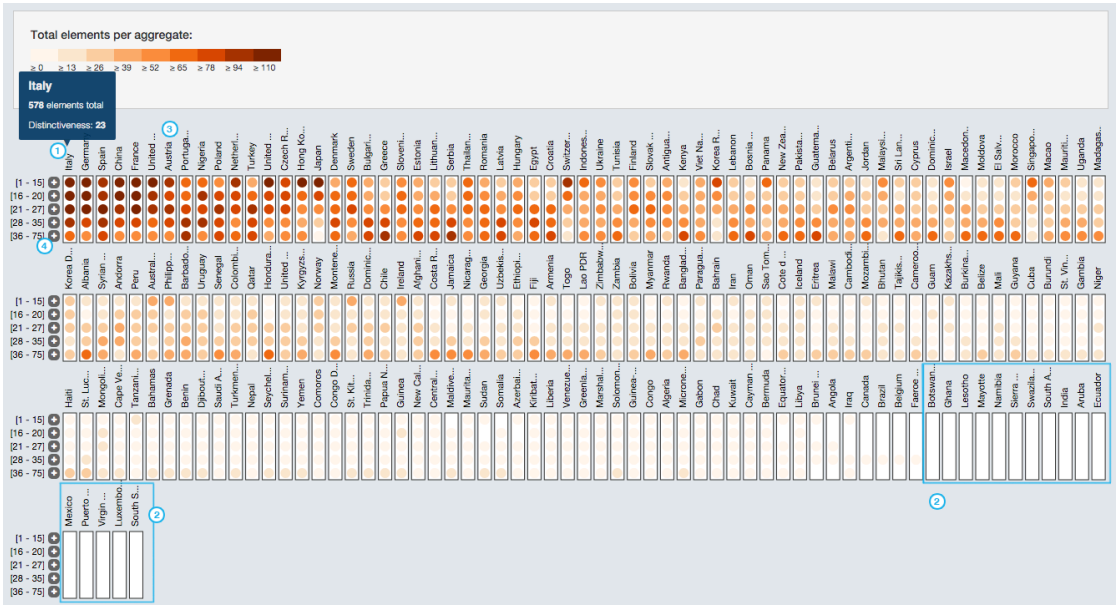


Figure 5.6: (1) Italy exports 578 products in total and is ranked first. (2) Austria ranks in seventh place. (3) Sorting by quantity reveals 18 countries which don't export any products. (4) The maximum degree is 75.

By taking a closer look at the bins and their ranges, one will notice that the maximum degree is 75. Thus, there is at least one good which is exported by 75 different countries. It also implies that there is no good which is exported by more than 75 different

countries (see Figure 5.6 (4)). As the user expands the last bin she will notice that all subsets in the last row (degree 75) are brightly colored, i.e., they contain relatively few elements and hence only a few elements are exported by 75 countries.

Another very interesting insight with this dataset which is worth reporting is the comparison of countries based on their distinctiveness. As described earlier in Section 3.2.7, a set's distinctiveness is given by its average degree across all bins. In the context of this dataset distinctiveness indicates whether a country's economy is more competitive compared to other countries which means that it produces products that only a small number of other countries produces. As an example, those countries in Figure 5.7 shown in blue tend to produce more distinctive products than non-distinctive products. This is characterized by the transition from dark on top to light at the bottom of the set. On the other hand, countries shown in red produce less of the distinctive products than of non-distinctive products. In this case the transition is from bright on top to dark at the bottom of a set. Thus, the majority of products produced by these countries are non-distinctive. Therefore it can be concluded, that their economies might be less competitive. Especially this example demonstrates the effectiveness of the visualization technique as these findings were neither visible in the raw data nor could be easily revealed by existing visualization techniques listed in Chapter 2.

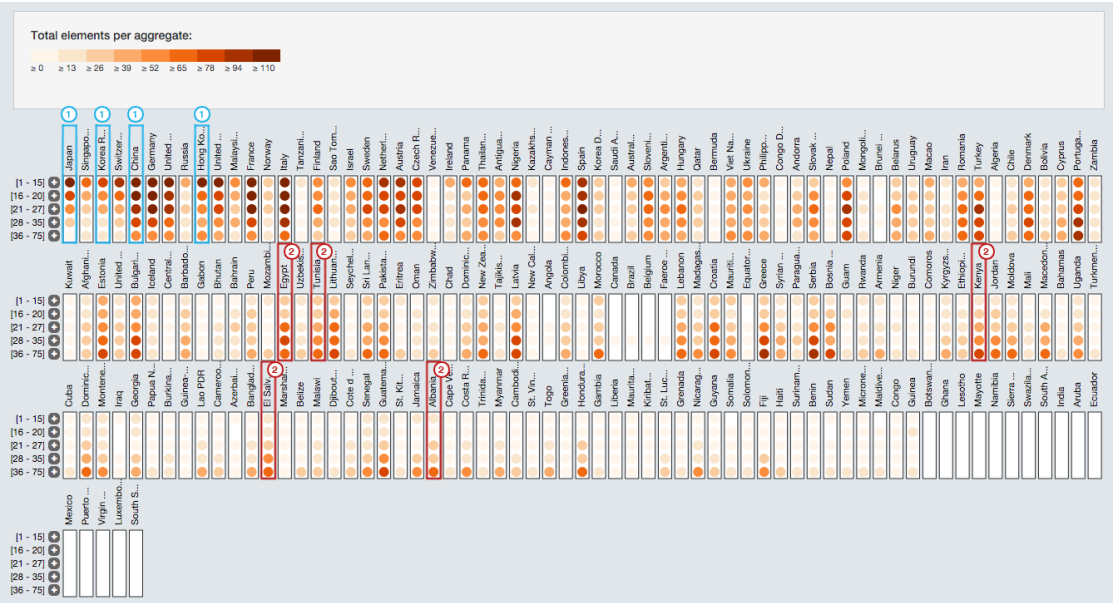
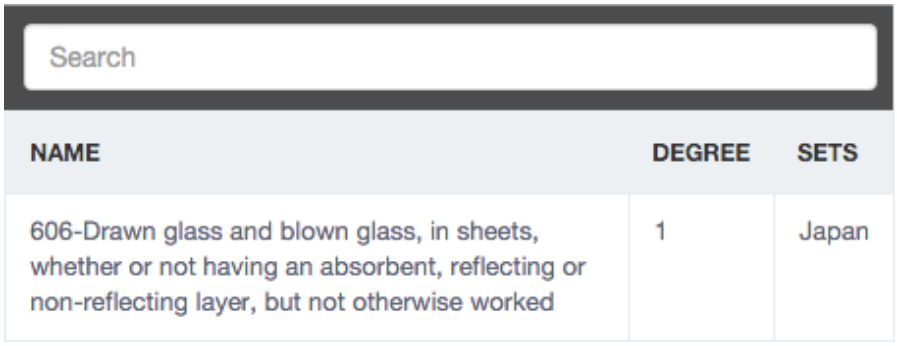


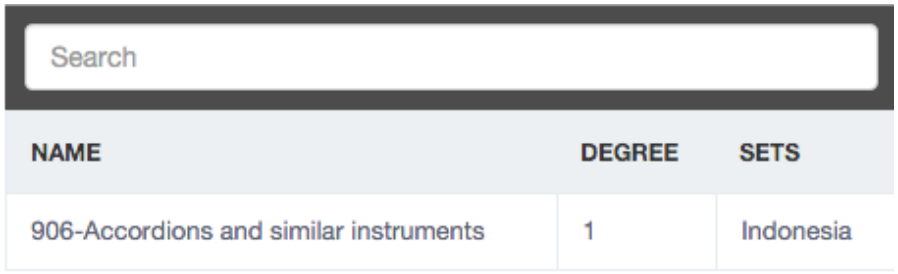
Figure 5.7: Countries sorted by distinctiveness. (1) Sets highlighted in blue are examples for countries which produce more distinctive products by comparison. (2) By contrast, sets in red show countries that produce more non-distinctive products.

Furthermore, it is noticeable that Japan and Indonesia are the only two countries which export unique products, i.e., goods which are not exported by any other country. In the

matrix layout this characteristic is shown as a subset of degree 1 in the corresponding sets. Figure 5.8 displays details about the product which is only exported by Japan and Indonesia respectively.



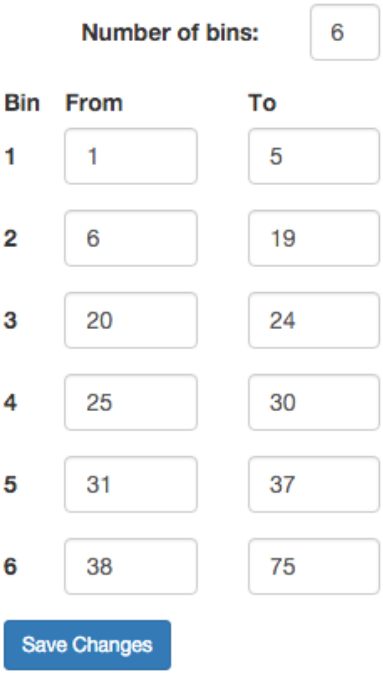
(a)



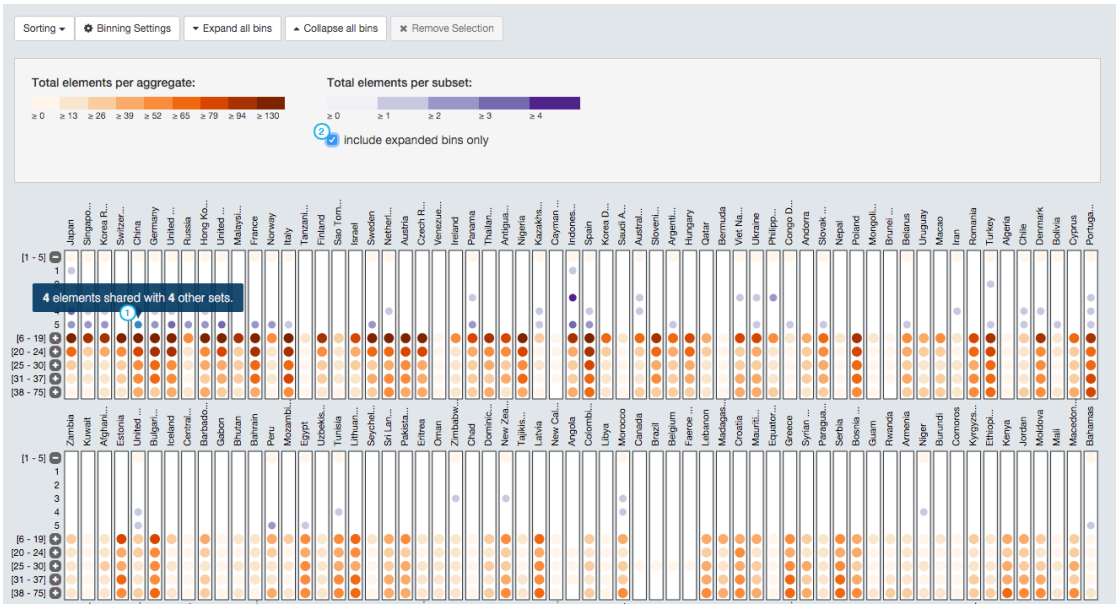
(b)

Figure 5.8: Element view showing the product's name which is only exported by (a) Japan and (b) Indonesia respectively.

Experimenting with different bin ranges often helps to gain deeper insights into the data. As an example, China produces most products of degree < 5, i.e, that four or less other countries produce. This gives China competitive and negotiation advantages over these products in comparison to products which are produced by many countries.
In order to reveal this pattern, several steps are necessary. First an additional bin which ranges from degree one to degree five is added manually through the binning view. In this respect the range for the second bin is changed to 6-19. Figure 5.5 shows the modified binning settings. Following this, in the matrix view bins 1-5 are expanded which displays the legend for subsets. Now, the checkbox 'include expanded bins only' is ticked. Sorting the data by 'Distinctiveness' ranks China among the most distinctive countries. The result is shown in Figure 5.9(b).



(a)



(b)

Figure 5.9: Modified binning settings are shown in (a). (b) (1) highlights China's degree 5 in the matrix view with subsets colored according to the active setting (b) (2).

CHAPTER 6

Evaluation

In this chapter the evaluation method which was used for evaluating the prototype is described in detail.
Section 6.1 provides a brief summary of evaluation methods for information visualization techniques based on literature research. Following this, a description of the evaluation method which has been used in Section 6.2 is given. The chapter is concluded with the results obtained from the evaluation phase in Section 6.3.

6.1 InfoVis Evaluation Techniques

As information visualization is increasingly adopted in different domains, there's also a strong need for comparing and evaluating existing tools. In general, evaluation methods for information visualization aim at providing proof for the effectiveness of such techniques. Literature research has shown that several different methods and approaches for evaluating information visualization techniques exist. However, since evaluation is not the main focus of this thesis I will only give a brief overview of the state of the art in this section.

Plaisant et. al. [Pla04] categorized current evaluation practices into four classes.

- Controlled experiments comparing design elements
- Usability evaluation
- Controlled experiments comparing two or more tools
- Case studies of tools in realistic settings

Plaisant also described challenges in the context of information visualization, such as improving user testing. In this context, she mentioned that users often need to look at the same data from different perspectives over a long time. For example, biologists might want to analyze the data set for months trusting that they will find certain patterns [Pla04].

Different criteria can be evaluated for systems that use visual representation, such as functionality, effectiveness, efficiency, usability and usefulness. [Maz09] However, the evaluation process can be limited to a subset of this list, depending on the visualization's type and the available resources. "Not having their own evaluation methodology, systems that use visual representations have adopted techniques from human-computer interaction. These techniques, which have been in use for years, can essentially be subdivided into two categories: analytic evaluations and empirical evaluations." [Maz09, p. 128]
Analytical methods, such as heuristic evaluation and cognitive walkthrough are typically carried out by experts to test whether a system complies with a set of defined heuristics. These types of evaluation don't involve users and are often applied in the initial design phase or on an early prototype in order to identify major usability issues. Empirical methods on the other hand require users to participate. Depending on the type of collected data, such experiments can be categorized into quantitative and qualitative studies.
Controlled experiments are one example for quantitative studies which focus on the verification of hypotheses. Such methods are often used to evaluate the effectiveness and efficiency of a visual representation. [Maz09] For example, the efficiency of a prototype has to be evaluated under the hypothesis that the prototype's efficiency is higher than a legacy system's efficiency. Therefore, a group of test users is split into two groups. One group is asked to perform a set of given tasks by using a legacy system, whereas the second group is asked to solve the same tasks by using the system to be evaluated. Finally, the time required by each of the two groups to solve the tasks is measured and compared in order to confirm or refute the hypothesis. Typically such methods require a high amount of resources. Qualitative methods such as user tests, interviews and focus groups enable the evaluation of a representation's usefulness and require a smaller group of participants compared to quantitative methods.

Lewis [Lew82] introduced a very popular approach for the evaluation of visualization techniques, called the 'Thinking Aloud' method. The basic idea of this method is to ask users to share their thoughts as they perform a set of tasks. All expressed thoughts, including feelings, expectations or questions which might arise as the users work, are recorded or noted by an observer. The recordings and notes are analyzed afterwards in order to get a better understanding of the system's usability.

Based on the analysis of over 800 papers in the field of information visualization of which 361 include evaluations, Lam et. al. [LBI+12] identified seven different evaluation

scenarios. These scenarios are categorized into scenarios for understanding data analysis processes and those which evaluate the visualizations themselves. In addition, the authors provide a list of evaluation questions and identified popular goals and outputs for each of those scenarios.

It can be summarized, that various methods for evaluating information visualization techniques and systems exist. Deciding which evaluation method to use depends on various factors, such as the stage of development, the quality of available test data, the number of test users, etc. Thus, there is no general solution to the problem of choosing a suitable evaluation method. Instead, each case has to be individually analyzed in order to find the most appropriate evaluation method.

6.2 Evaluation Method

Since this work is mainly focused on solving the problem of scalability with the help of a novel visualization technique for set-typed data, an extended empirical evaluation of the prototype would go beyond the scope of the thesis. Thus, I decided to conduct a rather compact study which involves only a small number of test candidates. The goal of the conducted user test is the gathering of qualitative data and feedback regarding the usability and understandability of the used design elements and visual metaphors.

6.2.1 Evaluation Process

The evaluation process can be split up into the following steps:

- **Introduction:** The test person is introduced to the topic and the test procedure.
- **User test:** The test person can explore the user interface and has to solve a list of defined tasks.
- **Final interview:** The test person is asked about his overall impression and in particular about potential issues and improvement ideas.

At first the user receives a brief introduction into information visualization in general and particularly visualization of set-typed data. In addition, the user learns about the motivational aspects of this work and the problem description. This introduction also includes a very concise summary of previous work in this field and the goal of this thesis. The user is also informed about the purpose of the evaluation and his role during the test procedure. Subsequently he is familiarized with the prototype's user interface.

In the next step the actual user test is performed. During this phase the screen and audio are recorded. A list of tasks (see appendix) is given to the test subject. While the user tries to solve each of these tasks the 'Thinking Aloud' method is applied, i.e., the test person is asked to share his thoughts and impressions during the test. The order in

which the tasks are completed is irrelevant since the tasks are independent of each other. Besides, there are no time constraints for solving the tasks which allows the test subject to take as much time as needed.

Finally a short interview is conducted whose aim is to question the user about his overall impression of the prototype. Moreover the test subject is asked about improvement ideas and any problems the user might have discovered during the evaluation process.

6.2.2 Equipment and Test Data

All tests were performed on the same notebook running Mac OS 10.11.3 as operating system. As testing environment Chrome version 47.0 was used. In the interest of readability the notebook was connected to a 24 inch external monitor. For screen capturing QuickTime Player version 10.4 was used. Audio was recorded with a smart phone.

In order to be able to test the developed prototype's effectiveness and its intuitiveness, preferably test data from real world scenarios should be used. For all tasks listed in the appendix the *Countries Exports* [Com] dataset which has been introduced in Section 5.2 has been used. This dataset comprises both a large number of elements and sets and it is best suited to demonstrate `Scets`' effectiveness among the available datasets.

6.3 Results

As the developed prototype does not focus on a particular domain, the test users do not necessarily have to be domain experts either. However, test subjects who have basic knowledge of mathematics and understanding for usability of software interfaces are well-suited. Thus, I decided to select three persons from my personal environment.

The first test subject (S1) was a male 29-year-old computer science student who has been working as a software developer for several years and is well-experienced in the field of visualizations. Both, his educational background and his professional experience qualified him as a suitable test subject.
The second test subject (S2) was a male 35-year-old former software developer who is working as an IT project manager today. He holds a degree in business administration with specialization in business informatics and has developed and evaluated various visualizations during his professional career.
The third test subject (S3) was a female 28-year-old management student working as competitor and market analyst for an international insurance company. She has basic experience with visualizations which she mostly gained during her studies.

The total time spent on the evaluation was about 60 minutes per person, of which approximately 20 minutes were spent on the introduction and a quick demonstration of the prototype's user interface. The actual user test took about 40 minutes.

6.3.1 User Test

Following the introduction, each test subject was asked to solve eight tasks in order to evaluate the usability of the prototype as well as the understandability of the visual metaphors and encodings in use. The user test has shown, that the overall usability is very good and most of the tasks could have been solved within reasonable time. The results as well as minor usability issues which could have been identified during the test are discussed in the following.

In the first task the user simply had to identify the highest element degree of the set Nigeria. The primary goal of this task was to verify if the user has understood the visual metaphor of sets, aggregates and subsets as well as the encoding of quantity. By expanding the last bin and trying to identify the bottommost subset in the relevant set, **S1**'s approach was absolutely correct. However, he could not identify the correct subset which would have been the subset of degree 68. Instead, the user's answer was 62. This wrong answer was very likely due to color encoding. Since the relevant subset includes only a single element, it is brightly colored. This and a set's light background color make it hard for users to identify such subsets without moving the mouse over. In addition, sometimes subset tooltip's don't show up immediately since the mouse is not exactly placed over a subset or moved away too fast.
Test subject **S2** expanded the last bin and could quickly identify degree 68 by hovering over the corresponding subset. However, he also pointed out that bright subsets are hard to identify.
At first, user **S3** had some problems understanding this task as she was not familiar with the used terminology (sets, elements, degrees, etc.). After a more detailed explanation, she understood the visual metaphor and could solve this task by herself.

Another problem occurred during the second task, which required the user to identify at least one product which is exported by exactly five countries. Again, **S1** approached the task in the correct way as he expanded the first bin and searched for subsets of degree 5. As he moved the mouse away from the bin labels, he lost track of the relevant row for a moment and had to start over by placing the mouse to the correct degree label again. In the end he managed to solve this task by identifying one product of degree 5 as he received the necessary information from tooltips while he moved the mouse along subsets.
First **S2** expanded the first bin and tried to find an element of degree 5 in Austria. He noted that Austria does not contain such a subset and moved on until he found a set which contained the requested subset. By selecting the corresponding subset he identified the element '206-Cobalt ores and concentrates'. **S3** could solve this task instantly

without any issues and came to the same conclusion as test subject **S2** by selecting the subset of degree 5 in the Congo.
It can be concluded that tooltips offer an intuitive way to provide additional information.

When asked to identify the set which contains the most unique elements, the task had to be explained more detailed as the meaning of the term 'most unique elements' was not clear to the first test subject (**S1**). Test subjects **S2** and **S3** also had difficulties in understanding the term 'most unique elements' as they confused it with a set's distinctiveness. After rephrasing this task, all test subjects could quickly solve this task by identifying Japan as the correct answer.

Identifying all sets that include the element '616-Tin pipes or tubes and pipe fittings' as well as the element's degree was quickly achieved by the three test subjects. All of them instantly used the search input field to solve this task which proves that they fully understood the purpose of this user interface element. **S1** and **S2** identified the sets that were searched for by means of the element view, **S3** used the matrix view in order to identify the highlighted set names.

Task 5 which was originally intended to be solved by using the bin configurator in order to change the bin settings, was approached in a different way by all test users. Instead of creating a new bin [5-5] which would have visually isolated the target degree, the test subjects did not modify the default bins but used the overview in combination with the expanding bin feature to explore the data. It seemed that the test subjects were not aware of the binning view and its purpose as this feature might not have been explained in greater depth in the beginning. Another reason for not using the binning view might be that the users decided to rather resort to well-known practices which have proved to be successful in previous tasks than trying a different approach. Nevertheless, this task was solved by all test subjects. While users **S1** and **S2** identified United States and Sweden, **S3** reported United States and United Kingdom as the result. One interesting observation made during this task was that **S1** independently recognized the visual metaphor of blue colored arcs which represents the portion of elements shared between a subset and the current selection. He pointed this understanding out as he drilled down into the expanded bin and investigated the data by selecting several subsets. This shows that the used visual metaphor was intuitively understood by the test subject.

The final three tasks were used to check if the test user had understood the concept of distinctiveness. Furthermore, these tasks also test if the sorting feature can be applied correctly by the user. The user test has shown that the concept of distinctiveness, i.e., the meaning of the color gradient, as well as the related sort method seemed to be well understood by all test subjects as they were able to quickly identify the most competitive country and compare two countries in terms of distinctiveness. Also, the three test persons were able to dig deeper and find out how many products exported

by the most competitive country are also shared with less than 15 other countries. In this regard it is noteworthy to mention that **S3** calculated the sum of all bins instead of hovering over the set's name label. In the last task which required the user to identify the least competitive country and report the number of elements shared with 19 other sets, **S1** and **S3** were also able to identify and exclude empty sets immediately and named Guinea as the solution to this task. **S2** first reported South Sudan as the least competitive country. Later he realized with the help of tooltips that this country does not export any goods at all. Eventually, **S2** reported Guinea as the least competitive country with 31 exported products in a total.

6.3.2 Interview

The concluding interview was conducted after the user finished all tasks. First, the test subjects were asked about their overall impression of the prototype. All subjects stated that they were very pleased with the user interface and they noted that the used visual encodings are intuitive and understandable. Moreover test subjects **S1** and **S3** remarked that a more detailed introduction would have helped to avoid any misunderstanding. As an example, **S1** mentioned that he sometimes confused the expression 'shared with 4 other sets' with degree 4 instead of degree 5. **S3** stated that she was a little overwhelmed with the used terminology in the beginning due to her lack of experience in that field.

Next the test subjects were asked about their opinion concerning the advantages of the developed prototype and any usability issues they might have discovered during the user test. **S1** and **S3** pointed out that the visual representation of sets and elements was very suitable. Furthermore, **S3** noted that the interaction methods such as tooltips and the possibility of expanding bins were very intuitive. Besides, **S1** and **S2** were impressed by the number of sets which were visualized in the test data and they could think of even a larger number of sets being depicted without compromising on usability.
As for usability issues, all three test subjects brought up that some colors were too bright, in particular those which encode a small amount of elements. Especially when brightly colored subsets are displayed on a set's white background recognizing these subsets may be problematic. As a result, **S1** suggested to either change a set's background color or to use darker colors for quantity encoding in order to increase the contrast. **S2** came up with a similar suggestion but recommended to change a set's background color only when the user moves the cursor over it.

Finally the test users were asked if they could think of any ideas to improve the usability of the system. One improvement idea **S1** reported is to implement a so called 'sticky header'. This means that the elements located in the control panel always remain in view at the top of the page even when the user scrolls down. Thus, users can always access the control elements without having to scroll all the way back up to the top to select a different filter method or remove the current selection for instance. This is a

common practice in web development which improves user experience a lot. In addition, **S1** noted that the sort menu could also be improved in the sense that the label always states the selected sort method instead of a static text. As an example, when switching to alphabetical sorting, the text would change to 'Name (A-Z)'. This way, users will always be aware of the active sort method. **S2** did not report any additional improvement ideas besides changing a set's background color. **S3** noted that she would have preferred full set names to be displayed instead of abbreviations.

6.3.3 Summary

The evaluation has shown that the prototype's user interface and the visual representation of element-set memberships was intuitive and comprehensible. Even though the users were no domain experts, they quickly got a main understanding of the visual mappings and the main concepts. After a brief introduction all test subjects were able to successfully solve all of the given tasks.
The test subjects **S1** and **S3** especially liked the overview and the possibility to view data in greater detail by expanding bins. In addition, the different sort methods as well as tooltips were used extensively by all test users for solving several tasks which suggests that these interaction possibilities were well understandable. Both the user test and the concluding interview have shown that encoding quantity by using colors seemed very intuitive. However, all test users considered subsets which encode low quantity being colored too brightly as they don't stand out from a set's background clearly. This makes it harder for users to identify those subsets. Moreover, the purpose of the bin configurator seemed not quite obvious to all test users as they didn't make use of this feature and maintained the default bin configuration instead. This can be attributed to the short introduction which did not cover this topic in sufficient detail. Besides, the test subjects reported a few minor usability issues and provided valuable input for further improvements.

CHAPTER 7

Discussion and Future Work

7.1 Discussion

The developed prototype and the evaluation have shown that a visualization technique which combines features from both matrix-based and aggregation-based approaches is on the one hand very suitable for representing aggregates of set elements and on the other hand scales well with an increasing number of elements and sets respectively (research question RQ1 of Section 1.2). By applying Ben Shneiderman's information seeking mantra, the visualization offers an abstracted overview but also enables users to drill down into details. The employed interaction possibilities support users in obtaining in-depth insights into the data in order to reveal previously unrecognized patterns. With the help of two different aggregation methods, the desired scalability in both the number of elements and sets could have been obtained. Therefore the main research question was answered by providing a novel visualization for set-typed data which solves scalability issues with an increasing number of sets (resp. elements) by combining matrix-based and aggregation-based techniques.

The possible number of sets which can be visualized with the novel visualization technique (research question RQ2) can be determined with the help of the provided usage scenarios and the evaluation. The '*countries exports*' data set which was used both in one of the usage scenarios (see Section 5.2) and for the evaluation comprises approximately 200 sets and 1350 elements. By applying advanced aggregation from Section 3.1.2 and arranging sets in a grid layout, the visualization technique could achieve the desired scalability as the system could easily process and visualize the data.
Moreover, the system was also tested with an input file comprising 500 sets. Due to the flexible matrix view and the dynamic formation of set groups this dataset could also be visualized successfully. The result of this test is displayed in Figure 7.1. However, with an increasing number of sets, the initial loading takes longer and does not increase

linearly. As an example, the loading time for 500 sets was approximately 65 seconds compared to 19 seconds for 300 sets. Since there was no data available which involved more than 500 sets, a data set including 550 sets was generated manually by duplication. Loading this data set led to the web browser running out of memory. After about four minutes the data was successfully loaded and visualized as shown in Figure 7.2. Despite the large number of sets interaction such as expanding bins and selecting items was possible without restriction.
Summing the above, tests have shown that data sets including up to 500 sets can be loaded within a reasonable timeframe. Performance becomes the bottleneck when dealing with datasets which include more than 500 sets. The main cause and possible improvements are described in Section 7.2. Apart from this, no further limiting factors could be identified. Depending on the user's screen size and the number of depicted sets, vertical scrolling is inevitable at some point. Nevertheless, the number of bins can be reduced manually anytime. As a result, the overview becomes more compact but the level of detail decreases in turn.

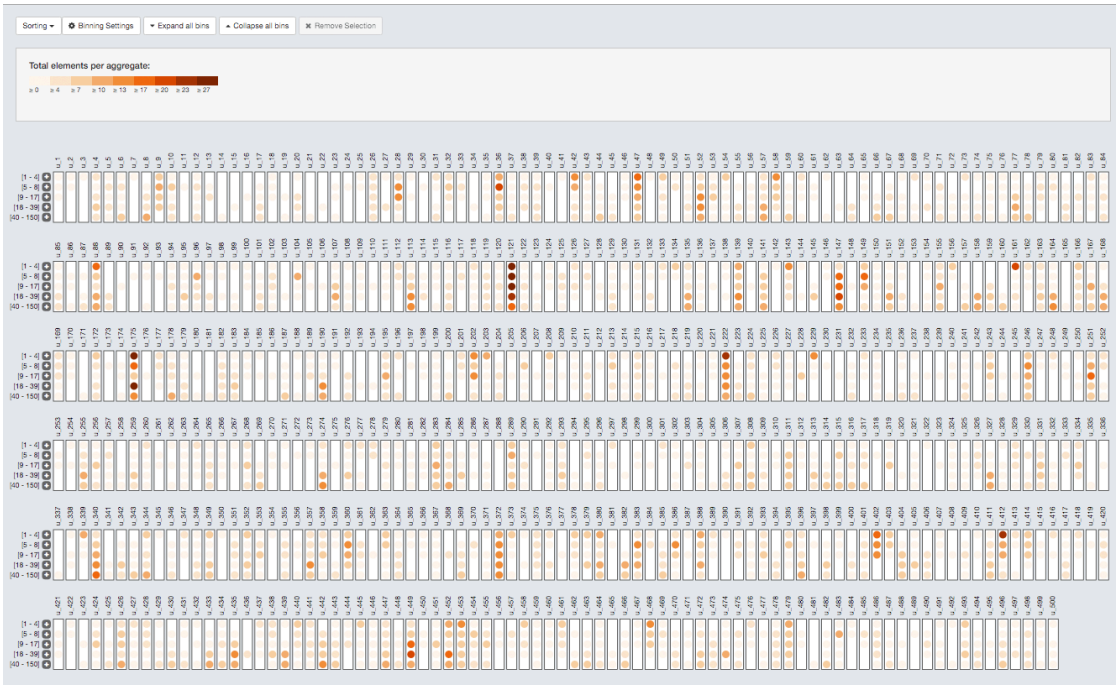


Figure 7.1: 500 sets visualized in `Scets`

Considering the evaluation, the introduced design and the employed visual mappings turned out to be intuitive and understandable (research question RQ3). The test subjects got intuitively familiar with the user interface and could easily solve all tasks. Furthermore, they easily understood the grouping of elements by degree as well as the aggregation of degrees into bins. The novel visual representation of set-typed data is especially useful as it enables users to get a detailed overview of the data and to perform

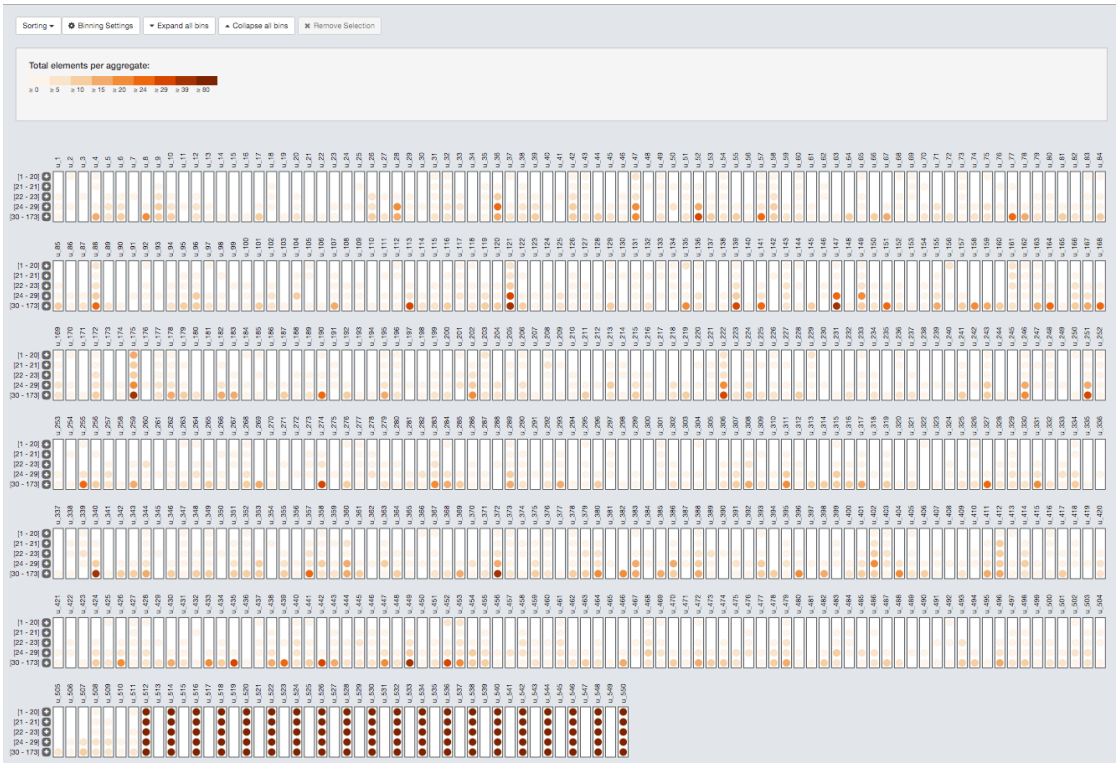


Figure 7.2: 550 sets visualized in `Scets`

basic tasks but also to inspect the data in greater detail in the same view. Moreover, the ranking feature and an interactive element search support users in the investigation process. Thus, the developed technique is a suitable approach for visualizing and analyzing set-typed data. However, some minor usability issues could be identified during the user test and the conducted interview, which provide opportunities for further improvements (see Section 7.2).
The usage scenarios described in Chapter 5 have demonstrated how the visualization can be used in order to visualize and explore real-world data. In Section 5.2 the visualization technique has proved to be useful when economic data which includes a large number of sets needs to be visualized and analyzed. In this regard, the distinctiveness ratio which corresponds to the average degree of a given set turned out to be very valuable as it helped to learn more about the data, such as which countries are more competitive than others and what are the unique products they export.

7.2 Future Work

During research and the evaluation of the prototype several suggestions were offered up for improving the prototype. These ideas are listed in the following:

- **Manual binning view:** The binning view allows users to modify the total number of bins and their ranges. For now users can change these settings through text inputs. As a UI improvement, range sliders can be used instead of input fields. The idea is to display slides vertically aligned to the bars of the degree histogram. Each slider corresponds to a particular bin and ranges from the first degree bar to the last degree bar of a given bin. This indicates a relationship between the degree histogram and the bin ranges and clearly visualizes which degrees belong to a particular bin. By increasing the total number of bins, a new bin could be added at the end. Decreasing the number of bins would remove the last bin. As an alternative, control elements for adding and removing bins could be added between range sliders. Figure 7.3 shows a mockup of the revised binning view.

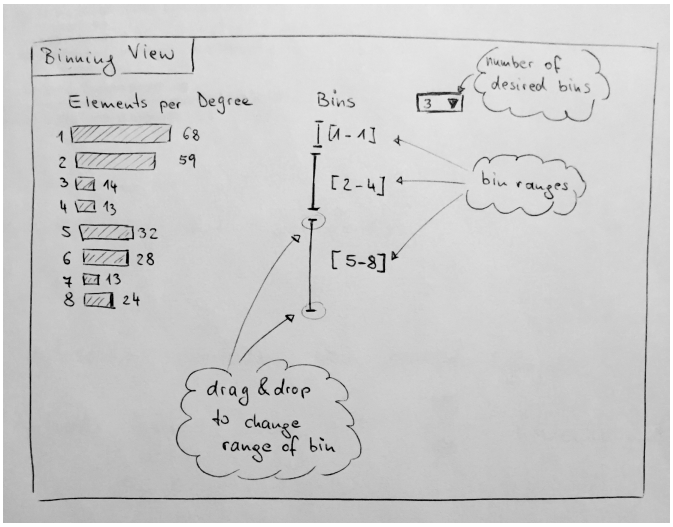


Figure 7.3: A mockup of the revised binning view uses range sliders instead of text input fields.

- **Performance:** At the moment there is one main bottleneck in terms of performance, which is the *createFullGrid* method. This method creates a grid, i.e., a two-dimensional array, with each cell containing an array of elements. This way, grid data can be tied to DOM elements in the further course of the visualization process. Due to nested loops, performance is adversely affected. As an example, it takes about 13 seconds in total to load and visualize the country exports dataset from Section 5.2. Of this, approximately 11 seconds are required by the *createFullGrid* method. One way to improve performance would be to avoid the computation of objects and use simple IDs instead. When binding data to DOM elements a map of IDs to objects could be used instead.

- **Tooltips:** Tooltips are used in many scenarios since they are very intuitive way to display additional information as described in Section 3.2.4. One of the main

challenges when using tooltips is to determine its placement. In `Scets` tooltips are always placed above the triggering element. This is especially problematic when the corresponding element is located close to the screen border and the tooltip displays a lot of information. Thus, parts of the tooltip won't be visible as they are cut off. To avoid this, tooltip position needs to be determined based on the tooltip's width and the element's location. The tooltip in Figure 7.4 would be better placed on the right-hand side of the corresponding aggregate. This way, the entire content can be displayed.

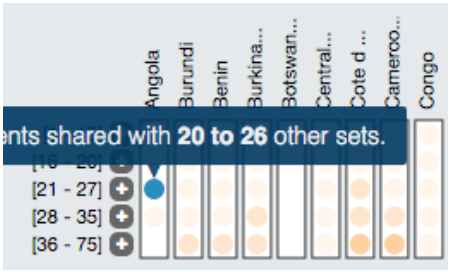


Figure 7.4: An aggregate's tooltip extends beyond the edge of the screen.

- **File parser:** As already noted in Section 3.3 there are some restrictions regarding the file structure when a CSV files is uploaded and parsed. Currently only files are supported which have sets represented as columns and elements as rows. Ideally this restriction should be removed. An additional setting in the description file could be added which indicates how sets and elements are arranged in the CSV file. Thereby the preprocessing task of transposing the matrix won't be necessary anymore which increases usability and reduces the overhead of using third party tools or scripts.
- **Element view:** By extending the data table in the element view with the possibility to sort by each column users will be able to rearrange elements in the table. In contrast to sorting of sets which is described in Section 3.2.7, this interaction method enables users to sort elements of the current selection. Like sorting of sets, it can also be classified as 'Reconfiguration' [YKSJ07]. A best practice method is to highlight the selected column name and to add little arrows next to the column title which indicate the sort direction, i.e., ascending or descending. Clicking on a column title changes to sort method. Multiple clicks on the same column title toggle between ascending and descending.

CHAPTER 8

Summary and Conclusion

In information visualization (InfoVis) set-typed data is mainly used to represent memberships between sets and elements. Existing visualization techniques often face the problem of scalability, i.e., they only scale to a small number of sets. The goal of this work was to develop a visualization for set-typed data which scales well with both a large number of sets and and large number of elements. This required investigating different alternatives for the visual design and finding suitable visual mappings and encodings in order to represent information in a meaningful way.

The desired scalability could be achieved by applying data aggregation methods and by using a flexible matrix layout to visualize aggregated information. Data aggregation turned out to be a very effective data reduction method for set-typed data. This is especially useful when large amounts of data need to be visualized. Thus, two aggregation methods were used. First, individual elements are grouped based on their degree. Consequently, data binning is used to reduce the number of depicted elements by grouping degrees into bins.
The developed prototype[1] consists of two interlinked views, the matrix view and the element view. In the matrix view sets are arranged next to each other in a flexible grid layout. A set contains one or more aggregates which are placed in bins. Each bin can be expanded which displays the subsets located in the corresponding aggregate. Color scales encodes the total number of elements placed in a subset resp. aggregate.
By applying the 'Overview first, details on demand' approach users are offered an aggregated overview and can retrieved detailed information through user interaction. The employed interaction possibilities, such as tooltips, enable users to explore the aggregated information interactively and learn more about the data. By selecting aggregates and subsets users can highlight specific items and distinguish data of interest from other

[1] The prototype is available at `http://scets.sybdev.com`

parts which get visually isolated. In addition to the matrix view, an element view was implemented which shows detailed information about selected elements, such as name, degree and the sets they belong to.
The prototype has been developed entirely with the help of modern web technologies, including Javascript and the d3 framework. This makes it accessible to a large group of users. In this regard it could be observed that client-side memory becomes a limiting factor when the number of sets reaches a certain threshold which is expected to be at around 500+ sets. This limitation leaves room for improvements in the future.

Two usage scenarios have demonstrated how the visualization can be used to visualize real-world data and how it helped to reveal formerly unknown patterns. Furthermore, an evaluation was performed which aimed at gathering qualitative data and identifying possible usability issues. The user test has shown that the user interface was intuitive and the used visual mappings and encodings were clearly understandable. Moreover, the concluding interview also identified further improvement possibilities, such as revising the color scale and implementing a control panel which always remains on top of the page.

Appendix: User Tasks

This appendix lists the tasks, which were performed during the evaluation phase described in Section 6.

Task 1

Task Description	Identify the highest element degree in *Nigeria*.
Preconditions	Countries exports data is loaded. The matrix view including all sets, the element view and the control panel are rendered.
Result	The user identifies the element degree by hovering over the bottommost subset in this set.

Task 2

Task Description	Identify at least one product which is exported by five countries in total.
Preconditions	Countries exports data is loaded. The matrix view including all sets, the element view and the control panel are rendered.
Result	The user identifies a product of degree five.

Task 3

Task Description	Find the set which has the most unique elements.
Preconditions	Countries exports data is loaded. The matrix view including all sets, the element view and the control panel are rendered.
Result	The user identifies the set which has the most elements in subset degree 1.

Task 4

Task Description	Identify all sets, which include the element *616-Tin pipes or tubes and pipe fittings* and identify the element's degree.
Preconditions	Countries exports data is loaded. The matrix view including all sets, the element view and the control panel are rendered.
Result	The user identifies all sets which include the element *616-Tin pipes or tubes and pipe fittings* as well as the element's degree by retrieving the information from the element table.

Task 5

Task Description	Identify at least two sets which have more than two elements shared with four other sets.
Preconditions	Countries exports data is loaded. The matrix view including all sets, the element view and the control panel are rendered.
Result	The user identifies subsets of two or more sets which have more than two elements shared with four other sets and also reports the number of elements for each of these subsets.

Task 6

Task Description	Compare *Japan* and *Austria* in terms of their distinctiveness and find out which country is more competitive than the other.
Preconditions	Countries exports data is loaded. The matrix view including all sets, the element view and the control panel are rendered.
Result	The user states whether *Japan* or *Austria* is more competitive.

Task 7

Task Description	Determine the total number of products the most competitive country exports. Also, identify how many of these are shared with less than 15 other countries.
Preconditions	Countries exports data is loaded. The matrix view including all sets, the element view and the control panel are rendered.
Result	The user reports the overall number of products which is exported by the most competitive country. Moreover, the user reports how many elements are shared with less than 15 other countries.

Task 8

Task Description	Report the number of elements of the least competitive country shared with exactly 19 other sets
Preconditions	Countries exports data is loaded. The matrix view including all sets, the element view and the control panel are rendered.
Result	The user identifies the least competitive country and the number of elements in this set shared with exactly 19 other sets.

Bibliography

[AAMH13] Bilal Alsallakh, Wolfgang Aigner, Silvia Miksch, and Helwig Hauser. Radial sets: Interactive visual analysis of large overlapping sets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2496–2505, 2013.

[ACD12] Julie Steele Alistair Croll and Mike Loukides. Edd Dumbill. *Planning for Big Data*. O'Reilly, 2012.

[AMA+14] Bilal Alsallakh, Luana Micallef, Wolfgang Aigner, Helwig Hauser, Silvia Miksch, and Peter Rodgers. Visualizing sets and set-typed data: state-of-the-art and future challenges. In *Eurographics Conference on Visualization*, 2014.

[api] apidoc. http://apidocjs.com/. Accessed: 2016-03-13.

[ARRC11] B. Alper, N.H. Riche, G. Ramos, and Mary Czerwinski. Design study of linesets, a novel set visualization technique. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2259–2267, Dec 2011.

[BH09] Michael Bostock and Jeffrey Heer. Protovis: A graphical toolkit for visualization. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2009.

[BOH11] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D3: Data-driven documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2011.

[boo] Bootstrap. http://www.getbootstrap.com. Accessed: 2016-03-13.

[col] Colorbrewer. http://github.com/mbostock/d3/tree/master/lib/colorbrewer. Accessed: 2016-03-13.

[Com] Comtrade. Exports of countries by products. Accessed: 2016-03-13.

[CPC09] C. Collins, G. Penn, and S. Carpendale. Bubble sets: Revealing set relations with isocontours over existing visualizations. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):1009–1016, Nov 2009.

[csv] csv-parse. http://csv.adaltas.com/parse/. Accessed: 2016-03-13.

[d3t] d3-tip. http://github.com/Caged/d3-tip. Accessed: 2016-03-13.

[DE98] Alan Dix and Geoffrey Ellis. Starting simple: Adding value to static visualisation through simple interaction. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '98, pages 124–134, New York, NY, USA, 1998. ACM.

[Dew12] Mike Dewar. *Getting Started with D3.* O'Reilly, 2012.

[DRRD12] Marian Dörk, Nathalie Henry Riche, Gonzalo Ramos, and Susan Dumais. Pivotpaths: Strolling through faceted information spaces. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2709–2718, 2012.

[exp] Express.js. http://expressjs.com/. Accessed: 2016-03-13.

[Few09] Stephen Few. *Now You See It: Simple Visualization Techniques for Quantitative Analysis.* Analytics Press, USA, 1st edition, 2009.

[FMH08] Wolfgang Freiler, Krešimir Matković, and Helwig Hauser. Interactive visual analysis of set-typed data. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1340–1347, 2008.

[fon] Fontawesome. http://fortawesome.github.io/Font-Awesome/. Accessed: 2016-03-13.

[Gro] GroupLens. Movielens 100k data set. Accessed: 2016-03-13.

[gul] Gulp.js. http://gulpjs.com.

[her] Heroku. http://www.heroku.com/. Accessed: 2016-03-13.

[Hof00] Heike Hofmann. Exploring categorical data: interactive mosaic plots. *Metrika*, 51(1):11–26, 2000.

[Jac12] Julie A. Jacko. *Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications, Third Edition.* CRC Press, Inc., Boca Raton, FL, USA, 3rd edition, 2012.

[jqua] Jquery. http://www.jquery.com. Accessed: 2016-03-13.

[jqub] Jquery form plugin. http://malsup.com/jquery/form/. Accessed: 2016-03-13.

[jsd] Jsdoc. http://usejsdoc.org/. Accessed: 2016-03-13.

[KLS07] Bohyoung Kim, Bongshin Lee, and Jinwook Seo. Visualizing set concordance with permutation matrices and fan diagrams. In *Interacting with Computers*, 2007.

[LBI+12] H. Lam, E. Bertini, P. Isenberg, C. Plaisant, and S. Carpendale. Empirical studies in information visualization: Seven scenarios. *IEEE Transactions on Visualization and Computer Graphics*, 18(9):1520–1536, 2012.

[Lew82] Clayton Lewis. *Using the" thinking-aloud" method in cognitive interface design*. IBM TJ Watson Research Center, 1982.

[LGS+14] Alexander Lex, Nils Gehlenborg, Hendrik Strobelt, Romain Vuillemot, and Hanspeter Pfister. Upset: Visualization of intersecting sets. *IEEE Transactions on Visualization and Computer Graphics*, 20(12), 2014.

[MA14] Silvia Miksch and Wolfgang Aigner. A matter of time: Applying a data-users-tasks design triangle to visual analytics of time-oriented data. *Computers & Graphics, Special Section on Visual Analytics*, 38(C):286–290, 2014.

[Maz09] Riccardo Mazza. *Introduction to Information Visualization*. Springer Publishing Company, Incorporated, 1 edition, 2009.

[Mis06] Kazuo Misue. Drawing bipartite graphs as anchored maps. In *Asia-Pacific Symposium on Information Visualisation (APVIS) (2006)*, pages 169–177. Australian Computer Society, Inc., 2006.

[mul] multer. http://github.com/expressjs/multer. Accessed: 2016-03-13.

[Mur13] Scott Murray. *Interactive Data Visualization for the Web*. O'Reilly, 2013.

[nod] Node.js. http://nodejs.org. Accessed: 2016-03-13.

[npm] npm. http://www.npmjs.com/. Accessed: 2016-03-13.

[Pla04] Catherine Plaisant. The challenge of information visualization evaluation. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '04, pages 109–116, New York, NY, USA, 2004. ACM.

[Sci] ScienceDaily. Big data, for better or worse: 90 of world's data generated over last two years. http://www.sciencedaily.com/releases/2013/05/130522085217.htm. Accessed: 2016-03-13.

[SGL08] John Stasko, Carsten Görg, and Zhicheng Liu. Jigsaw: Supporting investigative analysis through interactive visualization. *Information Visualization*, 7(2):118–132, April 2008.

[Shn96] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, VL '96, pages 336–343, Washington, DC, USA, 1996. IEEE Computer Society.

[SMDS14] R. Sadana, T. Major, A. Dove, and J. Stasko. Onset: A visualization technique for large-scale binary set data. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):1993–2002, Dec 2014.

[und] Underscore.js. http://underscorejs.org. Accessed: 2016-03-13.

[vel] Velocity.js. http://julian.com/research/velocity. Accessed: 2016-03-13.

[Voi02] Robert Voigt. *An Extended Scatterplot Matrix and Case Studies in Information Visualization*. Diplomarbeit, 2002.

[War13] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc., 3 edition, 2013.

[WWW+11] Xiaowu Wang, Hanzhong Wang, Jun Wang, Rifei Sun, Jian Wu, Shengyi Liu, Yinqi Bai, Jeong-Hwan Mun, Ian Bancroft, Feng Cheng, Sanwen Huang, Xixiang Li, Wei Hua, Junyi Wang, Xiyin Wang, Michael Freeling, J Chris Pires, Andrew H Paterson, Boulos Chalhoub, Bo Wang, Alice Hayward, Andrew G Sharpe, Beom-Seok Park, Bernd Weisshaar, Binghang Liu, Bo Li, Bo Liu, Chaobo Tong, Chi Song, Christopher Duran, Chunfang Peng, Chunyu Geng, Chushin Koh, Chuyu Lin, David Edwards, Desheng Mu, Di Shen, Eleni Soumpourou, Fei Li, Fiona Fraser, Gavin Conant, Gilles Lassalle, Graham J King, Guusje Bonnema, Haibao Tang, Haiping Wang, Harry Belcram, Heling Zhou, Hideki Hirakawa, Hiroshi Abe, Hui Guo, Hui Wang, Huizhe Jin, Isobel A P Parkin, Jacqueline Batley, Jeong-Sun Kim, Jeremy Just, Jianwen Li, Jiaohui Xu, Jie Deng, Jin A Kim, Jingping Li, Jingyin Yu, Jinling Meng, Jinpeng Wang, Jiumeng Min, Julie Poulain, Katsunori Hatakeyama, Kui Wu, Li Wang, Lu Fang, Martin Trick, Matthew G Links, Meixia Zhao, Mina Jin, Nirala Ramchiary, Nizar Drou, Paul J Berkman, Qingle Cai, Quanfei Huang, Ruiqiang Li, Satoshi Tabata, Shifeng Cheng, Shu Zhang, Shujiang Zhang, Shunmou Huang, Shusei Sato, Silong Sun, Soo-Jin Kwon, Su-Ryun Choi, Tae-Ho Lee, Wei Fan, Xiang Zhao, Xu Tan, Xun Xu, Yan Wang, Yang Qiu, Ye Yin, Yingrui Li, Yongchen Du, Yongcui Liao, Yongpyo Lim, Yoshihiro Narusaka, Yupeng Wang, Zhenyi Wang, Zhenyu Li, Zhiwen Wang, Zhiyong Xiong, and Zhonghua Zhang. The genome of the mesopolyploid crop species brassica rapa. *Nat Genet*, 43(10):1035–1039, 10 2011.

[YEB16] M. A. Yalcin, N. Elmqvist, and B. B. Bederson. Aggreset: Rich and scalable set exploration using visualizations of element aggregations. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):688–697, Jan 2016.

[YKSJ07] Ji Soo Yi, Youn ah Kang, John Stasko, and Julie Jacko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE*

Transactions on Visualization and Computer Graphics, 13(6):1224–1231, November 2007.

[zon] Zonu. http://www.zonu.com/images/0X0/2009-11-19-11185/The-languages-of-Europe.png. Accessed: 2016-03-13.