Collaborative Visual Analytics: Leveraging Mixed Expertise in Data Analysis

Andreas Mathisen

PhD Dissertation

Department of Computer Science
Aarhus University
Denmark
Collaborative Visual Analytics: 
Leveraging Mixed Expertise 
in Data Analysis

A Dissertation
Presented to the Faculty of Science and Technology
of Aarhus University
in Partial Fulfillment of the Requirements
for the PhD Degree

by
Andreas Mathisen
August 30, 2019
Advancements in data analytics research and the explosion of available data across public and private organisations have drastically improved the potential for deriving value from data. Thus, industry and research alike have long promoted the value of data for most parts of society. However, tapping into this pool of opportunity as an organisation is not necessarily easy since the adoption of data analytics often requires domain expertise and multiple collaborating groups of personnel. Machine learning and data mining techniques are already heavily used by dedicated data analysts, but it remains a challenge to incorporate existing knowledge capital and to communicate analytical findings. In the past decade, visual analytics research has therefore focused on the development of interactive technologies that combine automated data analysis and human knowledge using visualization interfaces. While this research has resulted in multiple important techniques for the analysis of large and complex data, there have been relatively little focus on supporting collaboration. However, experience suggest that collaboration is a central aspect in fulfilling the vision of involving real users in data analytics. Consequently, bridging this gap by augmenting visual analytics methods with technology that supports collaboration holds great potential for simultaneously advancing the adoption of data analytics in real-world domains and produce new research on the use of data analytics.

This dissertation bridges the gap by presenting novel visual analytics approaches that enable domain experts to make sense of complex data and by developing interactive data analysis technologies that support aspects of collaboration. These aspects include sharing and re-composition of atomic data analysis components, synchronous interaction on multiple abstraction levels, and integrated data-driven reporting that points towards a new paradigm termed literate analytics. The work is based on collaborations with organisations from domains such as production, healthcare, and public business administration. Thus, working with real-world data and interacting with domain experts have been central components of the research approach. Among others, the contributions include novel methods for the analysis of temporal event sequence data and interactive systems that support collaborative visual analytics. As the types of data and the collaborative challenges that have been addressed transcend the concrete domains presented here, the contributions are also relevant in a more general perspective. Furthermore, the novel systems directly enable future research into different areas of collaboration in data analytics. An example of this is our own work on automatic distribution of visualization interfaces across multiple heterogeneous devices.
Grundet store fremskridt i forskning indenfor data analytics, samt eksplosionen af tilgængelige data på tværs af offentlige og private organisationer, er mulighed-erne for at skabe værdi med data øget drastisk. Erhvervslivet, såvel som forskningsmiljøet, har derfor længe promoveret værdien af data for de fleste dele af samfundet. Det er dog ikke ubesværet at udnytte disse muligheder i en organisation, fordi det ofte kræver domæneekspertise og flere samarbejdende person-alegrupper at inklordere dataanalyse som et centrale værktøj. Selvom machine learning og data mining allerede anvendes af dedikerede dataanalytikere, er det fortsat en udfordring at inklordere eksisterende videnskapital og at kommunikere analytiske resultater. I det sidste årti har forskning indenfor visual analytics derfor fokuseret på at udvikle interaktive teknologier, der kombinerer automatiseret dataanalyse og menneskelig viden ved hjælp af visualiseringsgrænseflader. Selvom denne forskning har resultereret i vigtige teknikker til analyse af store og komplekse data, har der været et relativt lille fokus på at understøtte menneskeligt samarbejde. Erfaringer tyder dog på, at samarbejde er et centralet aspekt for at realisere visionen om at inddrage virkelige brugere i data analytics. Udvidelse af nuværende metoder indenfor visual analytics med interaktive teknologier der muligvis samarbejde, har derfor stort potenti for at fremme anvendelsen af data analytics i virkelige domæner samt producere ny forskning i brugen af data analytics.

Denne afhandling bygger bro mellem visual analytics og de nævnte udfordringer, ved at presentere nye metoder, der gør det muligt for domæneekspertier at analysere komplekse data, samt ved udviklingen af interaktive teknologier der understøtter forskellige aspekter af samarbejde. Dette inkluderer at videregive og rekombinere atomare analysekomponenter, synkron interaktion på flere ab-straktionsniveauer samt integreret datadreen rapportering, der peger imod et nyt paradigme, kaldet literate analytics. Forskningen er baseret på et samarbejde med forskellige organisationer fra domæner såsom produktion, sundhedspleje og offentlig virksomhedsadministration. Arbejdet med faktiske data og interaktionen med domæneekspertier har således været centrale komponenter i den anvendte til-gang. Afhandlingens bidrager blandt andet med nye metoder til analyse af temporo-rale eventsekvenser og interaktive systemer, der understøtter collaborative visual analytics. Da de adresserede datatyper og udfordringer med samarbejde også kan findes i andre domæner, er bidragene således også relevante i et mere generelt perspektiv. Desuden understøtter de nye systemer direkte fremtidig forskning i adskillige områder indenfor samarbejde og avanceret dataanalyse. Et eksempel på dette er vores eget arbejde med automatisk distribution af visualiseringsgrænseflader på tværs af flere heterogene enheder.
Acknowledgments

The last three years of my working life have been filled with both ups and downs—luckily mostly ups. Which is probably to be expected in any PhD project. This dissertation would therefore not have been possible without the support from a lot of people. A sincere thank you to all the people who helped me throughout in one way or another.

First of all, I want to thank my supervisor, Kaj Grønbæk, for creating unique opportunities, his continuous support and advice along the way, and for giving me the freedom to pursue any idea of mine.

I also want to thank the co-authors of the publications that have contributed to the making of this dissertation. Thank you Matthias Nielsen for introducing me to the world of visualization when I first started in the group as a student programmer. Thank you Henrik Korsgaard for introducing me to the wicked world of CSCW. Thank you Clemens N. Klokmose and Roman Rädle for introducing me to the world of Webstrates and literate computing. Thank you Niklas Elmqvist for inspiring guidance on by now several papers. And thank you Sriram Karthik Badam and Tom Horak for fruitful collaborations, nerve-wracking paper talks, and multiple long discussions.

A huge thanks to Niklas Elmqvist for hosting me on my stay abroad and for supervising me during this time. And also a huge thanks to Clemens N. Klokmose, Sriram Karthik Badam, and Tom Horak. You all made my stay in Maryland a truly great experience.

The list of people that made everyday life in Aarhus exciting is long. Thus, to not forget anyone, a humble thanks to all the awesome colleagues and fellow PhD students at the department that made the concept of work tolerable and Friday nights fun.

Without the funding and the collaborators of the DABAI project, this dissertation would not exist. Therefore, also a big thank you to all the people that have been involved in the project.

Still, the biggest gratitude goes to my family. Thank you for your encouragement, never-ending support, and enduring my complaints throughout the last three years. A special thanks to Lone Morratz for your love, always believing in me, and surviving these crazy years. Finally, a special thanks to Axel, you made me smile everyday in the final days of writing this time-consuming thing.

Andreas Mathisen,
Aarhus, August 30, 2019.
Publications Overview

The dissertation proceeds in two parts: Part I is an overview of my work throughout the last three years and Part II contains the publications that form the dissertation. An overview of the publications is listed below.


Contents

Abstract i

Resumé iii

Acknowledgments v

Publications Overview vii

Contents ix

1 Overview 1

1 Introduction 3

1.1 Research Objectives .................................................. 3

1.2 Research Context: DABAI ............................................. 5

1.3 Inspired by Domain Experts .......................................... 5

1.3.1 Danish Business Authority ........................................ 6

1.3.2 Grundfos ...................................................................... 6

1.3.3 Systematic ................................................................. 7

1.4 Summary of Challenges from Domains ............................. 7

2 Background 11

2.1 Visual Analytics ........................................................ 11

2.1.1 Machine Learning for Exploratory Analysis ................. 13

2.1.2 Methods for High-Dimensional Data .......................... 14

2.1.3 Methods for Temporal Event Sequences .................... 15

2.2 Characterizing Current Visual Analytics Research .......... 17

2.2.1 A Strong Focus on Tool Design ................................. 17

2.2.2 A Lacking Focus on Collaboration ............................ 18

2.3 Related Work on Collaboration in Data Analytics .......... 21

2.3.1 A CSCW Perspective ................................................. 21

2.3.2 Replicability and Shareability .................................. 22

2.3.3 Common Ground and Explainability via Storytelling .... 23

2.4 Inspirational Technologies ............................................. 24
2.4.1 Visualization Toolkits ........................................ 24
2.4.2 Literate Computing ........................................... 25

3 Research Approach ............................................... 27
  3.1 Experimental Computer Science: A Data Centric Approach .......... 27
  3.2 Field Work and System Design .................................. 30

4 Designing for Data & Task ......................................... 33
  4.1 Prototype 1: High-Dimensional Data .............................. 33
    4.1.1 Design & Workflow ......................................... 34
    4.1.2 Task: Finding Suspicious Companies ......................... 35
  4.2 Prototype 2: Temporal Event Sequences .......................... 36
    4.2.1 Composite Event Learning .................................... 37
    4.2.2 Sequence Aggregation & Quality Metrics ....................... 38
    4.2.3 Design & Workflow ......................................... 40
    4.2.4 Composite Event Inspection for Explainability ................. 41
    4.2.5 Evaluation with Company Event Data ......................... 41
    4.2.6 Task: Identifying Critical Event Flows ....................... 42
  4.3 Reflections on the Analysis Methods ............................ 43
    4.3.1 Reflections on Composite Event Learning ...................... 43
    4.3.2 Reflections on Domain Expert Adoption ....................... 44

5 Visual Analytics in an Organizational Context ...................... 47
  5.1 An Interview Study ............................................. 47
  5.2 Aspects of Collaboration in Data Work .......................... 48
    5.2.1 Sharing Across Organizational Boundaries ..................... 48
    5.2.2 Transparency and Explainability ................................ 49
    5.2.3 Repeated Sensemaking ........................................ 50
    5.2.4 Study Limitations ............................................ 50
  5.3 Design Implications for Visual Analytics ......................... 50
    5.3.1 Mixed Expertise Collaboration ................................ 51
    5.3.2 Levels of Data Analytics Integration ......................... 52

6 Designing for Collaborative Visual Analytics ...................... 55
  6.1 A Collaborative Component Model ................................ 55
    6.1.1 Design Concepts ............................................. 55
    6.1.2 Vistrates .................................................... 57
    6.1.3 Usage ....................................................... 60
  6.2 Bridging Data Analytics and Storytelling ........................ 61
    6.2.1 Replicability, Recall, & Transparency ......................... 62
    6.2.2 InsideInsights .............................................. 64
    6.2.3 Expert Feedback ............................................ 66
  6.3 Encapsulating Existing Technologies and Analytic Approaches .... 67
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.9.2 Implications</td>
<td>151</td>
</tr>
<tr>
<td>11.9.3 Universal Component Model for Ubilytics</td>
<td>151</td>
</tr>
<tr>
<td>11.10 Conclusion</td>
<td>152</td>
</tr>
<tr>
<td>12 Paper D: InsidelInsights: Integrating Data-Driven Reporting in Collaborative Visual Analytics</td>
<td>153</td>
</tr>
<tr>
<td>12.1 Introduction</td>
<td>155</td>
</tr>
<tr>
<td>12.2 Background</td>
<td>156</td>
</tr>
<tr>
<td>12.2.1 Tracking Exploratory Data Analysis</td>
<td>156</td>
</tr>
<tr>
<td>12.2.2 Data-Driven Storytelling</td>
<td>157</td>
</tr>
<tr>
<td>12.2.3 Notebooks and Toolkits</td>
<td>157</td>
</tr>
<tr>
<td>12.3 Design Framework: Data-Driven Reporting</td>
<td>158</td>
</tr>
<tr>
<td>12.3.1 Challenges for Integrating Data-Driven Reporting</td>
<td>159</td>
</tr>
<tr>
<td>12.3.2 Linking Annotations to Analysis States</td>
<td>160</td>
</tr>
<tr>
<td>12.3.3 Dynamic Insight Hierarchies</td>
<td>160</td>
</tr>
<tr>
<td>12.3.4 Hierarchical Presentation Views</td>
<td>162</td>
</tr>
<tr>
<td>12.4 The InsidelInsights System</td>
<td>163</td>
</tr>
<tr>
<td>12.4.1 Part A: Compose Analysis</td>
<td>163</td>
</tr>
<tr>
<td>12.4.2 Part B: Develop Insight Hierarchy</td>
<td>165</td>
</tr>
<tr>
<td>12.4.3 Part C: Share and Review</td>
<td>166</td>
</tr>
<tr>
<td>12.4.4 Implementation Details</td>
<td>167</td>
</tr>
<tr>
<td>12.5 Usage Scenarios</td>
<td>168</td>
</tr>
<tr>
<td>12.5.1 Baltimore Crime Peaks</td>
<td>169</td>
</tr>
<tr>
<td>12.5.2 Danish Company Event Analysis</td>
<td>170</td>
</tr>
<tr>
<td>12.6 Expert Review</td>
<td>172</td>
</tr>
<tr>
<td>12.6.1 Method</td>
<td>172</td>
</tr>
<tr>
<td>12.6.2 Expert Feedback &amp; Findings</td>
<td>172</td>
</tr>
<tr>
<td>12.7 Discussion &amp; Future Work</td>
<td>173</td>
</tr>
<tr>
<td>12.7.1 Limitations</td>
<td>173</td>
</tr>
<tr>
<td>12.7.2 Implications for Visual Analytics</td>
<td>174</td>
</tr>
<tr>
<td>12.7.3 Towards Literate Analytics</td>
<td>175</td>
</tr>
<tr>
<td>12.8 Conclusion</td>
<td>175</td>
</tr>
<tr>
<td>13 Paper E: Vistribute: Distributing Interactive Visualizations in Dynamic Multi-Device Setups</td>
<td>177</td>
</tr>
<tr>
<td>13.1 Introduction</td>
<td>179</td>
</tr>
<tr>
<td>13.2 Related Work</td>
<td>180</td>
</tr>
<tr>
<td>13.2.1 Visual Data Analysis in Multi-Display Environments</td>
<td>180</td>
</tr>
<tr>
<td>13.2.2 Multi-Device Frameworks &amp; View Distribution</td>
<td>181</td>
</tr>
<tr>
<td>13.3 Design Space: Interactive Visualizations in Multi-Device Environments</td>
<td>182</td>
</tr>
<tr>
<td>13.3.1 Visualization Properties</td>
<td>182</td>
</tr>
<tr>
<td>13.3.2 Visualization Relationships</td>
<td>183</td>
</tr>
<tr>
<td>13.3.3 User Preferences</td>
<td>185</td>
</tr>
</tbody>
</table>
13.3.4 Device Properties ........................................ 185
13.3.5 Device Relationships ...................................... 185
13.4 Heuristics for Deriving a View-Sensitive Distribution and Layout ............................... 186
  13.4.1 Grouping & Alignment Based on View Relationships ........................................... 187
  13.4.2 View Adjustments and Device Assignments ....................................................... 188
  13.4.3 User Preferences ........................................... 189
13.5 The Distribute System ......................................... 189
  13.5.1 Underlying Systems and Dependencies .............................................................. 189
  13.5.2 Deriving Properties ........................................ 190
  13.5.3 Distribution: Grouping, Assignment, and Adjustment ........................................... 191
  13.5.4 Control Panel for User Adaptations ................................................................. 192
13.6 Study: User-Created Distributions .................................................. 192
  13.6.1 User Feedback and Findings .......................................................... 194
13.7 Discussion & Future Work .......................................... 196
  13.7.1 Limitations, Framework Extensions, and Evaluations ........................................... 196
  13.7.2 From Heuristics Towards Formalism .......................................................... 196
  13.7.3 From Distribution Towards Visualization Generation ........................................... 197
13.8 Conclusion ....................................................... 197

14 Poster F: Integrating Guided Clustering in Visual Analytics to Support Domain Expert Reasoning Processes ........................................ 199
  14.1 Introduction ....................................................... 200
  14.2 Related Work ...................................................... 201
  14.3 Exploiting Domain Knowledge ........................................... 202
  14.4 Applied to the Business Auditing Case .................................................. 203
  14.5 Conclusion ....................................................... 203

Bibliography ......................................................... 205
Part I

Overview
1 Introduction

It is rarely a single person’s job to be an expert on all areas related to data utilization, thus the adoption of advanced data analysis methods depends on the degree to which our technologies afford collaboration.

Since the birth of computing, multiple areas of computer science research have produced impressive methods and approaches for the analysis of data. With the increasing availability of data across most parts of society, the promised value of data by such research is currently being tested. Data has become a currency of its own, and organisations across a host of public and industry domains are trying to adopt data analytics as a central component of their business. As a consequence of this development, it is no longer only statisticians or dedicated analysts that utilize data analytics or who are required to provide input to the use of such methods. Thus, while people in general are becoming better with technology and computational thinking may become a component in early school curricula, utilization of data analytics in real-world domains is still posing significant challenges—this include challenges with incorporating existing knowledge capital and supporting collaborative workflows.

This dissertation presents an exploration of how to better support the adoption of data analytics based on state-of-the-art research and challenges from real-world domains. It does so through a visual analytics perspective, as this approach already combines central research on data mining, machine learning, and information visualization.

1.1 Research Objectives

The research presented in this dissertation is motivated by a general ambition of enabling domain experts to utilize the data they already possess. The relevance of my work is grounded in numerous observations about the large quantities of digital data that is being collected today, typically referred to as *big data* [58, 92, 206, 211]. For example, it has been described how utilization of data is still in its infancy and has the potential to unlock great value by making information available at a much higher frequency, supporting decision making, and driving innovation of next generation products [206]. But, making use of existing data is not only challenging because of the sheer volume and heterogeneity of available data repositories. Currently, there exist a large gap between the complexity of advanced analysis methods and the knowledge of those who come in contact with the analysis results. In addition, to account for the heterogeneity of data and support decision making, it becomes unavoidable to utilize domain knowledge and consult domain experts. The first part of my PhD has therefore been focused
on developing novel data analytics tools for domain experts, and thus has been guided by the following research question:

- How can machine learning, information visualization and domain expertise be combined to enable analytics for large and heterogeneous data?

Through my initial work, it became apparent that novel tools for specific data and tasks does not necessarily result in successful adoption within larger organisations or across disparate contexts. Instead, my experiences suggested that being able to collaborate around data analytics is not only a necessity, but also a general solution to several challenges related to data utilization. While the development of targeted applications has been the focus in most related work within visual analytics, there has been a relatively small focus on technologies that afford collaborative workflows. Hence, the later part of my PhD has been guided by the following research questions:

- How can new data analysis solutions, tools, and methods be integrated in organisational contexts?
- What is the role of collaboration, and how can we support collaborative work practices around data analytics?

The former question has been addressed through a combination of getting an understanding of the challenges that our collaborators face and identifying gaps in the research literature. Thus, I have worked extensively with the real-world data presented by our collaborators and held several workshops. In this process, prototyping and ground truth have played central roles. The later questions have largely been addressed through similar approaches. However, there have been a stronger focus on collecting empirical data about current work practices, theorizing, and building prototype systems as proof-of-concept implementations. The research approach as well as the progression of my project will be further described in Chapter 3.

In general, the iterative nature of my PhD project has resulted in a gradual progression from the development of novel analysis tools for specific data and tasks (Chapter 4), to understanding current practices around collaborative data work (Chapter 5), and finally to building systems that integrate advanced analysis methods in ways that afford collaboration (Chapter 6). Thus, the contributions of my dissertation fall in multiple categories: (1) concrete analysis methods, e.g. for analyzing temporal event sequences, (2) empirical findings about current data work practices, (3) design concepts for collaborative visual analytics, and (4) technical contributions in the form of interactive systems. My work therefore builds on a tradition within visual analytics for building interactive tools that combine information visualization and computation, while simultaneously questioning the current research agenda and proposing new ways of thinking about visual analytics centered around collaboration.
1.2 Research Context: DABAI

This work has been conducted as part of the Danish Center for Big Data Analytics driven Innovation (DABAI). The vision behind the project is to establish Denmark as a pioneer in exploiting the full potential of big data \(^1\). To achieve this, DABAI is centered around a partnership between Danish research institutions, IT companies with big data competences, and government institutions representing a range of public interests. The overall approach is based on combining three research themes and three application domains. The research themes include (1) algorithms for efficient processing of data, (2) machine learning for data cleaning, analysis, and decision support, and (3) visual analytics to facilitate exploration and visual interaction with data. The application domains include (1) societal data, (2) food supply chain data, and (3) educational data. Through case projects that each represent an intersection between the research themes and the application domains, the aim is to generate not only research results but also business results and societal value. My work has mainly been centered around the third research theme (visual analytics) and the first application domain (societal data). However, it also addresses challenges from the second research theme (machine learning), and several of the solutions in this dissertation generalize to other application domains as well. My work contributes with research results such as new software principles and analysis methods as well as proof-of-concept implementations that address how future tools and services can be build and thereby generate business results. In addition, my work has the potential to provide societal value by improving public administration, drive innovation of next generation products, and support decision making in healthcare. In the following section, I will describe the concrete domains that have inspired my research.

1.3 Inspired by Domain Experts

Domain expert is a broad term used in many contexts, and thus hard to exactly define. In this context, domain experts refer to professionals of a given domain wherein data analysis is secondary to their main job. They may have limited knowledge of data analysis or none at all, but their expertise allows them to connect information found in data to real-world phenomena. Multiple job roles match this description including accountants, production personnel, and doctors. The work presented in this dissertation is not trying to define general archetypes related to data analysis, but merely acknowledging that the domain expert is an important role for successful utilization of data, and that they are working in an ecology of job roles that include data scientists, programmers, stakeholders, etc.

The real-world data and empirical findings presented in this dissertation is based on collaborations with three external partners: The Danish Business Authority, Grundfos, and Systematic (hospital use case). We have conducted multi-

\(^1\)https://dabai.dk/en
ple meetings with experts in these domains to understand their data, their work process, and to identify challenges that align with gaps in the academic literature. We have also participated in workshops, hackathons, presented visual prototypes when possible, and conducted a series of semi-structured interviews with data workers (i.e. personnel that already to various degrees use data in their everyday work) to document how data is currently used and how people collaborate around data. The domain knowledge for the hospital use case mainly comes from previous work by Stisen et al. [280], hence most of the effort has been to understand the DBA and the Grundfos domains. In this section, I will briefly introduce each of the collaborators. In the following section, I will then summarize common experiences and challenges from the domains that have guided my work.

1.3.1 Danish Business Authority

The Danish Business Authority (DBA) is a public agency within the Ministry of Business and Growth. The DBA is responsible for a wide range of tasks including the maintenance of historical registration data and financial statements for more than 1.5 million danish companies, supporting policy development, and conducting audit oversight. As part of their responsibility, they provide an online platform where companies can register legally required information and upload financial statements[2] as well as a platform for accessing publicly available data[3]. The registration database contains general company information throughout time, such as changes in important relations (e.g. accountants and board members) and changes to core company information (e.g. business type and name). One of the main functions of the DBA is compliance, i.e. making sure that the information they maintain is correct. Therefore, much of the personnel investigates whether the registration information and the financial statements reported by the companies are correct. There are currently several manual parts to their work process, such as extracting data to spreadsheet documents for further analysis and to coordinate activities. Their challenges therefore include maintaining data quality, combining data of many different formats, and utilizing statistical analyses for decision making. In addition, collaborations are often ad-hoc, and when personnel require a deep understanding of existing cases, they occasionally have to redo parts of the existing sensemaking. Besides compliance, the tasks of the DBA also include geographical analyses, fraud and bankruptcy detection, and understanding general market trends.

1.3.2 Grundfos

Grundfos is a large production company of primarily water pumps. In their production, they collect data about how individual production units perform in several different quality tests, which results in thousands of different logged values

---

[2]https://indberet.virk.dk
[3]https://data.virk.dk
for each unit. Currently, this data is used for isolated decisions about whether a given unit tests within certain boundaries. But the data also provides an opportunity for making cross-process decisions - e.g., to match sub-products optimally in the production, understand why products in the real-world fail, or make decisions about individual products based on how they performed in different contexts (e.g., where in the world the product is shipped to). Grundfos is currently in the process of collecting data from all parts of the company in one large data lake, such that different parts of the company can benefit from data analyses that transcend traditional organisational boundaries. One of the challenges with this approach is that the employees who collect and store the data are rarely the same ones who are supposed to utilize it. Therefore, the data is to some extent decontextualized and stored without all the necessary meta data. Grundfos also undertakes specific data experiments, where they collect and combine, e.g., business data, product information and production data in order to improve product modularization. Currently, they spend substantial time collecting, cleaning, and making initial sense of these data, e.g., by including numerous domains experts in the process.

1.3.3 Systematic
Systematic is a software company that runs several projects at the University Hospital of Skejby aiming to improve hospital logistics and analysis of patient data. Their use cases include analysis, prediction, and optimization of entity flows (people, materials, etc.) and task performance in large organizations (e.g. hospitals, airports) and spaces (large buildings, road networks). Prior to my PhD, much of my work dealt with the orderlies at the hospital, who need efficient and flexible coordination methods since they lack awareness of their co-workers’ tasks and progress. Mobile sensing-based data analysis provides the opportunity to solve this challenge and the hospital is an ideal setting to test such technologies, since there already exists much of the needed infrastructure. Patient flow optimization and analysis of healthcare records are two other important tasks at the hospital that are more relevant to the work presented in this dissertation and where data is not yet used to its fullest potential. Traditionally patient flow optimization efforts are only performed in local departments, hence there is little notion of complete enterprise wide side effects (e.g. bottlenecks). By optimizing patient flow across departmental boundaries, treatment pathways and total length of stay can potentially be minimized. Similarly, identifying effects in healthcare records has the potential to improve treatment quality.

1.4 Summary of Challenges from Domains
In this section, I have summarized challenges that transcend the individual collaborations and that have been central motivations for my research. While they are presented as individual challenges here, they are often tightly coupled as later sections will highlight.
C1: Volume and Variety

Comparing instances in high-dimensional data is well studied within both the visualization and machine learning communities, but there are still few solutions that bridge the best of both worlds in a way that enables people who are not data scientist to analyze high-dimensional data. When the number of dimensions increase to several hundreds, it becomes infeasible to manually compare all dimensions. In particular, it is challenging to include domain knowledge in the analysis process to distinguish noise from relevant patterns.

Temporal event sequence data is a specific type of data present in all three domains. A central challenge in utilizing temporal event sequences is to cope with both volume and variety, since the combination of these factors render most sequences unique. This, in turn, means that current visualization methods become impossible to read for humans at only a few thousand events. Moreover, discrete event sequences do not naturally comply with the tabular format needed for most machine learning algorithms. In the DBA domain, companies repeatedly update event information in the registration database and in the hospital domain each patient record includes the entire medical history. As an example, medical tests and treatments influence how future patient histories unfold, hence effective analysis of such event sequences will, e.g., allow medical professionals to optimize diagnosis and treatment pathways.

C2: Flexible Access to Data

A central challenge in all three domains is flexible access to data, i.e. being able to make appropriate data extracts without being capable in SQL or a similar query language. At the DBA, personnel sometimes wait substantial periods of time on data extracts because they have to communicate their criteria to a programmer, who then creates the extract. At Grundfos, their database contains many different tables with several complex relations, which makes the definition of data extracts that may be simple on a conceptual level hard to actually realize. Another factor of this challenge is data overview. Currently, it can be very challenging to get an overview of existing data and the relation between variables when data repositories grow. Thus, technical expertise and knowledge of the available data are two central aspects of this challenge.

C3: Uncertainty and Transparency

Uncertainty is a general challenge in data analytics and the subject is repeatedly addressed by related work. In the domains presented here, uncertainty is not only related to statistical assumptions of algorithms and analysis methods. Uncertainty often occurs when participants of data work activities lack knowledge of the data collection, how variables relate to real-world phenomena, how data has been handled, and the motivations behind critical decisions. Thus, uncertainty also relates to the transparency of the collection and processing of data.
For example, at the DBA it is challenging to define cohorts of an analysis as these depend on multiple combinations of discrete events. Hence, without proper knowledge of all event types it inevitably introduces a level of uncertainty to an analysis.

C4: Repeated Sensemaking

Repeated sensemaking processes occur frequently when people collaborate around data, and especially in scenarios where the process of finding, combining, and understanding data is important to also understand potential findings. This is especially a challenge in the DBA use case, where personnel sometimes repeat investigation steps when either taking over a case from a co-worker or a case is resumed after critical answers arrive. Furthermore, it is also common practice at the DBA to create data extracts, work on them in, e.g., Excel, and after arriving at the required answers store the work as a personal file. Thus, when someone has a similar task, they risk repeating the entire process including identifying an appropriate analysis method. Therefore, repeated sensemaking occurs both when personnel need to recall any reasoning and when they are unaware of past work.

C5: Sharing Data Work and Analysis Results

Several of the challenges relate to the ability of effectively sharing data work and analysis results. Shareability is therefore a challenge across all domains as well. This includes the ability to share methods in the form of scripts or processing pipelines as well as analysis results long with reasoning, motivations, and assumptions. There are at least three parts to this challenge: (1) being able to abstract away atomic analysis parts that then can be reused elsewhere, (2) being able to record the analysis process, and (3) being able to represent the knowledge in different ways for different audiences. A classic example is the difference between sharing an analysis with a stakeholder and another data analyst. The stakeholder is mainly interested in the findings and how they can inform decisions, whereas the analyst is also interested in the process itself and the analysis decisions that were omitted from the final result.

C6: Mixed-Expertise Collaboration

The domains presented here represent the type of large scale data utilization scenarios where multiple types of expertise are required; expertise which any single person rarely possess. Thus, being able to collaborate across expertise is important. However, establishing common ground between data scientists, programmers, domain experts, and stakeholders is challenging as these groups have different understandings of the data and the analysis methods. Effective communication, analytic explainability, and the ability for each collaborator to provide parts of a solution are therefore central challenges in all three domains. For example, data analysts at Grundfos are not also the engineers that are in
charge of the pump production and the tests that occur throughout. Utilizing the test data for analysis therefore requires the input from production experts. At the same time, the production experts also need to understand parts of the data processing in order to confirm that certain computed values make sense in the real-world.
In this chapter, I present the theoretical background and related technologies which the results of this dissertation builds upon.

2.1 Visual Analytics

Visual analytics originates from the multidisciplinary idea that combining information visualization and computational transformation will allow humans to effectively analyze large and complex datasets. The strength of information visualization is that it amplifies human cognition in six basic ways [53]—by (1) increasing cognitive resources, (2) reducing search, (3) enhancing the recognition of patterns, (4) supporting easy perceptual inference of relationships, (5) supporting perceptual monitoring and (6) providing a manipulable medium. Computational transformation then allows these strengths to be utilized for transforming large data into usable information. Due to its multidisciplinary nature, it can be difficult to establish a definition of visual analytics that does not overlap with the research fields of information visualization, data mining, and machine learning. Still, there exists fundamental differences. For example, the main focus in much visualization work is to produce appropriate views for specific types of information. The main focus in visual analytics is on data analysis and how an integrated process between humans and machines can transform data into useful information that can guide decisions. One of the early definitions of visual analytics was presented by Wong et al. [316] in a special issue of the IEEE Computer Graphics and Applications journal in 2004.

“Visual analytics is the formation of abstract visual metaphors in combination with a human information discourse (interaction) that enables detection of the expected and discovery of the unexpected within massive, dynamically changing information spaces.” [316]

While this definition was made with homeland security as the main focus, it captures an important distinction between information visualization and visual analytics; information visualization focus on developing appropriate information representations whereas visual analytics focus on the analytic process, i.e. humans interacting with data in order to generate information — a human information discourse. The following year, Illuminating the path [69] was published which included a broader definition of visual analytics.
“Visual analytics is the science of analytical reasoning facilitated by interactive visual interfaces.” [69]

The central part of this definition is analytical reasoning, which is the human process of arriving at sound conclusions based on data despite conflicting information and cognitive biases. In 2008, Keim et al. [162] elaborated on this definition.

“Visual analytics combines automated analysis techniques with interactive visualizations for an effective understanding, reasoning and decision making on the basis of very large and complex data sets.” [89, 162]

The authors further clarified their definition; stating that the goal of visual analytics is to: (1) synthesize information and derive insight from massive, dynamic, ambiguous, and often conflicting data, (2) detect the expected and discover the unexpected, (3) provide timely, defensible, and understandable assessments, and (4) communicate assessment effectively for action. Their definition emphasizes the use of automated analysis methods coupled with interactive visualizations as central to the analytic process (cf. Figure 2.1).

Figure 2.1: The visual analytics process [89, 162].

The original research agenda Illuminating the Path [69] describes five high-level aspects of visual analytics research: (1) Analytical reasoning, (2) visual representations and interaction techniques, (3) data representations and transformation, (4) production, presentation and dissemination, and (5) evaluation. Thus, visual analytics research deals with both the theory of analytical reasoning as well as concrete technologies that effectively support the envisioned workflow (cf. Figure 2.1). Illuminating the Path [69] has shaped the research agenda for almost a decade now. Thus, it has also been the theoretical foundation for the initial work of my PhD. I will return to several of the aspects of visual analytics
in this and the following chapter, but for an in-depth description of each, I refer to the original research agenda [69].

2.1.1 Machine Learning for Exploratory Analysis

Machine learning is a collection of automated analysis techniques based on statistical learning theory—i.e. machine learning consists of numerous algorithms for inferring patterns in data and computing statistical models. Thus, machine learning is frequently used to derive models in visual analytics and as a prepossessing step for information visualization. A recent Dagstuhl Seminar [163] described how; (1) machine learning provides a collection of automated data compression techniques, which can be utilized to address scalability issues in information visualization, (2) predictive analysis can enhance the usefulness of visual analytics tools, and (3) information visualization can support the understanding of machine learning behavior.

Machine learning can generally be categorized in three types; supervised, unsupervised, and reinforcement learning [99 328]. Supervised learning methods aim to construct predictive models using input data as well as desired output, i.e. the goal is to learn a function that correctly maps input to output. Supervised approaches include both classification, where the output is a set of categories, and regression where the output is a continuous variable. Contrarily, unsupervised learning methods aim to infer statistical patterns only based on input data. Unsupervised learning is typically associated with clustering, where the goal is to group data in a number of clusters. Finally, reinforcement learning deals with software agents that based on a set of actions and a reward/cost function learn to behave optimally through trial and error. Numerous algorithms exist in each of the three categories for various types of data and with different strengths and complexities [99 328]. Thus, choosing appropriate learning algorithms is very context dependent. It is out of the scope of this dissertation to provide a deep introduction to all types of machine learning, but I will mention relevant learning algorithms throughout the background chapter and the thesis in general.

Visual analytics and exploratory analysis in general are iterative and non-linear processes, that are likely to include several avenues of investigation [69 241 299]. The main purpose with exploratory analysis is to derive new hypotheses, ideas, or instances worth further investigation, thus learning accuracy is often not the main requirement. Rather, computations should be fast in order to maintain the user’s ability to explore new patterns. In my work, I have therefore mainly use fast clustering algorithms such as K-means [13] and dimensional reduction techniques such as Principal Components Analysis [295]. In addition, the novel analysis approaches presented in this dissertation can also be categorised as semi-supervised learning, which utilize only partial output to derive statistical models. For example, a partial labeling can be the set of companies known to have committed some type of misconduct, or the pumps known to have failed in
a real-world setting—the goal is then to identify the remaining instances within
the full dataset.

While the main approach in my work has been to combine existing machine
learning algorithms, dimensional reduction techniques, and evaluation metrics to
construct semi-supervised learning methods that incorporate some form of domain
knowledge, I briefly want to mention two recent approaches that are relevant for
future work in visual analytics. In progressive visual analytics \cite{94, 287, 329},
partial results are repeatedly presented to the user throughout the computation
to maintain user attention. This way, users can reason about the relevancy of
computations already before they have concluded. However, a central challenge
with this approach is to support users to not immaturely trust or disregard results.
Weak supervision \cite{133, 254} address the problem of generating labelled training
data for machine learning by leveraging incomplete datasets and labels; "For many
tasks, we can leverage various heuristics and existing datasets as weak supervision.
We can express these various signals in one unifying framework: as functions
which label data."\footnote{hazyresearch.github.io/snorkel/blog/weak_supervision.html}
An exciting part about this type of machine learning is that it can be used to incorporate domain heuristics (e.g. common patterns, rules of
thumb, etc.) in machine learning, and not only from a single domain expert, which
makes it especially relevant for collaboration. However, such methods are still to
be explored in the context of visual analytics—which include designing interfaces
that allow domain experts to express heuristics about the data as functions. While
progressive visual analytics and weak supervision methods have not been explored
within the time frame of my project, they are highly relevant as future avenues
of research. In the following two sections, I will present background work within
visual analytics for the analysis methods I present in Chapter 4.

2.1.2 Methods for High-Dimensional Data

Much research has focused on developing tools that combine information visual-
ization and machine learning to analyze high-dimensional data—a generic type
of data found in most use cases. Several techniques have been developed for
visual mapping of multiple dimensions including the scatterplot matrix \cite{91}, par-
allel coordinates \cite{144, 227}, the radial layout \cite{160}, hybrid constructions \cite{93}, and
glyph-based techniques \cite{56}. Still, visualizing all dimensions severely limits our
ability to spot meaningful patterns. Thus, a common approach in many tools
is to use computational techniques for projecting high-dimensional data to lower
dimensional spaces, and this way enable simpler visual mappings. This is for
instance done using unsupervised methods such as Principal Components Analysis \cite{155, 196} or supervised methods such as Linear Discriminant Analysis \cite{63}.
Combining multiple linked visualizations and projection then allow users to in-
teract with the high-dimensional data space. Either, projections alone constitute
the models affording an open-ended exploratory search for patterns \cite{155}, or the
combination of linked-visualizations and projection is used to inspect other machine learning models [63, 100]. For the latter, a large body of work deals with identifying meaningful ways of modifying the underlying models and this way incorporate domain knowledge. Examples include interactive local operations in parallel coordinates to support clustering [116] and the ability to steer distance functions [49]. Numerous papers also specifically highlight the importance of utilizing domain knowledge, e.g., by using visual queries [123, 278], allowing for regrouping of objects [139], or by computing functions that directly explain a domain expert’s binary labeling [105].

Much related work has interpreted the visual analytics model as giving the human full control to entirely change the outcome of an analysis. In SeekAView [172] users can combine multiple clusters from different sub-spaces, and in the work by Brown et al. [49] users can change the distance function and thereby fit the algorithm to match their expectations. However, such systems introduce the risk of relying on cognitive biases rather than the actual data. Thus, a different focus is to have users interact with a machine learning model that have been trained using unbiased methods. In the work by Krause et al. [173, 175] users can experiment with different input to a prediction algorithm or get instance level explanations and thereby learn what the algorithm is doing without introducing bias. Thus, how to combine machine learning, information visualization, and domain knowledge in a way that provides human control while simultaneously reducing the risk of introducing cognitive biases is still an open challenge. In Section 4.1, I will present my first work on developing a visual analytics approach for high-dimensional data that allow domain experts to identify relevant instances.

### 2.1.3 Methods for Temporal Event Sequences

Temporal event sequences are a data type of particular importance to the domains presented in Section 1.3. Series of discrete events can indicate relevant behavior of both companies, production performance, and patients. Thus, analyzing event sequences has the potential to guide decisions across all three domains. Visualizing and retrieving insights from temporal event sequences has also been the subject of much tool design within visual analytics, particularly in application domains such as electronic health records [109, 219, 319], and different types of click streams [180, 200]. But, as alluded to in section 1.4, major challenges remain for dealing with volume and variety. Firstly, because temporal event sequences do not fit standard table formats used for most machine learning methods. And secondly, because the combination of volume and variety render most sequences unique. In addition, discrete events often require substantial interpretation and explanation as they are often abstractions of complex real-world events. For example, the high-level event for going bankrupt is captured by a series of discrete events at the DBA, each representing different steps. Consequently, companies exhibit several different patterns all representing the process of going bankrupt. As a result, research [219, 319] has explored how to design tools for event sequence
simplification based on user specified queries and rules. Such tools often incorporate custom visualizations based on the icicle plot \cite{177} to display common event paths together with discrete or continuous time. Alternatively, researchers have proposed simple lists \cite{52} and matrix-based visualization views \cite{330}. But these methods are mainly useful after substantial simplification, to the point where the number of unique event types and sequences is smaller than needed for the icicle plot-based visualizations.

There has also been substantial work on incorporating machine learning methods to extract useful patterns. Notably, frequent sequential pattern mining techniques are often used \cite{179, 200, 238}. These methods extract common sub-patterns for a minimum level of support, i.e. the minimum number of sequences that exhibit the pattern. Although these methods reduce the need for manual simplification, it was recently described how these approaches still come with several limitations \cite{199}—such as interpretability and utility. Related approaches include clustering of common sequences \cite{305, 308} and searching for similar sequences based on a reference sequence \cite{84}. The later example better supports interpretability, as users already to some degree know what they are looking for. Still, it can be hard to evaluate clustering results, especially as irrelevant events might have excessive impact. Thus, related work has addressed these challenges by developing tools where sequence outcomes drive the analysis process, similar to supervised learning. Outcomes are powerful for narrowing down relevant patterns and provide context to the user. Gotz et al. \cite{109} have proposed a design where users define event milestones and continuously explore how the addition of new milestones affects the sequence outcome. Conceptually, this method adopts a constructive approach where the users in a bottom-up fashion extends the current model with new milestones. While this address the problems of utility and cluttered visualizations, the bottom-up fashion also severely limits the number of different patterns that can effectively be explored. Related approaches include sequence summarization to provide all possible paths to a given outcome \cite{318} and the use of session outcomes for context \cite{180}. A sub-problem related to defining sequence outcomes, is the construction of relevant cohorts. For this, Krause et al. \cite{173} presents a tool to visually establish cohorts directly from a database, and Malik et al. \cite{205} presents a tool where multiple cohorts are automatically explored and presented to the user through statistical significance.

As discrete events are often not only ordered but also timestamped, time is another aspect that complicates the analysis of temporal event sequences. While time may not be relevant in every scenario, time can directly impact sequence outcomes. An example is the analysis of healthcare records, where the time between different treatments may directly impact the chance of successful recovery. Different approaches have been proposed to assess the role of time, such as the cloudlines visualization technique for detecting event episodes \cite{176}, a tool for creating temporal summaries \cite{306}, and the TimeSpan tool for exploring stroke treatment \cite{202}. In addition, related work has also reported on challenges for querying event sequence data \cite{220, 320}—e.g. for specifying similarity measures,
time intervals, and absence requirements.

While there has been substantial research and tool development for the analysis of temporal event sequences, both from a data mining perspective and a visual analytics perspective, several challenges remain as described in this section. Furthermore, it has not been shown that event sequence analysis is beneficial in the domain of business data. In Section 4.2, I will describe how we both extended the existing visual analytics methods for analyzing event sequences as well as demonstrated its usefulness in the business domain.

2.2 Characterizing Current Visual Analytics Research

The current research agenda in visual analytics has produced several important methods for analyzing different types of data and including human interaction in meaningful ways. Still, several challenges persist for the adoption of visual analytics. In the following, I will present two overall characteristics of current visual analytics research that have guided the work presented in this dissertation.

2.2.1 A Strong Focus on Tool Design

Sections 2.1.2 and 2.1.2 illustrates a defining characteristic of current research within visual analytic—the dominant research approach has been to build tools for specific use scenarios that fulfill the vision behind the general process model presented in Figure 2.1. While theoretical models on occasion have been the subject of related work, current models in visual analytics still resemble the generic model. A recent example is the work of Sacha et al. [263] about uncertainty, awareness and trust in visual analytics. They present how these concepts influence knowledge generation as shown in Figure 2.2. Among other, the authors point out

![Figure 2.2: The knowledge generation model of Sacha et al. [263].](image)

that when uncertainties are present in an analysis and the user has a low degree of awareness, there is a large chance of human-error, which you need to account for in tool design. Still, as with the original model, their work says little about how to concretely design such tools. Thus, visual analytics research has produced several tools interpreting the model in different ways—ranging from full user control of
the underlying analysis method \cite{173} to a more passive user role where the focus is on inspecting the computed results \cite{175}. Nonetheless, as the general process model illustrates, most work is about designing computational methods and visual interfaces that supports a single human deriving insight from data.

2.2.2 A Lacking Focus on Collaboration

Collaboration was mentioned as an important focus already in the original research agenda in the chapter on analytical reasoning—"analytical reasoning must be a richly collaborative process and must adhere to principles and models for collaboration" \cite{69}. Thus, research should utilize existing work on collaboration to develop methods for visual analytics. This includes: (1) task coordination, (2) sharing of information and beliefs, (3) cooperative problem solving and decision making, and (4) decision and rationale reports. The basic theoretical starting point was the typology of collaborative situations (also known as the time/space matrix), in that visual analytics should support various same place/different place and asynchronous/synchronous settings. However, despite important theoretical work on design considerations for (asynchronous) collaborative visual analytics \cite{128} and work on concrete techniques for collaborative aspects \cite{131,302}, collaboration has not received the warranted attention in visual analytics research. A sentiment further supported by the empirical work within the domains of my PhD project (see Chapter 5). To illustrate my point, I present two meta analyses; (1) a topic modelling analysis with all paper abstracts from the InfoVis and VAST conferences \cite{151}, and (2) a manual analysis based on established topic clusters for the same collection of papers, where I identify collaborative themes in the visual analytics research.

An Automated Analysis of Visual Analytics Research

Topic modeling is a group of machine learning techniques for deriving statistical models of topics within text documents. A topic is a list of keywords along with corresponding probability scores. The probability score captures the importance of a particular keyword to a topic based on frequency and significance. In addition, text documents and topics are assigned membership weights describing how well a given topic matches a document. Two of the most famous algorithms for topic modeling are based on non-negative matrix factorization (NMF) \cite{12} and Latent Dirichlet Allocation (LDA) \cite{36}. I used implementations of these algorithms from the open source python library Scikit-learn, which also provides generic utility methods for text documents such as stop-word removal and feature generation. I conducted the analysis with the open visualization publications dataset presented by Isenberg et al. \cite{151}, which includes 1216 publications for the two conferences up until 2016. Both VAST and InfoVis literate was included, since multiple papers about visual analytics are also published at InfoVis due to its impact factor. By computing models for a varying number of topics, it was
Figure 2.3: Topics over time of the visualization publications dataset [151]. A slice depicts a single topic over time. The height of a slice is proportional to the topic relevance in the particular year. When users hover a topic, the associated list of key words are visualized beneath in order of importance. The selected topic is the white slice, which is the topic that includes collaboration.

apparent that collaboration has a role in the visual analytics literature—in most models there was a topic where collaboration is among the most relevant words. Thus, to assess the overall impact of collaboration throughout time, I developed a visualization for depicting topics over time (see Figure 2.3). The visualization is based on the Themeriver visualization presented by Havre et al. [124] However, my prototype uses color replication in order to scale with an increasing number of topics. A topic is represented as a single river (slice) that increases and decreases in size over time. The height of a slice is proportional to the topic relevance in the given year, i.e. the number of documents where the topic relevance is over a given threshold. When hovering a topic (white) the corresponding weighted keywords are displayed. The main collaboration topic is highlighted with white in Figure 2.3. The visualization shows that collaboration was marginally present as a topic before the VAST conference started in 2005. In the immediate years following the introduction of VAST, collaboration became a stronger focus in the literature. Since then there has basically been a decline, and in recent years collaboration has been a very minor topic at the two conferences. While this experiment includes only the two main conferences for visual analytics research, it still illustrates the limited focus.
A Manual Analysis of Visual Analytics Research

Along with the visualization publications dataset, Isenberg et al. [151] presented topic clusters based on the keywords specified by the authors. These include three topics related to collaboration; **Collaborative Visualization, Storytelling, and Presentation, Production, and Dissemination**, which together consist of 47 papers out of the total 1216. However, 5 can be argued to be miss-classified, e.g. through the word coordination that can also refer to coordinated multiple views—a common visualization technique. I manually identified four main collaborative themes within the list of papers; (1) *synthesis and common ground*, (2) *building blocks for collaborative systems*, (3) *co-located table-top activities*, and (4) *communication of findings*, as well as two papers on collaborative visual analytics theory. Work on common ground mainly deals with creating knowledge graphs (or similar constructs) and combine these from multiple analysts [33, 47, 59, 64, 161, 203]. Thus, there is little work on how to collect and integrate such methods within the visual analytics process model (cf. Figure 2.1). Work on building blocks for collaborative systems mainly deals with individual techniques that are useful when implementing future systems. This include; view synchronization [119, 193], locate analysis work [229], annotate visualization states [302], and collaborative construction of dashboards [214]. Work on table-top activities mainly deals with interaction techniques and visualizations suitable for this exact scenario [37, 147, 204, 293, 296]. Work on communication mainly deals with narrative visualization and storytelling techniques [86, 141, 142, 274], but also with the design of visualizations [50] and collection of interaction history [130] for the purpose of communication.

Outside of these themes, visualization literature in general has dealt with at least two other large themes related to collaboration—**group awareness** and **multiple heterogeneous devices**. Different techniques for supporting group awareness [118] include collaborative brushing [119, 148], history mechanisms and notifications [128], and data coverage representations and widgets [267, 311]. It has been described in various ways how data analysis has moved beyond traditional desktop setups [90, 187, 256]. Thus, significant efforts within visualization has been to bridge multiple heterogeneous devices. This includes techniques for transitioning between coupled and decoupled activities across devices [213], frameworks for distributing visualization interfaces [15, 17, 65, 182], and techniques for utilizing different input devices [135, 158, 188].

While visual analytics research has produced several individual techniques for supporting collaboration, the tool-centric agenda has resulted in a limited focus on integration—such as combining techniques into complete systems or integrating new analysis methods into existing software and organisational contexts. In addition, empirical work within visual analytics research has mostly been restricted to lab-based usability studies, hence it remains a challenge to design such integration. For example, it is currently unclear how to integrate the existing analysis techniques for temporal event sequences with techniques for collaborative sense-making and sharing of partial analysis products and findings.
2.3 Related Work on Collaboration in Data Analytics

In the following, I will present related work on collaboration in data analytics, which I have used to expand the focus of visual analytics research. This includes general aspects of computer-supported cooperative work as well as collaborative aspects of working with data analytics.

2.3.1 A CSCW Perspective

Literature on computer-supported cooperative work (CSCW) has a long tradition for studying collaborative work practices in a variety of domains. Thus, it seems worthwhile to build on this literature when designing technologies that should support and improve existing practices for working with data.

There are relatively few empirical studies of applied data analysis work outside of academia. Chin Jr et al. [61] presented a workshop-based study of intelligence analysts in which they identify how they work, collaborate, and analyze information. For example, the paper describes how the analysts’ workflows are inherently dynamic and ad hoc, which have implications for the design of supporting technologies. Kandel et al. [159] interviewed enterprise data analysts across a variety of sectors. Among other results, they describe three analyst archetypes (hackers, scripters, and application users) and identify recurring challenges for adopting visual analytic tools. Batch and Elmqvist [24] presented a study of data scientists where they identify challenges for using interactive visualizations in the initial phases of exploratory analysis. Choi and Tausczik [62] conducted a study of work practices in analysis of open data and applied the MoCA framework [191] to characterize existing collaborations. They found that most projects were of exploratory nature and collaborations were interdisciplinary, small in scale and with low turnover. Boukhelifa et al. [43] presented a study of how non-data scientists deal with uncertainty when using data analysis methods and propose a new process model for how data workers analyze uncertain data. Recently, Passi and Jackson [236] conducted a study of trust in data science based on ethnographic fieldwork with a corporate data science team. They describe four common tensions in applied data science work when considering different organisational actors and they highlight the importance of negotiation and translation work. While most studies acknowledge or identify the critical role of collaboration in applied data work, only recent studies have begun to consider the impact on organisational actors outside the traditional data analyst [43, 236]. However, especially the work of Passi and Jackson [236] describe the importance of considering such actors when designing technologies to support data work in the real-world. My research builds on these studies by extending the empirical knowledge of how collaborative data work occurs in practice, and how knowledge workers with little knowledge of data science methods integrate elements of data analysis into their existing work practices.
Related literature on knowledge work and data utilization have also highlighted different aspects of working with data in practice. The first aspect is the classical notion of analyzing (working on) data, which is the focus of most visual analytics work as well as most empirical studies as these often focus on data analysts \[62, 159, 236\], e.g.]. This type of work includes applying statistical methods, building visualizations, and generating insights from data. Activities often generate additional data in the form of models, key parameters, or reasoning artifacts. It is to some degree this type of work the visual analytics process model describes (cf. Figure 2.1). The second aspect is working by data as presented by Cabitza and Locoro \[51\]. They describe this type of work as activities that depend on data or where data play an organizing role. Thus, analyzing data is not the main focus, but the main tasks and decisions rely on data or analysis results. Examples from related work include the study by Lee et al. \[192\] of drivers for ride-sharing applications and the study by Irani and Silberman \[140\] of people that work on Mechanical Turk. A large part of this type of work is handing over data or analysis results, either by a human or a system. Thus, it is related to work on coordination mechanisms for general systems design \[271\], such as forms, protocols, and manuals, but with an explicit focus on data. The third aspect is articulation of data work, which is described in several papers as the additional work that as required when multiple actors increasingly rely on data. Examples include the importance of negotiation and translation \[236\], articulation work in healthcare and sensor data usage \[51, 93, 97\], and communication of metadata \[34, 87, 240\]. The latter is in particular important in scenarios where multiple actors with different agendas rely on large shared data repositories. In addition, this type of work is related to previous CSCW work on knowledge- and expertise sharing \[1, 2\]. While the three aspects are indistinguishable in practice, they highlight that working with data is more than merely applying a certain analytic algorithm. My research suggests that is important to consider all three aspects as a whole when trying to integrate visual analytics methods within organisational contexts. Thus, there are additional layers to consider beyond the visual analytics process model (cf. Figure 2.1) when designing systems for collaboration.

2.3.2 Replicability and Shareability

Being able to replicate and share work are important aspects of collaboration in data analytics—both to review and to reuse work. Consequently, much literate deals with recording data analysis history, i.e. provenance tracking \[78, 252\]. When talking about data analytics in general, provenance tracking includes both interaction histories in visual interfaces as well as output from executable scripts. GRASPARC was an early system for tracking problem solving tasks with both numerical and graphical components \[48\]. The goal was to record the search process used by a scientist in reaching the optimal solution. Later, the approach was extending to also allow users to append annotations to parts of the search hierarchy \[279\]. VisTrails \[27\] was an early provenance management system for
2.3. RELATED WORK ON COLLABORATION IN DATA ANALYTICS

scientific visualization and analysis workflows and Heer et al. [130] extended the focus of provenance tracking to also include interaction histories in visual interfaces. Additional work includes Burrito [117] for capturing low-level computational activities and the work of Gotz and Zhou [110] that proposes ideas for capturing high-level semantic meanings from low-level user events. While provenance tracking methods usually work automatically in the background, related work has also identified the need for simple and lightweight tracking of alternatives in exploratory data analysis, such as inline versioning for programming environments [164]. Recently, Head et al. [125] presented a method for gathering the code that produced a certain output in messy computational notebooks. I will return to computational notebooks in Section 2.4.2.

As presented by many, exploratory data analysis is iterative, branching, and non-linear in nature and often involves multiple cycles of cleaning, modeling, visualizing, and interpreting data [241, 294, 299]. Thus, provenance tracking methods often produce large and complex graphs which after the fact can be difficult to make sense of. In addition, tracking the full interaction history may pollute the provenance graph with unnecessary information, as users may simply have investigated the functionalities of an interface without actually doing analysis. While work has proposed automatic methods for inferring semantic meaning from user interaction [110], it remains a challenge to collect the motivation and reasoning behind each step. Therefore, being able to replicate data analysis results does not necessarily equate to effective sharing. A sentiment repeatedly mentioned in various ways in related work [159, 203, 262, 276], and exemplified by the additional articulation work that is often required (see prior section).

2.3.3 Common Ground and Explainability via Storytelling

Another important element of collaboration in data analytics is effective communication during and after the analysis process [69, 128], which is closely related to the ability of sharing and explaining analytic work. A prerequisite for successful communication is the establishment of common ground [67], which has been described as the generation of mutual knowledge, mutual beliefs, and mutual assumptions. Therefore, a large part of the CSCW literature has been dedicated to the theory of boundary objects. Boundary objects are artifacts which “have different meanings in different social worlds, but their structure is common enough to make them recognizable” [284]. Establishing boundary objects can therefore be thought of as way to establish (or support) common ground between different actors. However, a criticism of the concept is that the artifacts we call boundary objects often does not satisfy the informational requirements for each community of practice [190]. This has led to the notion of boundary negotiating artifacts [190], which describe the kinds of artifacts used to transmit information in non-routine collaborative processes. Both concepts can be important to data utilization as this type of work often oscillates between routine and chaos (cf. Section 5). Furthermore, to support the increased need for articulation
work, creating artifacts that serve as boundary (and/or negotiating) objects can potentially ease such communication. However, it is not straightforward to define what kinds of artifacts establish common ground. It has been suggested that data itself rarely meet the requirements to constitute a boundary (negotiating) object. At best, shared repositories can function as boundary negotiating artifacts as the formats and content of these are often negotiated between multiple actors. Still, data rarely sit in ready-as-is formats or repositories and often require significant rework for the intended use scenario.

In the visualization community, communication of findings is often addressed through the concept of data-driven storytelling. Here textual descriptions and rich media are combined with interactive visualizations, often in a sequence of steps that constitute a linear story. The strength behind data-driven storytelling is twofold; (1) it allows the author to combine different external representations with information suitable for a specific audience, which provides a guided way of understanding complex analysis results, and (2) the interactive visualizations afford active reading, which means that readers can explore the content and not merely consume static information. Thus, utilizing data-driven storytelling to establish common ground and support explainability in data analytics can be a way to facilitate collaboration between people with different backgrounds and competences. Therefore, recent advances have led to several approaches for sharing data analysis results by embedding data-driven exploratory elements in existing media. This includes the concept of live documents, web-based explorable explanations, active ink for externalizing insights during analysis, and embedded explorable multiverse analyses in research papers. In this dissertation, I have explored how to further integrate elements of data-driven storytelling in the analytic process to support collaborative workflows that allows the sharing of findings, explanations, and motivations at any stage of an analysis. Among other things, this work extends prior work about capturing visualization state along with textual annotations and forming narratives from analytical provenance by embedding these ideas in an open-ended computational environment.

2.4 Inspirational Technologies

There are several related technologies that already possess some of the requirements presented in the previous section. In the following, I will outline important academic and commercial tools for working with visualization as well as tools that represent the literate computing paradigm—tools which have been influential to the work presented in this dissertation.

2.4.1 Visualization Toolkits

Numerous tools have been developed to ease the creation of interactive visualizations. Most famous is the Tableau suite that promotes a drag-and-drop approach
in everything from data preparation, to visualization, and storytelling [290]. The iVoLVER toolkit [215] supports a visual programming approach to extract data from existing visualizations and reconstruct this data into new ones. Recent tools, such as Data Illustrator [201] and DataInk [324], have also started to focus on highly customizable and artistic visuals. In the other end of the spectrum are tools like Keshif [325] and Voyager [317] that allow users to quickly browse a myriad of visualizations for fast exploration of new data. While the core focus of the aforementioned tools is on supporting non-programmers, there has also been focus on supporting typical programming workflows. A popular approach has been to utilize declarative specifications to define visualizations and interactions, i.e. simple specifications encoded in e.g. JSON. Examples include Vega [268], Vega-lite [270], and Atom [234]. In addition, actual programming libraries have also been developed. One of the most used examples is the JavaScript library D3 [41, 42] that supports the creation of web-based SVG visualizations. A common challenge with these tools is integration as users often need to move from one tool to another when switching focus. Even in the Tableau suite, data preparation, analysis, and presentation are separate stages. For example, there is no support for note taking during the analysis which then later can be used to create a presentation. In addition, the current toolkits force collaborators to negotiate a common set of tools that everybody can use in order to contribute to the shared work. For example, it is cumbersome to alternate between developing statistical models in Python and visualizing the results in Tableau, and this type of workflow does not match the general visual analytics model (cf. Figure 2.1). These challenges are among the focus of the later work in this dissertation (cf. Chapter 6).

A recent trend in general data analytics tools has been to support external representations of the analysis flow or analysis steps. The data science tool Rapidminer [253] and the Tableau preparation tool [291] both adopt a pipeline-based approach for depicting the data processing and analysis flow. The data wrangling tool Trifacta [298] present a series of wrangling steps as a recipe that can be applied to a larger dataset or shared with collaborators. These approaches are arguably supporting collaboration in that the visual representations can be used to establish common ground. Still, these tools present the same challenge of integrating tailored communication into the analysis workflow.

2.4.2 Literate Computing

At this point in history, it is inevitable to include a discussion of the popular paradigm literate computing [217] when talking about data science and collaboration. An entire conference is dedicated to the computational notebook Jupyter [170] and entire organisations are starting to adopt computational notebooks as a standard tool in their software ecologies [300]. Literate computing is a descendant of the literate programming paradigm introduced by Donald E. Knuth [171], with the motivation to not only write programs for the sake of the computer but also to support human understanding. Part
of the idea was to write a narrative as an intertwined part of the programming process. Knuth suggested doing this by having integrated documents with both code and narrative that subsequently could be *weaved* to typeset text appropriate for humans or *tangled* to machine executable code. Besides mixing narrative and code, an important part of literate programming was to create a hierarchical structure within the integrated document to support and guide the understanding of the program. The descendants of literate programming are evidence of the singular way narration and textual descriptions can support human understanding of complex phenomena [217, 321]. Literate computing [217] has extended literate programming by integrating narrative and executable code within the same document. This allows authors to not only document functionality but also to comment on the output of the individual program parts, which has proven to be particularly well suited for data analysis. Thus, several computational notebooks have been developed to realize the concept of literate computing [108, 170, 231, 250].

Notebooks already present several qualities suitable for collaboration such as elements of data-driven communication and replicability of data work. Thus, recent efforts have focused on enhancing their functionality; including collaborative editing [108, 250] and static programming analysis to reasoning about execution order [231]. However, it has also been described how the linear structure of computational notebooks has certain limitations [125, 165, 261, 262]. For example, there exists a tension between exploration and presentation [165, 262], which means authors frequently have to delete parts of the notebook when preparing it for presentation—parts that may later be useful. I.e., it can be difficult to support collaborative workflows with notebooks that oscillates between generating and presenting analysis parts. In my work, I address these challenges, e.g., by combining literate computing with the hierarchical structure of literate programming (cf. Chapter 6).
3 | Research Approach

The work presented in this dissertation can be grouped in three overall stages. First, my project dealt with the specific data from the presented domains and analytical tasks that the data could support (cf. Chapter 4). This work followed a standard visual analytics approach—trying to fulfill the model shown in Figure 2.1. Second, my project took an empirical focus to further develop our understanding of collaboration in general data work (cf. Chapter 5). The empirical grounding together with related theory and technologies resulted in potential design implications for how visual analytics can be integrated in an organisational context. Third, my work focused on designing and constructing systems that have the potential to fulfill the visions set out in the empirical analysis (cf. Chapter 6). This resulted in two concrete interactive systems that; (1) further our understanding of how visual analytics can be utilized in larger collaborative contexts and (2) serve as concrete platforms upon which new research can be build. In the following, I will elaborate on the research approach employed throughout my PhD project.

3.1 Experimental Computer Science: A Data Centric Approach

The main research approach of this dissertation is experimental computer science as presented by Grønbæk [115] (cf. Figure 3.1). By now, it has been the foundation for several PhD dissertations prior to this one [151, 226, 236]. Thus, the approach has been a successful model for generating research results in the intersection between theory and the real-world. While the model presents a general and high-level overview of research, it captures that the main contributions in my line of research is the design and construction of novel interactive technology that enable people to do things they were otherwise unable to.

Experimental computer science is grounded in a dual understanding of state-of-the-art technologies and challenges from real-world domains. The deep knowledge of both existing research and the domain allows for the identification of important research questions and the design of new technologies. Hence, the desired outcome of this type of research is to arrive at new understandings, advance state-of-the-art technologies, and produce general methods that transcend individual domains. The outcomes are not based on theoretical proofs, but rather they are approximate and of qualitative nature. To achieve this, the overall approach proceeds through iterations of analysis, design, prototyping, and evaluation. Thus, a
core part of the approach is to develop prototypes and build interactive systems. Early prototypes can be vehicles for improved understanding and interactive tools can show how and which conceptual ideas are in fact realizable. The continued analysis of both current research, the domains, and prototypes have been instrumental to the development of this research project. While the initial prototypes solve concrete challenges for analyzing specific types of data (cf. Chapter 4), they also provided necessary understanding that have inspired the later research questions of this dissertation (cf. Section 1.1).

Unsurprisingly, data have had a central role both for the understanding of the domains and in the evaluation of the presented analysis methods. I have spent substantial time analyzing the data from both Grundfos and the DBA. Besides illuminating concrete challenges, this process has also been important for having concrete dialogues with domain experts. Interviews, observations, and workshops have been utilized in combination with my understanding of the data to elicit how data can be used, which tasks need support, and later to suggest how tools can be integrated in existing work practices. For example, we have had frequent workshops as well as interviews with employees at the DBA to understand the current work processes, identify recurring pains, and to propose potential solutions. Similarly, workshops, meetings, and hackathons at Grundfos have also improved our understanding of data analytics in an organizational context. In addition, the engagement in multiple domains has confirmed that several of the real-world challenges surpass the individual domains—further validating their importance.

The grounding in both current research and real-world challenges has led to the design and prototyping of the novel analysis methods and systems presented in this dissertation. While each individual project has gone through multiple iterations, e.g. to improve the analysis methods, the iterations of my research project as a whole has also led to the design of the latter systems (cf. Chapter 6). Concretely, the first iterations of designing methods and tools for specific data
and tasks allowed us to identify that it requires are larger focus on collaboration to support the adoption of data analytics within an organisational context. This prompted us to expand the focus of our investigations as described in Chapter 5.

There are several aspects to the evaluation of visual analytics research. In Illuminating the Path [69], the authors describe three levels of evaluation. The first level is evaluation of individual components. This involves evaluation of both analytic algorithms, visual representations, interaction techniques, and interface designs. Thus, evaluation at this level involves both computed results such as accuracy and empirical studies about efficiency and effectiveness. The second level is evaluation of systems which involve increasingly complex tasks. Metrics for this type of evaluation include utility and learnability, hence evaluation at this level is already a challenging task. The third level is evaluation of work environment adoption, which include elements such as productivity and satisfaction. The third level of evaluation often requires some type of deployment of the systems in real-world scenarios, thus these types of evaluations are the most challenging to conduct. In particular, such evaluations often require extended use to produce meaningful results. The analyses of the current research literature (cf. Section 2.2.2) highlighted that evaluation in visual analytics research mainly address the first level and occasionally the second level. Often, evaluation in visual analytics and information visualization consist of small-scale lab studies aiming to prove that a tool is usable for a particular purpose. Thus, current evaluations mostly focus on simple tasks that does not reflect the complexity or longevity of most real-world data analytics tasks [69]. In addition, visual analytics research rarely provides analytical evidence for the correctness of the analysis results beyond confirmation from the selected participants. Thus, visual analytics research can also benefit from more evaluations that use real-world data to illustrate the usefulness of different methods.

Data and ground truth have therefore been central to the evaluation methodology in the first part of my PhD project. The analysis methods and visual analytics tools for specific data and tasks have been evaluated at the component level with ground truth data as in most standard data mining and machine learning research (i.e. to argue about correctness and accuracy)—i.e. I have shown that the proposed methods produce valid results. The employed evaluation metrics include unsupervised measures, such as the silhouette coefficient [260] and visual complexity measures [219], as well as supervised measures, such as the V-measure [260] and entropy-based information gain [247]. While machine learning methods are often evaluated using randomized cross-validation [314], this approach is not suitable for time dependent data. Thus, in the evaluation of the temporal event sequence methods, I used a simple hold-out method based on time to evaluate prediction accuracy [314]. However, presenting the first prototypes to domain experts confirmed that it is increasingly challenging to conduct usability studies of tools that drastically change the way current data work occurs. The second part of my PhD project has therefore been centered around the design of collaborative visual analytics systems that have the potential to eventually enable...
researchers to address the higher-level evaluation goals such as evaluations on the third level—evaluations which are currently almost non-existing. In the following, I will elaborate on the research approach in the second part of my PhD project.

### 3.2 Field Work and System Design

As mentioned, the first visual analytics prototypes led to an improved understanding of the domains and highlighted relevant problems for both research and industry. Thus, the early development prompted our investigation into integration and collaboration aspects of visual analytics. To develop a deeper understanding and collect additional empirical evidence about existing collaborative practices and challenges in data work, we conducted an interview study with employees at the DBA. The study allowed us to more formally get multiple perspectives on data utilization within an organizational context and how collaboration was used to mitigate and solve challenges related to data work. The main outcome of the interview study was potential design implications for future systems—implications which we found had been under-investigated within the research literature. Hence, contributing with concrete design concepts and constructing interactive technologies was an obvious method for pushing the boundaries of how data analytics can unfold. While system design in research can involve high uncertainty as this type of research often propose new ways to work, this type of research also has the potential to achieve large gains. As such, constructing new interactive systems for tool integration and further involvement of people in visual analytics presented an opportunity to make contributions that go beyond visualization designs for specific use scenarios—a common focus in much related work.

The interview study was of semi-structured format and conducted in context at the DBA. We employed an inductive approach to analyze the data consisting of several iterations with coding, identifying common categories and themes, comparing results between authors, and re-analyzing the empirical data with the proposed themes. This process allowed us to identify key concepts representing the participants shared experiences with data work. This process is similar to the meaning condensation presented by Kvale [178] and inspired by interpretative phenomenological analysis from psychology [145], where systematic analysis in a bottom-up approach is used to derive meaning from participants experiences. The coding and grouping of categories are similar to the three coding methods of Grounded Theory [70]—open coding, axial coding, and selective coding. But in our case, the analysis did not start until after the completion of all interviews. This is different from the general approach in Grounded Theory, where the analysis starts already after the initial data collection and findings are then used to structure the following data collection. As these types of analyses are interpretive, and thus subjective, the results are not solid facts. However, it is a way to explore an area of investigation without (explicitly) imposing any hypotheses or prior assumptions. Still, the subjective nature means that any results should
be interpreted as such. As interview studies rely on the recollection of the participants, findings are also biased thereof. Ideally, we would have been able to combine interviews with observations of the participants during their everyday work with data—as in the contextual inquiry approach [32]. However, due to several confidential aspects this was not an option. Furthermore, the kinds of activities we were interested unfold over long periods of time in unpredictable ways, which means that doing observations would not guarantee any insights. I further discuss limitations of the interview study in Section 5.2.4.

An important part of my work has been to consolidate any findings with related work to further ground potential implications and identify opportunities for new technology. The approach employed in the latter part of my PhD can best be described as constructive problem-solving based on an empirical grounding [233]. By viewing HCI as problem-solving, we are forced to think about whether research contributions solve significant problems and whether solutions increase our problem-solving capacity [233]. The proposed systems in this dissertation combine concepts, theories, and methods in a way that link empirical and constructive work—i.e. they propose new ways to construct interactive technology for collaborative visual analytics. Thus, the presented systems "advance our ability to solve important problems relevant to human use of computers" [233], which is the essence of viewing HCI as problem-solving. In many ways, the employed research approach also resembles action design research, as recently described in the context of visualization design [212]. For example, in how the principles of practice-inspired research and theory-ingrained artifact have grounded my work, and in how continuous reflection allowed us to identify and shape design concepts that can be useful beyond the context of the presented systems. Still, evaluating systems research is challenging, as described by Olsen Jr [232]. While Olsen Jr [232] discuss user interface systems research specifically, several of the insights apply to the systems research presented in this dissertation. This include common evaluation errors such as the usability trap and the fatal flaw fallacy. For example, systems rarely conform to the “walk up and use” and the standardized task assumptions required to do usability testing. In addition, the scale and complexity of the relevant tasks often go beyond what is possible within a controlled study. Instead, Olsen Jr [232] argues that systems research should clearly state the situation, tasks, and users it addresses, and among other demonstrate importance, generality, and reduction to solution viscosity. Therefore, evaluation of systems research can also include analytic arguments about importance and demonstrations of generality. These aspects resemble evaluation criteria from viewing HCI as problem-solving [233]—e.g. to judge significance and transferability. Thus, we have spent substantial time on developing use scenarios (fictive and real-world based) as well as demonstrating the use of the proposed systems with different types of data and for diverse tasks. We also conducted a small-scale expert review as these are suitable for complex and high-level tasks [297]. Still, the best evaluation of systems research is through real use by real users in real contexts. Due to the challenges of designing meaningful user studies for complex tasks that unfold
over longer periods of times, exploring extended use of the presented systems has been infeasible within the time frame of this PhD project. Nonetheless, it is a goal of my future work to conduct such investigations—maybe by extending the expert review to include industry personnel (cf. Section 6.2.3). For now, all systems are available online. This way, other researchers can extend the code-base and use the systems for their own research, which in itself is a form of exposure to real users.
4 | Designing for Data & Task

The first research question presented in Section 1.4 has led to the development of prototype tools that; (1) address challenging data and tasks from the presented domains, (2) follow the visual analytics process model, and (3) allow domains experts to utilize their knowledge for data analysis. Besides working with the real-world data of the domains, this work has also allowed us to have more concrete dialogues and workshops about how data should be utilized. Thus, while the prototypes present contributions in their own right with novel combinations of machine learning and visualization techniques, they have also been instrumental to the understanding of the domains.

In the following, I will present the two prototypes that have resulted in poster F and paper A, respectively. Besides iterations of these two prototypes, I have also been involved in the development of a prototype for product modularization and production optimization at Grundfos. The prototype was essentially an advanced calculator with an underlying cost model that stakeholders and decision makers through a visual interface could use for experimenting with different production scenarios. The main success of this prototype was to convince stakeholders to initiate a new project internally, which meant the continued development of the prototype was put on hold. Thus, the prototype did not lead to any significant contribution within research. The general task type that the following prototypes address is to find relevant patterns within large quantities of data in order to identify important or interesting instances. This include companies with suspicious behavior, pumps with certain characteristics that, e.g., suggest failure or suitability for a certain environment, and patients that have a higher risk of failed treatment. The prototypes therefore mainly address challenges C1 and C2 as presented in Section 1.4. I will conclude this chapter with reflections on the intended usage of the prototypes and how the development processes inspired the continued work of this dissertation.

4.1 Prototype 1: High-Dimensional Data

As described in Section 2.1.2, understanding correlations in high-dimensional data is inherently difficult for humans, especially since visualizing more than two or three dimensions limits our ability to instantaneously recognize relevant patterns. Analyzing and understanding this type of complex data has therefore been the focus of much related work—and as previously described, it is also highly relevant to the domains presented here. Thus, this work contributes with a unique way
of utilizing domain knowledge to evaluate several clustering results and this way identify the most relevant patterns.

At the DBA, we found that the personnel base most of their investigations on previous experiences. For example, cases with non-compliant behavior is collected and subsequently used to manually maintain a list of risk factors. The available examples of companies or persons can be thought of as a partial labeling, which can be used for semi-supervised learning—a particular class of machine learning techniques. In other words, one of their tasks can be described as; given a set of examples, find common parameters that describe these examples and thereby find additional examples. This type of task is typical for many domains including production and healthcare. For example, when defect pumps are returned from clients or the outcomes of a subset of the patients have been determined. However, a partial labeling provided by domain experts may be inaccurate or not match the patterns found in the data. Concretely, we found that such a labeling often does not only suffer from being incomplete, but that distinct subgroups within a single label may also match different patterns in the data. Thus, it would be more beneficial to have sub-labels, both to generate better models and to subsequently make sense of the results. In addition, it is challenging for domain experts to input a partial labeling to a machine learning algorithm and inspect the impact when being unable to program.

As exploratory analysis requires several iterations and experiments it is important to maintain interactivity, user attention, and provide a fast feedback loop. The following prototype has therefore been developed with a simple and fast machine learning approach that can indicate relevant patterns. Afterwards, stronger and more advanced methods can be used.

### 4.1.1 Design & Workflow

To exploit domain knowledge in the form of a partial labeling we propose an analytical process consisting of three steps: (1) define labeling and input variables, (2) generate and rank multiple clustering models, and (3) inspect results and repeat. The prototype is based on conventional methods for visualizing high-dimensional data; parallel coordinates [100, 144, 227] for the multidimensional features space and scatter plots for the reduced feature space. Figure 4.1 depicts the web-based prototype with two coordinated views that displays one of the potential clustering results. As described in poster F, the prototype is implemented as follows:

1) **Input:** The user provides labels using brushing [31, 132] in the parallel coordinates visualization, which then constitutes a binary distinction. The instances of the combined selection are one group and the remaining instances is the other group. The combined selection can for instance be companies with a known record and large turnover. The user can furthermore choose to limit the feature space by selecting only those features of interest to the current analysis.
Figure 4.1: The visual analytics prototype for high-dimensional data presented in poster F. The tool enhances the parallel coordinates visualization (c) with clustering functionality. Users can select features (a) and through brushing provide a partial labeling (b) that describes instances of interest. Subsequently, users can inspect the best results (d) using the coordinated views (c) and (e).

(2) **Cluster:** A two-round clustering approach is used with the binary distinction defined by the user. In the first round, clustering is performed on each of the two groups. In this round, the silhouette coefficient \[ \text{silhouette coefficient} \] (an unsupervised score) is used to reason about the structural properties of the clusters to find the optimal number. The result of the first round is a sub-labeling of the two groups. In the second round, clustering is performed on the combined dataset. In this round, different combinations of the sub-labels found in the first round together with the V-measure \[ \text{V-measure} \] (a supervised score) is used to find the optimal parameters. While our method is not specific to a single clustering algorithm, we use the K-means clustering algorithm \[ \text{K-means} \] in the prototype due to its speed. In the second round, clustering models is computed for different number of clusters and different combinations of the input features. The best results are continuously reported to the user to maintain attention.

(3) **Inspect:** The clustering results is presented to the user as an axis in the parallel coordinates visualization. Additionally, the result is color-coded in the PCA \[ \text{PCA} \] based scatter plot. The views are coordinated, so users can update both views either by hovering the scatter plot or by creating filters in the parallel coordinates visualization.

To indicate the usefulness of the method, we first used the popular Iris data set \[ \text{Iris data set} \] consisting of 3 types of flowers. However, most standard clustering algorithms will yield a total of 2 clusters since two of the classes are located near each other within the feature space. But, if an expert can provide a partial labeling that separates parts of the two similar classes, our approach will compute 3 clusters.

4.1.2 **Task: Finding Suspicious Companies**

To illustrate a use scenario for the tool, I used data from the DBA and the task of identifying fraudulent or otherwise troublesome companies. As previously described, current investigations are based on whether companies look suspicious
based on risk factors derived from past experiences and known cases. These investigations often use historical registration data, such as registrations of board members and employment counts, or financial data. By converting the registration data into features, like the number of occurrences for each registration type, the tool can in theory be attached directly to the database. To account for the lifespan of a company, it can be beneficial to normalize the features based on the first and last occurrence. The screenshot of the prototype in Figure 4.1 depicts results from using data with the companies in Denmark that have the most registration updates. The results shown in Figure 4.1 are based on a partial labeling where all companies that have a status different from normal are used as a single class, since the status does not necessarily capture why a company was forced to close. A potential insight from this example is that if a company changes name more frequently than business type and legal district, they are within a cluster where 100/202 of the companies have stopped. Thus, the 102 remaining companies within that cluster are more suspicious than a random one out of all the 3836 normal companies and therefore these warrants further investigation. Using the tool, we were also able without any prior knowledge to identify suspicious companies that personnel at the DBA had classified with minor infringements.

4.2 Prototype 2: Temporal Event Sequences

As described in Section 1.4, gaining useful insights from temporal event sequences is challenging. Figure 4.2(b) depicts how volume and variety makes it difficult to effectively aggregate multiple event sequences, which then severely limit humans’ ability to distinguish good from bad event flows. In essence, the contributions of paper A enables the visualization shown in Figure 4.2(c). This includes; (1) the concept of composite event learning that allows event sequences to be aggregated more effectively, and (2) the event flow visualization combined with sequence outcome probability encoded in the transitions between events. The later allows domain experts to identify critical and normal event flows. In addition, paper A presents a prototype supporting the general visual analytics process model (cf. Figure 2.1) and an evaluation with company event data indicating the usefulness of the approach. The prototype allows domain experts to inspect the computed patterns as well as adjust input parameters, including event types, number of composite events and the size of time windows. Furthermore, domain experts are supported in interpreting the computed event flows with appropriate linked visualizations of cluster centers and time as well as indications of prediction potential. In the following, I will present the underlying concepts of composite event learning and sequence aggregation, the prototype, and the evaluation with company event data from the DBA.
Figure 4.2: The event tree visualization method (a) used in paper A. The y-axis represents sequence quantity, the x-axis represents time, and colored bars depict the different event types. In (b), the same visualization shows a large collection of company event sequences without any simplification, and in (c), the visualizations shows the same collection but with composite event learning and outcome percentages (grey scale).

4.2.1 Composite Event Learning

To automatically reduce the volume and variety of temporal event sequences, we introduce the concept of composite event learning. The idea behind composite event learning is to combine several atomic events into high-level events. In our approach, this is done by replacing sub-sequences in the original data with new inferred event types—new event types which subsequently need interpretation. An
example from the DBA is the start of a new company: New companies usually have similar startup processes, which often can be replaced with a single high-level event type consisting of the addition of several board members and accountants as well as updates to core company information like name and business types. The combination of atomic events into high-level events have previously been described as a useful strategy for analyzing event sequences [85], but composite event learning is to our knowledge the first method for inferring high-level events with computational methods.

Inspired by common methods from activity recognition, we employ a simple bucketing by time period approach to composite event learning. The event sequences are divided into time segments of equal size that each will be associated with a composite event type using conventional clustering methods. The features in our initial approach consist of simply counting event type occurrences within each segment (and potentially attribute occurrences). However, our method leaves the opportunity to engineer additional and more advanced features, such as different timespans. As described in Paper A, for time window size $w$ and number of clusters $k$ we do the following:

1. **Segmentation:** Divide each temporal event sequence into segments of equal time window size $w$. Alternatively, the segmentation can be based on number of events.

2. **Feature Generation:** Compute features based on event type occurrences for each segment, effectually ignoring sequence order within the window segments.

3. **Clustering:** Partition all segments into $k$ clusters using the k-means clustering algorithm [13]. The $k$ clusters constitute the composite event types.

The proposed method for composite event learning introduces the challenge of finding the optimal time window size $w$ and number of clusters $k$. In the presented use case, this includes an event tree visualization with good separation of the chosen sequence outcomes and good predictive power. For example, if a too large $w$ and a too small $k$ are used, the composite events will likely represent an oversimplification yielding an event tree visualization with low outcome separation. Contrarily, if a too small $w$ and a too large $k$ are used, the composite events will likely be very similar to the original events and have little usefulness.

4.2.2 Sequence Aggregation & Quality Metrics

To enable effective use of composite events and the visualization shown in Figure [4.2](c), we rely on sequence outcomes and aggregation methods as well as quality metrics for the resulting event tree. An outcome is simply a special event occurrence. This can be an existing atomic event or a series of events. In the DBA domain, the most relevant outcomes include bankruptcy and different types
of fraud. Given a single outcome, the event sequences can be divided into two
groups—the sequences that include the outcome and those that do not. While
the analysis scenario in our work is based on this binary distinction, the presented
methods can also be extended to multi-class or numeric outcomes.

To compare the concept of composite event learning with state-of-the-art
methods for temporal event sequence visualization, we included three different
methods for aggregation in our work—a simple aggregation method used with
the composite events and two advanced methods that try to infer the most use-
ful atomic events to include in the event tree. As described in Paper A, the
aggregation methods are the following:

**Simplified Aggregation (SA):** The sequences are simplified with composite
event learning. Sequences will follow the same path in the data structure
until they no longer consist of the same series of events, which will introduce
a branch in the data structure. Thus, the hierarchical structure will contain
a path for each unique event sequence. The number of sequences and event
timestamps are maintained in each node of the tree.

**Most Common Pattern (MCP [199]):** The Rank-Divide-Trim algorithm pre-
sented in [199] recursively ranks all events w.r.t. frequency (the number of
sequences an event occurs in), divides the sequences based on the highest-
ranking event type, and then trims the sequences up to the first occurrences
of the chosen event. These events are then included in the resulting hierar-
chical structure, which means only the most important events are included.

**Most Separating Pattern (MSP):** This algorithm is based on the same rank-
divide-trim method as the MCP algorithm but with a different ranking
function. In essence, it is a new aggregation method we developed that
utilizes sequence outcome to extract relevant events. By using entropy-
based information gain with the sequence outcomes to rank the events,
the resulting hierarchy will include the most separating events w.r.t. the
outcome. Thus, the outcome is used to infer the most important events
with a similar approach used to construct decision trees.

Sequence outcomes are not only useful for visually inferring interesting pat-
terns, but also to compute quality metrics of the generated event trees. For this
we use entropy-based information gain for the splits at a certain level in hierar-
chy. Information gain basically describes how well a split separates the sequence
outcomes. Furthermore, we also use two metrics describing visual complexity pre-
presented by Monroe et al. [219], as high information gain alone does not necessarily
imply good generalization beyond the dataset. For example, the simple split into
raw event sequences, i.e. multiple samples of size 1, will give the maximum in-
formation gain. Lowering the visual complexity is therefore important for both
interpretability as well as generalization purposes. Visual complexity is similar
to the generalization heuristics of decision trees, where smaller trees should in
theory generalize better.
4.2.3 Design & Workflow

The prototype tool presented in paper A visualizes probabilistic event flows of large data collections using any of the aggregation methods presented above. For the SA method, users can define the segmentation time window size ($w$), the number of composite event types ($k$), and which atomic event types to include (Figure 4.3(f)). The prototype has two central views, one for the event tree, visualizing also outcome probabilities at the event transitions, (cf. Figure 4.3(a, b)) and one to investigate the composite events (cf. Figure 4.3(c, d)). The hierarchical event flow visualization (cf. Figure 4.3(a)) is based on a visualization presented by Wongsuphasawat et al. [319]. However, we have extended the method to also include outcome percentages encoded in each event transition. Time is encoded on the x-axis (see Figure 4.2(c)) and percentage of the total number of sequences is encoded on the y-axis. Colored vertical elements represent a specific type of event or a composite event. For example, the large vertical dark gray element shows that almost 60% of the sequences begin with composite event 1 and then approximately 50% of the sequences continue with composite event 0 or 9. Chosen composite events can be inspected using the aster charts (Figure 4.3(c)), i.e. the color of the vertical element corresponds to the color of the composite event number. Composite event inspection will further be elaborated in the following section. Outcome percentages are encoded with a black and white scale in the event transitions, where completely white corresponds to 0% and completely
black corresponds to 100% (Figure 4.3(e)). This way users can identify the relevant event (good) flows from the (bad) flows, and reason about when certain flows start to become critical. The current prototype can also report on the future outcome if the data is split into two parts—before and after a certain date. This allows the analyst to conduct experiments for assessing how well certain patterns can be used for prediction.

4.2.4 Composite Event Inspection for Explainability

To support explainability of the derived models, it is important to not only understand the event flows but also the composite events. As mentioned, the composite event types can be inspected using the aster charts (Figure 4.3(c)) and the event type legend (Figure 4.3(d)). The aster chart method contributes with a new way of visualizing cluster centers, i.e., each aster chart represents a composite event type. A slice represents an original atomic event type, which can be identified using the color encoding and the event type legend. The width of a slice encodes the feature mean (i.e. average number of event occurrences), proportional to the sum of all feature means within the given cluster. This way the width encodes how important an atomic event type is in a cluster compared to the other event types. The height of a slice encodes the feature mean proportional to the means of the same event types in the other clusters. This way the height encodes whether there exists other clusters where a given event type is more prevalent. This compact visualization approach of the cluster centers is especially suitable in this scenario as several of the feature means (event counts) are zero. This means that the event types that do not occur in a cluster are not taking up visual space. In addition, we combine features with very low means into one category to make the circle charts even simpler. This visualization approach allows users to quickly identify common or distinctive characteristics of the composite events by looking at both color, height, and width of the slices.

4.2.5 Evaluation with Company Event Data

We conducted a comparative evaluation of the different aggregation methods presented in Section 4.2.2 on three different datasets from the DBA with the proposed quality metrics. Statistics for the chosen datasets are shown in Table 4.1. We only use data prior to 2014 in general, which allows us to reason about how extracted patterns generalize to the following years. For reference, Table 4.1 also shows the chance of picking a company at random that goes bankrupt after 2014. The table in Figure 4.4 (top) shows the evaluation results for the three different aggregation methods on dataset 1 with a minimum support of 1% for the subgroups. Similar results were achieved with the other two datasets. Unsurprisingly, the MCP method provides the least information gain, as there is no guarantee that the most common event flows correlates with sequence outcome. Still, the MCP method is the best method for reducing visual complexity when measured using
Table 4.1: Dataset characteristics

<table>
<thead>
<tr>
<th>Business type</th>
<th>Dataset 1</th>
<th>Dataset 2</th>
<th>Dataset 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of sequences</td>
<td>IT-Service 3475</td>
<td>Building Contractors 4008</td>
<td>Retail 25122</td>
</tr>
<tr>
<td>No. of events</td>
<td>30494</td>
<td>41151</td>
<td>222138</td>
</tr>
<tr>
<td>Future pred. prec. (%)</td>
<td>3.5</td>
<td>2.6</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 4.4: Quality metrics (top) for the three different aggregation methods on datasets 1 and prediction precision for the period 2014-2017 (bottom) on all datasets.

average height of the visual elements. The MSP and SA methods provide similar information gains despite that the MSP method is using the outcome to generate the aggregation and the SA method is completely unsupervised. In addition, the SA method computes the fewest visual elements in the event tree. However, these elements are composite events that need further investigation after relevant event flows have been identified. Thus, the SA method is therefore not necessarily superior in terms of explainability, hence whether the SA or the MSP method is superior likely depends on the use scenario.

4.2.6 Task: Identifying Critical Event Flows

The overall task with the tool is to identify critical event flows and ideally event flows that can be used to predict future outcomes. This, besides summarizing how companies have behaved so far, the prototype tool also allows the analyst to conduct experiments for reasoning about prediction potential. The most troublesome event paths shown in Figure 4.3 have at least 30 % closed companies. If
an analyst employs this heuristic to narrow down the search for future troublesome companies, i.e. companies that go through a sub-sequence where at least 30% have the negative outcome and with 1% minimum support, the resulting prediction potential is presented in Figure 4.3 (bottom) for all three aggregation methods on the three datasets. The visualization shows the chance of choosing a company from the current normal population that will go bankrupt in the future, i.e. prediction precision for the period 2014-2017. Both the MSP and SA methods allow the analyst to identify larger interesting subgroups with higher prediction precision compared to the MCP method—with the SA method being slightly better despite the similar information gain of the two methods.

4.3 Reflections on the Analysis Methods

While both prototypes present novel ways for domain experts to analyze and use existing data, they also present very different ways of working compared to existing workflows. Thus, the proposed prototypes still present barriers for adoption even though the challenges C1 and C2 were specifically addressed. In the following, I will first present reflections on the concept of composite event learning and then on tool adoption for domain experts.

4.3.1 Reflections on Composite Event Learning

The presented evaluation has shown that combining events into composite events prior to sequence aggregation can provide a better separation of temporal event sequences w.r.t. sequence outcome in the domain of business data. The results suggest that the method can be used for initial exploration and separation of the sequences, i.e. the companies, such that guesses about future outcomes are more informed and significantly better than random guessing. Even if an event sequence dataset is not rich enough to provide perfect outcome predictions, the proposed system can show what the most critical flows are and how well they generalize.

In general, composite event learning opens new possibilities but also posses several challenges. Understanding clustering results is inherently difficult, hence the average user will probably find it difficult to make sense of the high level events that are formed from several event types. Further investigations into how best to convey the components of a composite event are therefore necessary, i.e. the visualizations in Figure 4.3(c). Basic interactions like manual updates to a composite event or labeling of an event with a user-friendly name for future reference could be incorporated in future iterations of the system. Choosing appropriate parameters for the composite event learning, i.e. segmentation window size and number of clusters, is also a difficult task for the average user, thus it becomes important to show users how varying parameters affect both the resulting composite events and the quality metrics. Additionally, automatic suggestions for parameters that
score well on the different quality metrics can be included in the user interface to support users that are less familiar with parameter tuning.

When event sequences are not only ordered but also timestamped, several choices have to be made regarding proper data extracts for both pattern extraction and evaluation. Future work about how different choices to these parameters can seamlessly be incorporated into the user interface is also interesting, such that history is continuously used to assess the relevance of the computed patterns. This also means that timely insights, i.e. information about absolute time as in Figure 4.3g), should be incorporated at opportune steps in the overall analysis flow. For instance, in the business investigation domain, using information about when certain updates happened can narrow down the subset of interesting companies even further compared to only using the event tree.

4.3.2 Reflections on Domain Expert Adoption

The two prototypes have enabled useful dialogues, meetings, and workshops about data utilization with domain experts at the DBA and Grundfos. Outside of general usability issues, this continued exposure revealed several insights and barriers for tool adoption. Insights which have shaped the list of challenges presented in Section 1.4 and the continued work of this dissertation. I will briefly describe some of these insights that are not yet documented in any publication.

Distributed knowledge: The initial assumption behind the proposed prototype was that domain experts were able to distinguish relevant patterns in the data from noise due to their expertise. However, being a domain expert and being an expert on all parts of the data are not equivalent. The proposed prototypes leaves a lot of freedom to the users, but often the domain expert does not posses adequate knowledge of each individual data variable which makes the task of identifying relevant patterns difficult. The required knowledge is often distributed between multiple people within the organisation.

Steep learning curve: When a user has no experience with machine learning or interactive visualizations, the learning curve for using tools like the proposed prototypes is very steep. Besides making tool adoption difficult, this also makes it very difficult to design evaluations that can guide future tool development. In essence, the adoption of data analytics often requires significant changes to the current work processes.

Many types of uncertainty: Especially in the DBA domain there exist several types of uncertainty which makes data analysis difficult. Firstly, there are data uncertainties: (1) is the available data a good enough basis for making the needed conclusions and (2) what is the data quality, i.e. how accurate is the data? Secondly, there are model uncertainties: (1) are the correct
models and parameters utilized and (2) what statistical uncertainties does the specific models introduce?

Collaboration as a solution: Domain experts at the DBA often seek assistance from various personnel throughout the organisation. Even during our meetings, the participants would repeatedly discuss the meaning of certain data variables, potential analysis ideas, and patterns found with the prototype. Thus, the initial assumption that a single domain expert sits with an analytic tool and eventually makes a decision is naive at best. Our meetings and workshops have suggested that collaboration around data utilization is of high importance for tool adoption. Collaboration is the current solution for several of the challenges data utilization presents.

The presented insights suggest that it is not sufficient to only develop novel analysis methods and visual interfaces for a specific type of user in order to support the adoption of data analytics. As critical knowledge and required expertise is often distributed between multiple people, collaboration is a key requirement for utilizing data analysis methods within an organisation. These reflections prompted me to alter the focus of my work and start exploring collaboration in data work and how to develop support for the utilization of analysis methods like those presented in this chapter.
Visual Analytics in an Organizational Context

While designing specific methods for visual analytics is an important part of the current research agenda, our experiences from interacting with domain experts indicate that adopting visual analytics within an organizational context requires additional support beyond highly targeted tools. For example, it is not trivial how the methods we have proposed can be integrated within existing technology ecologies. In particular, as articulation and negotiation are major parts of current work practices, complex visual analytics tools for specific data and tasks does not necessarily comply with such processes. In this chapter, I will outline our work on further understanding how technology can support data utilization in an organizational context. By organizational context, I refer to instances of data work that incorporate multiple actors beyond small scale team work. Thus, the following is not per se a study of organisational structures or social relations, but rather a study of data utilization in a large scale setting. In the following, I will present our interview study and some of its findings about collaboration—work that resulted in paper B. I will conclude the chapter with potential design implications for visual analytics.

5.1 An Interview Study

In the fall of 2017, we conducted an interview study with personnel at the DBA to document how case workers, domain experts, and analysts currently collaborate around data utilization and to identify recurring challenges. The distinctions between these roles are ill-defined, fluent at best, as the main differences have to do with expertise in different areas of data work. Thus, the recruitment for the interview study was fairly open-ended—based on our requests and intentions for the study a negotiation with contacts from the organization allowed us to recruit 9 participants. The participants had different job roles and varying degrees of contact with data analysis in their everyday work. In addition, several of the participants were also part of organizational-wide initiatives to improve the use of data, including the development of a shared risk model consisting of different parameters and heuristics for suspicious behavior. Apart from the interviews, we also got access to the shared document containing the risk parameters.

We followed a semi-structured format to allow for flexibility in the data collection. As further described in paper B, the interviews focused on everyday tasks, work process, data, tools, and collaboration. The interviews were recorded
and subsequently transcribed. We then did a coded analysis of the interviews—
involving identifying codes and themes individually, collectively comparing the
results of the first phase, re-analyzing with the selected categories, and identifying representative quotes. I refer to the paper for further details as well as for the participant ids used with the quotes.

5.2 Aspects of Collaboration in Data Work

While collaboration and cooperation are essentially two different aspects of teamwork, the terms will be used interchangeably for the purpose of this work. Thus, I will use collaboration to describe aspects of data utilization both where multiple actors share a common goal and where multiple actors share information but have individual agendas. Firstly, because it is challenging to distinguish when what occurs, and secondly, because becoming data-driven within an organization already creates the common goal of utilizing shared data efficiently and correctly. We use the term data work to also include the types of work around data utilization that are not strictly data analysis (see Section 2.3.1).

The findings of our study consist of overall aspects of collaboration and concrete examples of data work scenarios that can guide future design of technology support. While several of the aspects are also mentioned in related work, the introduction of advanced data analysis methods suggests an enhanced need to focus on these aspect in order to support adoption within organisational contexts. In the following, important findings for the focus of this dissertation will be presented. Additional details about the study can be found in paper B.

5.2.1 Sharing Across Organizational Boundaries

Our study has shown that being able to share data work across organizational boundaries is critical to the success of introducing data analysis methods. Being able to share data work is both a requirement as well as an opportunity for improving existing work practices.

It is a requirement due to the fact that an increasing number of people are required to use data from the shared repositories. Thus, shared repositories introduce dependencies between people in that; changes to the data can affect work elsewhere in the organisation, more people are required to understand parts of the data, and alignment of interpretations and analysis methods becomes important. Currently, much data work at the DBA relies on creating data extracts which a team may collaborate around in spreadsheets. This means that if modifications to the data occur it is challenging to detect by other teams working on data extracts of their own. In addition, this way of working creates a loose coupling between steps in the process; from defining requirements to a data extract, to programming the script that produce it, to multiple data analysis steps in different tools, and to interpretations and conclusions. Retracing these steps is not only an individual challenge, but a collective one, as it rarely only involves a single person.
It is an opportunity since the same analytic parts can be beneficial for multiple tasks across the organization. The evolving risk model at the DBA is an example of this. Recently, the DBA starting consolidating the risk parameters from different teams in the organisation, as they experienced that individual teams often came up with similar indicators even though the focus of their tasks were different. But the shared document containing the indicators also illustrates several challenges with sharing data analysis work. For example, the indicators mainly consist of textual descriptions and they are rarely described with variable names from the actual data. This detachment from the data means it can be difficult to concretely implement the indicators. Furthermore, the different indicators do not conform to a fixed type, i.e. some indicators are concrete checks in the data, others are calculated values, and several require contextual judgment. Utilizing the risk model across organizational boundaries therefore often requires significant translation work, and text alone is often ineffective to describe all the relevant knowledge—including metadata, implementations, motivations, usage, etc.

### 5.2.2 Transparency and Explainability

Uncertainty in visual analytics research often deals with statistical uncertainty of computational methods, and thus this is the typical challenge associated with explainability. However, both our study and the field work involving the presented prototypes have illustrated that much uncertainty also relates to the transparency of data work. Having a deep understanding of how data is collected, transformed, interpreted, and related to other data sources is important to fully utilize advanced analysis methods, e.g. like the methods we have proposed. Thus, collaborative data work at the DBA involves much articulation work where different aspects of the data are explained. As a consequence, data utilization at the DBA requires significant exploration of internal data and capabilities:

“It means a lot when you talk to someone with a deep understanding of the data, but as we often only have a combined small understanding of the data, it requires a lot of digging to understand completely.” (DBA-M3)

Transparency is also important for personnel beyond those doing analytic work. Effective data analysis does not occur in a vacuum as results need to be utilized elsewhere in the organization by different stakeholders:

“The numbers you pass on in the system can be used politically, so I am trying carefully to explain the limitations of my work.” (DBA-M2)

Thus, communicating analysis decisions and limitations is equally important to facilitate that others can make informed decisions. A challenge with maintaining transparency is the loose coupling between the tools utilized at the DBA. It can be hard to trace how scripts, spreadsheets, and other analytic tools have transformed data, and e.g. how this has led to the interpretations in the risk model.
5.2.3 Repeated Sensemaking

The participants also reported that repeating steps in data work was necessary to get the appropriate overview, either because an existing case was handed over or because they returned to past tasks:

“Typically, they would probably start over, because you need the overview of a case, which is difficult to pass on.” (DBA-F1)

Repeating sensemaking steps occurred for several reasons; to get an adequate level of understanding, to ensure quality, or due to being unaware of past analytic work. The participants also reported that they would frequently forget or lose past analysis steps when a task was solved—steps which likely would be reusable in future work. However, the participants also employed different methods for tracking how they arrived at conclusions. For example, through simple note-taking or through more advanced initiatives like storing snapshots of the shared database to maintain the provenance of a certain analysis. Tracking analysis and reasoning steps to support recall, review, and reuse is therefore a significant challenge at the DBA—a challenge where technology support has the opportunity to minimize the need for repeated sensemaking.

5.2.4 Study Limitations

In general, interview research relies on the point of views of the participants and that the chosen participants match the study focus. Therefore, the challenges of interacting with a large organization makes it difficult to assess how generalizable the findings are. Furthermore, generalizing from a single case study can be difficult without relying on similar findings from existing work. Thus, the findings should be interpreted in light of related work. Still, the interviews have been central to our understanding of existing work practices and the domain challenges described in Section 1.4. In addition, the continued interaction with collaborators from the other domains have also supported several of the findings. Organizational limitations have also restricted us from doing more in-depth studies of the current work practices (e.g. observations). As a consequence, some of the findings are high-level in nature as concrete collaborative practices or scenarios can be hard to recall. On the contrary, the current work at DBA is still in its beginning with respect to data utilization. Thus, it is not clear that observations would provide significant value, as part of the goal with the work of this thesis is to support the adoption of collaborative visual analytics and therefore also to reshape existing work practices.

5.3 Design Implications for Visual Analytics

The presented findings are not an exhaustive list of collaborative aspects. However, they present significant challenges that can be supported by altering how
visual analytics technologies and tools are developed. In particular, the findings show that no single person has all the knowledge or competences required for effective data utilization, thus visual analytics should address collaboration as a core part of the process. In the following, I will propose design implications for visual analytics based on the findings and related work.

5.3. DESIGN IMPLICATIONS FOR VISUAL ANALYTICS

5.3.1 Mixed Expertise Collaboration

As implied several times throughout this dissertation, effective utilization of data require multiple types of expertise which a single person rarely posses. In large organisations it is simply infeasible that a single person has to maintain a deep understanding of all parts relevant to data analytics—such as data collection, maintenance, analysis, relevant decisions, etc. Visual analytics applications should therefore focus on enabling mixed-expertise collaboration. At the DBA, they described how ambassadors (a term used by the participants) with partial expertise in multiple areas were important to support the adoption of data analysis:

“He is a good example of an ambassador that is capable of having a foot in two camps, and that is really great.” (DBA-M9)

For example, ambassadors would provide technical support within teams of domain experts and translate misunderstandings across organizational boundaries. There are several potential design implications for technology that can support this type of collaboration.

Ideally, multiple collaborators should be able to contribute with analysis parts based on their expertise. Currently, technical expertise is a barrier for such collaboration. Participants that can build advanced cleaning, transformation, and analysis with scripting have a difficult time collaborating with users who prefer, e.g., excel. For collaborative work, often a shared set of tools needs to be negotiated. Thus, technologies should support the integration of different types of interfaces appropriate for users abilities.

Another potential design implication is to support the high-level notion of boundary objects or boundary negotiating artifacts. I.e., build artifacts that multiple users can use and understand despite their community of practice. To some degree, spreadsheets already present such qualities. The familiar format supports users despite their expertise to trace how results were computed, and the concept of a document allows users to update parts without being a data analysis expert. In addition, the tool has advanced options for experts which integrates easily with the simpler functionalities. Thus, extending the notion of a malleable object with familiar structures that still provide an open-ended workflow may be powerful for collaborative visual analytics.

Furthermore, people comprehend things optimally in different ways—a sentiment embodied in the concept of external representations [167]. Providing the same knowledge in different ways may therefore be another design implication
for collaborative visual analytics. It should be possible to build external representations for different analysis parts and on multiple occasions throughout a collaborative process to continuously maintain common ground.

5.3.2 Levels of Data Analytics Integration

The study has highlighted that there are more aspects to the adoption of visual analytics in an organizational context beyond designing specific analysis tools. Solely focusing on data types and analysis tasks does not suffice as illustrated by the collaborative nature of data work and the challenges involving shareability, transparency, and technical expertise. Thus, I propose three expanding levels for supporting the adoption of data analytics in large scale settings with multiple interleaving actors (cf. Figure 5.1). This model is for data analysis in general as general data utilization requires the combination of multiple types of technologies—including visual analytics tools.

Figure 5.1: This onion model depicts expanding aspects of data analytics adoption in large scale settings that are important foci for future research.

The three levels in Figure 5.1 resemble the levels of evaluation presented in Illuminating the Path [69] in that they consider expanding aspects of data analytics integration. However, instead of presenting increasingly holistic (and challenging) evaluation metrics, the levels of integration propose to consider and design for three areas of data utilization; (1) concrete data analysis methods and tools, (2) integration of methods and tools with existing technologies, and (3) involvement of people with different expertise and competences in data analytics. By considering all three levels of integration when designing visual analytics technologies, we are effectually consolidating existing research on information visualization, data analytics, computer-supported cooperative work, and software architecture. While IT infrastructure, interoperability, and involvement of people through technology
are not by themselves new aspects in research, the proposed levels illustrate that there are still unexplored opportunities for designing supportive technology with data analysis as the central focus. In addition, the combined focus suggests that research into increasingly holistic aspects of data utilization is important. The following chapter illustrates how this focus has led to the design of technologies for visual analytics that inherently supports collaboration. First, I will briefly discuss each of the three levels.

The first level is about developing methods and tools that enables the analysis of different types of data for different purposes. As illustrated in Section 2.2.1, this has been the defining focus of most visual analytics research thus far, and it will likely remain a large focus as the opportunities for analyzing data are continuously evolving. In visual analytics research, the guiding model for tool design is the general process involving both visual exploration and automated analysis (cf. Figures 2.1 and 2.2). The process model has also shaped the prototype designs presented in this dissertation (cf. Chapter 4). However, as the model is very general, it does not capture the nuanced goals of data analytics or how collaboration is part of visual analytics. For example, it can be argued that the same process can be followed in a computational notebook which further allow users to tinker with the model without interacting in the visualization. Thus, the model provides little guidance for design beyond the general flow of visual analytics tools.

The second level is about integrating new analysis methods and tools with existing technology. As illustrated by our study, it is not trivial how new methods can be combined with already existing software and augment or modify current work practices. In addition, as analytical needs are constantly evolving, it becomes a continuous requirement to incorporate new methods and tools. Among other, this level of integration is related to the challenge of sharing across organizational boundaries. For example, being able to incorporate new scripts can be important to always align how data is used. Nouwens and Klokmose [230] have described how the current application-centric focus in computing has certain consequences for knowledge work. In data analytics, the application-centric focus requires collaborators to negotiate common data formats for transferring data and analysis results across tools. However, this integration strategy places additional effort and responsibility on the user to develop and follow such formats—challenges which have been elaborated in related work [34, 84, 240]. Thus, the loose integration of tools suffers from several of the collaborative challenges previously described. In addition, fixed applications require substantial effort (and programming expertise) to modify or extend, and thus to adapt existing interfaces to new analytical requirements. Currently, integration of technology is typically an industry focus. Organisations can utilize the complete application stack from Tableau [290] or the Microsoft infrastructure for handling data including Power BI for visual analytics [210]. However, I argue that there is an opportunity for research to also make an impact in this area and to explore ways to enhance how tool integration can be supported. In Section 6.1, I will therefore present an alternative model for integrating new methods and tools that afford collaboration and sharing of
atomic analysis components.

The third level is about involving people with different expertise and competences in data analytics. This includes people who may not actively modify the analysis, but who will review, comment, clarify, or otherwise use analytic results. Our study shows that substantial knowledge capital exists within organisations—knowledge which is underutilized and valuable for data analytics. Furthermore, our study indicates that such knowledge is often highly distributed between people. On a conceptual level, this suggests that technology for data analytics should to a larger degree utilize some of the driving theories behind collaboration such as establishment of common ground [67], boundary objects [284], and boundary negotiating artifacts [190]. More concrete objectives include supporting external representations while maintaining malleability and interactivity and providing explicit traces between the tools in use and the explored analysis steps. Related work within this area of data analytics integration includes allowing multiple participants to comment on visualizations [131, 302], the concept of data-driven storytelling [189], allowing users to seamlessly switch between analysis and storytelling [113], and shared knowledge graphs [47]. For example, it has previously been shown how the combination of visual summaries, analytical provenance, and narratives can be a powerful tool to capture analytical decisions and support critical thinking [304]. Still, there is a limited focus on mixed-expertise cooperation, and we only scratched the surface of how multiple people can be involved in data analytics—both as consumers and contributors.

The work presented in this dissertation contributes to all three parts of data analytics integration. The tools described in Chapter 4 present novel analysis methods for challenging data types and tasks. In Chapter 6, I present our work on addressing both tool integration and people involvement. Section 6.1 introduces Vistrates—a model for tool integration that supports composing, extending, and sharing of analysis components. In addition, the Vistrates platform supports synchronous collaboration and development of multiple interface types for different users. Section 6.2 describes our model for integrating elements of data-driven storytelling and provenance tracking in collaborative visual analytics as a way to support involvement of people. In essence, the system extends current capabilities for replicability, shareability, and communication in data analytics. Thus, my work shows how the expanding focus presented in Figure 5.1 allows us to think about collaboration and visual analytics which goes beyond the process model presented in Figure 2.1.
This chapter presents two novel systems that explicitly address the expanding areas of data analytics integration described in Section 5.3.2—tool integration and involvement of people. The presented work is based on paper C and D.

6.1 A Collaborative Component Model

Tool integration for data analytics can also be characterized as an issue of interoperability between software that combines, modifies, computes, or visualizes data. Both research and industry (cf. Chapter 2) produce numerous applications for different data analysis tasks, expertise levels, and devices. However, it has previously been described how the focus on application-centric computing forces collaborators to make collective software compromises and how task fragmentation across applications introduces increased efforts for sharing, importing, and exporting information—findings echoed in our study of data work (cf. Chapter 5). Therefore, in paper C we introduce an alternative software model that supports collaboration and interoperability in data analytics, and thus directly supports integration of different analysis tools. The goal behind the model is to apply the vision of *shareable dynamic media* to data analytics, effectually blurring the distinction between application and document. To achieve this vision, we propose that the approach should be based on the web, as web documents already to some extent embody both content, computation, and interaction. Furthermore, as described in paper C, the approach should be open, extensible, reusable, and composable. Furthermore, it should support multiple, adaptive, and simultaneous layers of access. A system that supports all this, would be a large step towards truly ubiquitous analytics, where multiple people are able to analyze data anytime and anywhere.

6.1.1 Design Concepts

Concretely, we propose a component model for data analytics. In total, the suggested design consists of six concepts as described in paper C, which I will briefly summarize here:

**Component-based Pipeline:** Components are the building blocks of the architecture. These can be composed and reconfigured in reactive data-flow
pipelines forming the analysis. Components are executable blocks of code with input sources, a data output, a state and a view. Thus, a component can essentially contain any analytic functionality, including loading data, computational transformations, and visualizations. The notion of reactive means that updating the output of a component will trigger recalculations in any components that have the updating component as their input source.

**Collaborative Pipeline** In this context, collaborative means that the pipeline should be accessible and modifiable by multiple participants simultaneously. Thus, the analysis pipeline should synchronize component states across clients. However, clients should execute code locally as this provides flexibility for the collaborative functionality of a component. The collaborative pipeline therefore have shared execution flow between clients. This way, it is up to the developer of a component to, e.g., specify which interactions in a visualization should be synchronized.

**Prototype-based Components** Instantiating a component should consist of copying the code from an existing prototype. While this violates known software principles, it allows components to be completely modifiable and existing components can this way form the basis for new ones. In the basic scenario, developers simply instantiate the empty component template.

**Multiple Levels of Abstraction** The architecture should support multiple levels of abstraction in the sense that users can work in different interfaces simultaneously. Developers can program new components, analysts can instantiate and compose existing components, and domain experts can interact with resulting visualizations. This can be supported by continuously synchronizing the lowest level of the system, i.e., the code and the state specification. By doing this, any changes will immediately be reflected in any type of interface across all clients. Thus, developers may even build entirely new interfaces on top of the component model. An example of this follows in Section 6.2.

**Component Repository** Component prototypes are maintained in a repository, which enables sharing of atomic parts across different analysis scenarios. Thus, components can this way be reused across organizational boundaries.

**Transcend Application Boundaries** This design concept follows from having multiple levels of abstraction. Nonetheless, it highlights that the results and visualizations should be able to integrate into other types of media such as presentations and reports. This way, sharing is not only supported for analytic activities but also for presenting findings to, e.g., stakeholders.

A central goal behind the combination of these design concepts is to support tool integration. The design allow users to share new tools and methods and
either extend existing solutions or recompose entire new solutions. In addition, the model allows participants to collaborate in a way that suits their abilities without negotiating a lowest common software denominator. Thus, the component model is a step towards the type of mixed-expertise collaboration described in Section 5.3.1. In essence, a system implementing the proposed design is an enabling platform for the development and use of data analytics. Visual appearance of the individual components, appropriate pipeline compositions, and interface designs are thus left up to the developer. However, as mentioned throughout this dissertation, these aspects are exactly the focus of most existing research in visual analytics. Thus, there is a rich body of work that can directly be implemented in such a platform, which would turn existing solutions into collaborative, modifiable, and extensible data analytics approaches.

6.1.2 Vistrates

The conceptual design has been realized in the prototype-system called Vistrates. Examples, code, and presentations can be found at https://vistrates.org. Vistrates is implemented with basic web technologies, including JavaScript, HTML, and CSS, as well as the two existing web frameworks; Webstrates [169] and Codestrates [39].

Webstrates [169] is the realization of the shareable dynamic media vision. The main functionality of Webstrates is DOM synchronization, which makes webpages collaborative and editable in real-time. When modifying the DOM of a webstrate (a webpage in Webstrates), changes are persisted and shared between all clients—similar to collaborative text editors. For example, clients of a webstrate can edit both the Javascript, HTML, and CSS code directly in the developer tools of a browser. Besides the main functionality, Webstrates provides different low-level APIs—e.g. for signalling other clients, versioning, and asset management. Codestrates [250] is a computational notebook build with Webstrates. Thus, a codestrate is a specific type of webstrate that contains editing tools for modifying its own content following in a format that is similar to other literate computing environments. A codestrate is structured with paragraphs and sections. Paragraphs are individual blocks of either executable JavaScript code, JSON data, CSS style, or web content. Sections are encapsulating several paragraphs. Furthermore, sections can be turned into packages that can be shared with other codestrates [39]. As Codestrates utilize the standard execution in the run-time of the browser, it is easy to integrate external JavaScript libraries. Vistrates is essentially a core framework, composable components, and view abstractions build as packages in Codestrates. The architecture stack is shown in Figure 6.1.

The core framework manages the execution-flow defined by the component pipeline, implemented with software principles such as inversion of control and dependency injection [101]. The core framework consists of a singleton [101] that registers all components of a vistrate, a component controller class implementing the observer pattern, and a model for executing code provided by the user. The
Figure 6.1: The Vistrates architecture and its relation to Codestrates and Webstrates as presented (paper C).

```javascript
vc = {
  data: "id-of-vis-data",
  src: ["mySourceName_1", ..., "mySourceName_n"],
  props: ["myProp_1", ..., "myProp_m"],
  libs: ["myLibraryStoredAsAsset.js", "https://somecdn.com/anotherLibrary.js"],
  init: function() { /* code goes here */ },
  destroy: function() { /* code goes here */ },
  update: function(source) { /* code goes here */ }
}
```

Listing 6.1: The code paragraph template for a Vistrates component (paper C).

singleton is responsible for updating observers accordingly when components are updated. A basic component consists of three paragraphs grouped in a section; a code paragraph, a data paragraph, and an optional view paragraph (web content). However, a component can also contain an arbitrary number of additional paragraphs, e.g. for CSS styling. The code paragraph contains the methods and properties of a component following the template in Listing 6.1. Thus, the code paragraph of a component contains a reference to the data paragraph, a list of input sources and property mappings, libraries to load, and three methods for the general pipeline execution flow. Instantiating an empty component will come with the basic template (cf. Listing 6.1) and the developer can then insert code with the desired functionality. Instantiating an existing component will thus come with already inserted functionality.

The data paragraph contains the configuration of a component and its current state following the template in Listing 6.2. Thus, this is where the user specifies the mapping between the source and property variables of the code and the actual instantiated components. The data paragraph also stores the state specification of a component, which can for instance be used to store those selections of a visualization that should be synchronized or the output of a computation. The view
6.1. A COLLABORATIVE COMPONENT MODEL

Listing 6.2: The data paragraph template for a Vistrates component (paper C).

```json
{
    "config": {
        "src": {"mySourceName_1": "source_1_id", ..., "mySourceName_n": "source_n_id"},
        "props": {
            "myProp_1": { "src": "mySourceName_1", "prop": "somePropOnSource"},
            ...
            "myProp_m": { "src": "mySourceName_n", "prop": "someOtherPropOnSource"}
        },
        "view": "id-of-vis-view"
    },
    "data": { /* The data field of the component for storing state */ }
}
```

Paragraph contains the visual output of a component. The view is wrapped in a transient element [169], which means that the view is not directly synchronized. Instead, it is the synchronized code that produce the view on each client also using any state stored in the data paragraph. This way, the developer can control which parts of a visualization should be synchronized. For example, the viewport of a map visualization may not be synchronized while selected areas on the map should. To support this, components observe not only their input sources but also changes to its own data paragraph. Updating the state specification on one client will thus trigger updates on all clients. In Paper D, the Vistrates core framework was further extended with composite components. This allows developers to combine atomic components into composites, and thus share more advanced functionality. An example is the combination of an aggregation component, a visualization component, and a filter component which are frequently composed in the exact same way. The implementation of composite components follows the composite software pattern [101].

Besides the core framework, Vistrates consists of various view abstractions and a component repository containing a series of already implemented analytical components. The repository is implemented using the package management of Codestrates [39]. The basic view abstractions include a pipeline view, a canvas view, and a mobile view. In the pipeline view, users can inspect the structure of the current pipeline as well as reconfigure components in a drag-and-drop manor. In the canvas view, users can insert and arrange component views along with hand-drawn elements on a canvas. With the mobile view, users can simply select a specific component view to fullscreen. In Section 6.2 I will present how we developed a more advanced interface on top of Vistrates for integrating data-driven reporting. For more details on the implementation of Vistrates I refer to papers C and D.

With Vistrates, we have contributed with a concrete platform that; (1) enables many activities related to data analytics interchangeably, including development, exploration, and presentation, (2) supports multiple heterogeneous devices, and (3) facilitates collaborative workflows. Besides solving real-world challenges for tool integration, the platform has the potential to support a shared knowledge base in visualization and data science—it can be a direct way for researchers and analysts alike to share work related to the use of data. In addition, by supporting
6.1.3 Usage

The basic workflow begins by creating a codestrate and installing the Vistrates packages. The easiest way is to use the HTTP API of Webstrates [1] and copy


multiple levels of abstraction, Vistrates is an ideal environment for teaching visualization and analysis techniques—users can gradually progress from high-level analytic interfaces, to pipeline composition, and finally to actual code within the same environment. Vistrates also enables future research into expanding aspects of collaboration in data analytics. Our work on bridging data analytics and storytelling (cf. Section 6.2) and on distributing visualization interfaces (cf. Section 7.1) are concrete examples of the potential research implications of Vistrates. As there is yet little support for composing appropriate analysis pipelines, an idea for future work is to use the concept of pipeline templates to support the sharing of high-level reusable analysis structures.
an existing instance. Users then instantiate components which will insert the corresponding paragraphs in the notebook. Users can configure the components directly in the data paragraph or using the pipeline view. An analytical pipeline will most often start with a component for loading data followed by a series of transformations and visualizations. In Vistrates, visualizations are not only endpoints—visualizations will also have a data output, such as a the current selection. Figure 6.2 shows a simple component that creates a line chart. The current selection is both stored as its state in the data paragraph and used as the data output for subsequent components.

The pipeline for generating the line chart is shown in Figure 6.3. Besides the line chart itself, the pipeline consists of a component for loading data with Baltimore crimes, a component for aggregating the data, and a component for filtering the data with the selection in the line chart. At any time, developers can modify the code of instantiated components and upload new components to the repository. Furthermore, the resulting component views can for instance be inserted on an interactive canvas or embedded in a slideshow presentation. Thus, Vistrates can be an analytical environment itself, but also form the basis for other analytical interfaces.

6.2 Bridging Data Analytics and Storytelling

As described in Section 2.3.3, data-driven storytelling can be a way to establish common ground and support explainability. However, to truly support collaborative workflows, data-driven storytelling should not only be a tool to communicate final results after the analytical activities has concluded. Rather, collaborators
should be able to share findings throughout the analytic process, which suggests a need to bridge data analytics and storytelling. While Vistrates presents a concrete model for supporting tool integration as well as the concept of view abstractions that facilitate collaboration between different expertise groups, there is little support for sharing motivations, reasoning, and the history of analytical activities. Thus, to take a step further and develop a model for better utilizing the knowledge of different people throughout the analysis process, we developed a system on top of Vistrates—effectually supporting the involvement of people as presented in Section 5.3.2. The conceptual design and the system have resulted in paper D.

6.2.1 Replicability, Recall, & Transparency

As our study suggested (cf. Chapter 5), transparency and repeated sensemaking are two central aspects of collaboration in general data work. Being able to replicate results, trace how they were computed, and recall any reasoning are central to both support the construction of common ground and the ability to explain potential findings. In addition, research have described several challenges with tracking and sharing exploratory data analysis—especially since exploratory data analysis is iterative, non-linear, and often investigates multiple avenues of inquiry [18, 241, 279, 299]. In particular, it has been described how communication of insights and hand-off of partial progress is challenging, especially before the analysis has concluded [128, 159, 203, 262, 276, 331]. Thus, the goal with this work was to integrate elements of data-driven storytelling and provenance tracking with the literate computing paradigm in order to support reporting as an intrinsic part of the analytic process. In paper D, we elaborate on three concrete challenges for integrating data-driven reporting, including; combining annotations and external representations, having various levels of details, and supporting fragmented workflows. These challenges led to three concrete design concepts that the system in the following section embodies:

**Linking Annotations to Analysis States:** To create a close coupling between the narrative and the provenance of the analysis, we propose to maintain links between annotations and the states of the analysis parts they describe. This is similar to work on capturing visualization states [131, 302], but extended to include all components of a given analysis—including also data and computation. We further introduced the concept of active and inactive annotations to describe whether a given annotation currently matches the state of the analysis, and thus whether the annotation applies. Figure 6.4(A) illustrates this bidirectional dependency between annotations and the underlying analysis. This enables bidirectional exploration, where users can either explore annotations and restore past analysis states or modify the analysis states directly and explore which annotations no longer apply.

**Dynamic Insight Hierarchies:** To support different levels of detail and maintain transparency, we propose to maintain the narrative in a hierarchy. Com-
Hierarchical Presentation Views: To support the notion of external representations, annotations can also be linked to presentation views containing e.g. multiple visualizations or drawings. This concept is illustrated in Figure 6.4(B). The concept of linking annotations to analysis states ensures that the current views match the active narrative. The concept of dynamic insight hierarchies effectually makes the final report a hierarchical presentation. In addition, this concept enables the opportunity to create dedicated presentation interfaces that for example limit user control and thus better supports the navigation of the hierarchy.

For more details on the challenges and the design concepts, I refer to paper D. The proposed design is a concrete way to bridge data analytics, provenance tracking, and elements from data-driven storytelling. The approach is supporting transparency of the analysis steps taken, while simultaneously supporting a way for collaborators with limited expertise in data analytics to explore and comment on the current results and progress. Thus, it is a way to involve people throughout the analytic process by having technology that serves as a form of boundary object. By bridging these techniques, the proposed design significantly extends...
current capabilities for capturing and presenting analytic reasoning, interpretation, and motivation.

6.2.2 InsideInsights

InsideInsights is the realization of the proposed design, build with Vistrates as the underlying analysis system. Code can be found at https://github.com/90am/insideinsights. In essence, InsideInsights is an interface abstraction on top of the computational notebook, where the component model forms the bridge. This means that you can switch between the two interfaces interchangeably. This way, users can build hierarchical analysis reports and develop new components as needed throughout an analysis process.

First of all, InsideInsights improves the pipeline view for composing analysis components. Figure 6.5 depicts the improved pipeline visualization. By introducing composite components, the pipeline essentially becomes hierarchical, i.e. composite components can be collapsed or expanded depending on whether the users wants to understand the inner functionality of a particular analysis part. In addition, to support understanding of the analysis flow, components are automatically colored based on whether they represent; input data with no input source (e.g. data or an input field), a computation (e.g. clustering or aggregation), a visualization, or a composite.

Second of all, the system supports annotation-driven provenance tracking. When the user annotates a specific component (see Figure 6.6), the provenance for the current state of that component is automatically maintained in the background, which include the states of the previous components in the pipeline leading up to the annotated one. Thus, the system maintains a dependency graph of analysis states which is defined by the structure of the pipeline. But, to avoid cluttering the provenance graph and avoid the need to subsequently make sense of a large graph, it is only extended whenever the user annotates something. The provenance graph simply utilize the JSON state specifications of the components.
Finally, InsideInsights introduces hierarchical narratives and presentation views that together form the interactive reports. Figure 6.7 shows the interface with the hierarchical narrative on the left and a presentation view on the right. The right side can either show a presentation view or the analysis pipeline, thus users can compose an analysis directly within the interface. Currently presentation views can either be a particular view of the pipeline or a canvas with one or multiple component views arranged along with hand-drawn elements. The individual annotations are maintained in cells in computational notebooks, which enables; (1) a one-to-one mapping between cells in the InsideInsights interface and the underlying notebook and (2) the ability for users to continuously structure the hierarchical narrative as needed. To support the user in adding annotation in semantically meaningful locations of the hierarchy, annotations describing the same components are automatically maintained in a stack (group). Figure 6.6 shows how two annotations about the same component are represented as dots (tabs), where the visible one is the orange (active) and the other grey (inactive) is currently hidden. Annotations can be activated as needed, which restores that analysis state, or the analysis state can be modified and the annotations updates accordingly. For additional implementation details see paper D.

The presented elements along with the underlying component model supports an open-ended workflow similar to the one already utilized in computational notebooks. As needed, users can interleave composing analysis parts, developing components, creating the narration hierarchy, and sharing current progress with collaborators. Paper D presents concrete use scenarios for how such collaborative workflows can unfold. With InsideInsights, we both continue a trend for improving the usefulness of computational notebooks and show how high-level reasoning interfaces can be built on top of existing analysis systems. In theory, InsideInsights can also be built for other analysis systems that expose internal states. In addition, the system sets a new trend towards a data analysis paradigm we call literate analytics. I will further discuss literate analytics and implications for future research in Chapter 7.
6.2.3 Expert Feedback

As described in Section 3.2, it is infeasible within the time frame of this project to do evaluations with extended use of the systems. Thus, to elicit initial feedback about the potential of such a system, we opted for an expert review with two experienced visualizations researchers. Qualitative expert reviews have previously been suggested to be the better option when evaluating high-level analytic work involving visualization [54, 242, 297]. For this purpose, we build different analyses with the InsideInsights system. Figure 6.8 shows an analysis of Baltimore crime peaks in which protests at the funeral of a young person shot by the police coincides with an increase in burglaries throughout town. The experts were given both a full report, as the one shown in Figure 12.8, and a report that only included the analysis pipeline, i.e. without the narrative. For the first report, they were tasked with explaining the analysis and its components, and for the second report, they were tasked with writing the narrative themselves.

In general, both participants were able to solve the tasks. They both found that the system have great potential, was engaging, and that progressively expanding the level of detail was a good way to understand an analysis. However, the open-ended workflow and generality of the system meant that it was difficult at times to understand both the hierarchical narrative and the hierarchical pipeline simultaneously—the freedom of the system "comes with responsibility" as one participant noted. The other participants noted that it could be challenging...
6.3 Encapsulating Existing Technologies and Analytic Approaches

Crime Peaks

The main result shows that burglars occasionally take advantage of chaotic situations to commit crimes in the city. This is likely because they know the police will be elsewhere.

Occasionally, there is a strong connection between the crime peak and the occurrence of an event that required a lot of police attention.

The crime location map is one way to identify this.

The data is based on the first component named Baltimore Crimes.

The five part of the analysis is to identify crime peaks. The Crime Over Time chart displays peaks and select the appropriate time periods for further analysis.

April 27th is the day of the funeral of Franky which was eventually led to poverty.

Link: https://en.wikipedia.org/wiki/2015_Baltimore_protests

Next step is to identify what types of crimes occurred in the selected crime period using the Crime Type Graph below. In this plot, categories of interest can be selected for further investigation.

Burglary is the most frequent crime type in the selected time period.

It is always ideal to confirm that the peaks in individual crime types are in fact also on the ordinary in general. This can be confirmed using the Exotic Crime Age chart, that shows the average number of crimes per day of that particular type.

The locations of the selected crimes might reveal why the peak in crime occurred. The locations of the selected crimes can be viewed in the Crime Location chart.

The burglaries are scattered throughout the city.

Figure 6.8: An interactive report containing an analysis of crimes data in the city of Baltimore presented in paper D.

to understand the narrative when mixing annotations about how, what, and why. He suggested that the system should allow users to explicitly tag annotations with types and let reviewers choose a certain perspective to read. While the system takes time to learn and currently presents usability limitations, the expert review along with the investigations leading up to the design suggest that integrating data-driven reporting can be a way to better share analytic work and improve transparency.

6.3 Encapsulating Existing Technologies and Analytic Approaches

An essential part of evaluating system design is to argue for importance and generality [232]. The motivation behind the design of Vistrates and InsideInsights has not been to replace existing technologies or ways of analyzing data, but rather to augment existing capabilities with synchronous collaboration, interoperability, shareability, and transparency. The importance of the proposed systems stems from the presented domains and the findings of our field work. In addition, it is not only the DBA that experience the need to share analytic work across organisational boundaries. Also, at Grundfos and in the healthcare sector multiple entities are required to utilize data from large shared repositories. I will further discuss the other domains in Section 7.3.
To argue for generality and completeness with respect to the proposed levels of data analytics integration, it is important that the proposed systems can encapsulate existing technologies and prototypes. In the paper C, we show how the component model can encapsulate several existing visualizations libraries—such as D3 [77], Vega [269], Vega-Lite [270], Plotly [244], and Leaflet [185]. We also show how Vistrates can bridge multiple heterogeneous devices, which I will return to in Section 7.1. The generality of InsideInsights is basically inherited from Vistrates, as any analysis that can be implemented in Vistrates can also be tracked and annotated in InsideInsights.

To complete the expanding areas of data analytics integration presented in Section 5.3.2, I have re-implemented the event analytics methods and visualizations from Section 4.2 in Vistrates and created a narrative with InsideInsights. Figure 6.5 shows the clustering functionality of the event analytics implemented as components. Figure 6.6 shows an annotation for the event analysis in which specific types of company status updates are chosen. Figures 6.7 and 6.9 depicts the interactive report with a presentation view and part of the analysis pipeline as the right view, respectively. It would even be possible to build the exact same interface as presented in section 4.2 with Vistrates and only use the reporting functionalities of InsideInsights as needed.

A main limitation of the proposed systems is currently scalability. Thus, immediate future work involves the development of components that do not necessarily have to be executed within the browser and additional user-control for the update flow—in a scaleable solution, every update should not trigger recalculations of all subsequent (and potentially heavy) components unless it is the...
intended functionality. Concretely, we have considered the idea of python-based components that can be executed on the Webstrates server, but still be composed with any JavaScript components as needed. However, this also creates interesting design challenges for the InsideInsights interface, e.g. to highlight when certain components needs to be recalculated.
While this dissertation contributes new interactive technologies that solve concrete challenges from real-world domains, the presented work also opens several opportunities for future research. In the following, I will present additional perspectives of data utilization that can inspire continued work.

7.1 Collaboration with Multiple Heterogeneous Devices

An aspect of collaborative technology that has received little attention in this dissertation thus far is to support scenarios with multiple heterogeneous devices. The reason for this is, that it has not been a prevalent scenario in the discussions with the domain experts and in our empirical study. Nonetheless, it has received significant attention in related work and future work scenarios may very well incorporate multiple heterogeneous devices simultaneously, either for single person use or for collaboration. Thus, the Vistrates platform was also designed with a vision to transcend multiple devices. The platform can therefore support data analysis that incorporates different types of devices as well as be a foundation for research into these types of work scenarios.

In the paper C, we show how the component model can be used to offload heavy computations to stronger peers. Concretely, we developed two types of components that can encapsulate a certain part of the analysis pipeline. If the document is loaded on a weak peer (e.g. a phone), the heavy-start component will broadcast "help" to other clients of the same document. A strong peer can then execute the heavy sub-part of the pipeline and in the heavy-end component the weak peer will receive the computed result. This way, users can easily develop an analysis pipeline where the results of advanced analytical methods can be visualized on e.g. a phone. In the paper, we show how the phone can visualize the resulting word embedding from an analysis of the restaurant reviews in the vicinity of the user.

We have also used Vistrates for research on distributing visualization interfaces. In paper E, we present a framework for automatic distribution of visualizations and UI components across multiple heterogeneous devices. The paper contributes with; (1) a design space that considers specific aspects and relationships of interactive visualization interfaces (cf. Figure 7.1), (2) heuristics for the development of distribution algorithms, and (3) a system implementing the heuristics using the Vistrates platform. Furthermore, the paper provides a user study showing that the automatic approach produces similar quality distributions as manually created by the participants. A key strength of Vistrates is that
Figure 7.1: The design space for distributing interactive visualization interfaces presented in paper E.

most aspects of the design space are directly accessible in the pipeline compromising the visualization interface. For example, visualization type, data sources, and encodings are all available through the Vistrates framework. In addition, the underlying Webstrates API can deliver the necessary information about the connected devices. Thus, Vistribute is an example of the type of research that Vistrates enables. For more details on the automatic distribution of visualization interfaces, I refer to paper E.

There are still several aspects regarding multiple heterogeneous devices that warrant future research. Two of these are: (1) empirically investigating when and how such scenarios occur or when they are beneficial in the analysis of data, and (2) extending the methods proposed by Vistribute to multi-user scenarios. The Vistrates platform and the insights from Vistribute are obvious foundations for exploring these objectives.

7.2 Literate Analytics

As described in Section 2.4.2, literate computing has become a core technology in the toolkit for data analytics—even entire organisations are starting to build infrastructure around the use of computational notebooks [300]. Combining textual narratives with executable code in a document format has proven to be highly suitable for supporting human comprehension of computing. Thus, research has started to explore different aspects of literate computing and narration to further support data analytics and enhance explainability. While the machine learning communities address aspects such as explainable AI [134] and opening the black box [173], which is about the inner workings of statistical models, my research suggests that transparency of all steps related to data utilization is another important aspect of explainability. Similar findings have also been described in related studies on data utilization [34, 87, 159, 236, 240]. Thus, related work has explored aspects such as live documents [307], interactive web-based articles [68], handling messes in computational notebooks [125], and supporting multiverse analysis results in research papers [85]. All aspects that specifically address how we capture and present data analytics.

As described in Paper D, we argue that InsideInsights is a step towards a new analysis paradigm named literate analytics. Much like the focus in literate programming was to support human comprehension of programming [171], the focus in literate analytics is to support human comprehension of data analytics.
7.3 Towards New Domains

Concretely, InsideInsights adopts the hierarchical structure from literate programming and integrated narrative and executable code from literate programming. The goal of literate analytics is to capture the entire analytic rationale including data insights, methodology description, and interpretations during the analysis process, and thus not just as an afterthought. Knuth described how thinking about human comprehension of his programs while writing them also made him develop better code. I conjecture that the same is the case for data analytics. By thinking about aspects of transparency during an analysis, we are forced to consider, critique, and explain our own approach and results. As literate analytics is not yet an established concept, there are still several potential avenues of future research. For example, how can we support completeness of the analysis documentation. A participant in the expert review suggested to explicitly support what, how, and why annotations in the hierarchy. Besides allowing reviewers to explore a certain perspective of the analysis, it can also support the analyst in reasoning about which parts of the analysis are yet to be properly explained. On a high-level, exploring what literate analytics should exactly incorporate (and how to support it) is another potential future research direction.

7.3 Towards New Domains

While the work presented in this dissertation is inspired by all three domains, the research contributions has mainly used the DBA as a reference due the availability of data and human resources. However, similar tasks and challenges exist in the other two domains as well. In particular, the endeavor of establishing a large share data repository that transcends organisational boundaries and the need to establish common frames of reference for data utilization also exist in healthcare and at Grundfos. Thus, the Vistrates and InsideInsights systems also has the potential to make an impact on how data is utilized in these domains.

We also conducted a series of interviews at Grundfos with the same approach as described in Chapter 5. But navigating a large organisation can be very challenging and you often end up exploring avenues of inquiry that present limited use in research. Several of the participants in these interviews did not yet utilize data in any significant way, but rather they expressed desires to start using data and challenges in doing this. While these interviews did not match the intended focus, they still confirmed that the challenges presented in Section 1.4 exist within other organisations as well.

In the healthcare domain, patient flows and treatment records can also be described as temporal event sequence data. Thus, it has been the long term plan in my PhD project to explore the methods presented in 4.2 on this type of data as well. First of all, because it is an opportunity to improve the methods and provide validation for their usefulness. And secondly, because enabling this type of analysis in the healthcare domain has the potential to improve how experts reason about treatment pathways and support their decision making. The analysis
of temporal event sequence data has the potential to make very concrete societal impact. Thus, as highlighted in this section, there exist rich opportunities for exploring the presented work in other domains as well, which can improve both research and provide societal value.

7.4 The Evolving Research Agenda

The outset of my PhD project was to build interactive systems for domain experts rooted in the traditions of visual analytics. Throughout my three year project it became increasingly apparent that collaboration is an essential part of real-world data analysis. Thus, it seems reasonable that collaboration should be a core focus in visual analytics research as well. While the initial research agenda [69] for visual analytics stressed the importance of collaboration, my investigations suggest that a similar emphasis does not exist in current research literature. However, as visual analytics technology has matured, it seems worthwhile to now put a stronger focus on collaboration—i.e. the visual analytics research agenda should evolve in a similar fashion as my project has. Further exploring methods for tool integration and people involvement as described in Section 5.3.2 is an excellent starting point for doing this.

The Vistrates and InsideInsights systems are only two examples of research that resides in the outer levels of Figure 5.1. Still, the systems can serve as foundations for future research. The main limitation of Vistrates is currently scalability—both in terms of computational power and in terms of challenges with the growing number of component types and ways to compose pipelines. Easily distributing computations, supporting multiple programming languages, and user support for constructing pipelines are all potential avenues of future work. In addition, the systems presented here exemplify that by exposing internal state (through an API or through state specifications), it is possible to design system support for integrating different tools. This is an implication that transcends the specific component model of Vistrates. InsideInsights may just as well be built on top of another analytical system, or a combination of multiple applications that all expose a way to retrieve and reactivate internal states. Thus, a potential direction of future work is also to explore such cross-tool interfaces, that allow multiple analysts to utilize different tools while developing shared analysis results that are not detached from the systems that produced them.

Involving multiple people with different expertise and utilizing existing knowledge capital are central aspects for successful adoption of data analytics. At least, the investigations presented here suggests as much. Incorporating domain expertise is not only about designing interfaces for users with little knowledge of data science methods, but also about supporting users in sharing and combining the knowledge each individual posses. I have presented one approach that allows multiple collaborators to have different points of entry into the same analytic work. Still, the proposed system has both usability limitations and conceptual limi-
tions as presented in paper D—limitations that can inspire future work into the involvement of multiple people in data analytics. In particular, how to design methods that strike a balance between the freedom required to explore new ways of analyzing data and the requirement to create the necessary documentation without imposing a too large burden on the users, is an interesting question.
8 | Conclusion

As data analytics is becoming an increasingly important part of society, public institutions and industry alike face many challenges with effective data utilization that generates value. In this dissertation, I have explored challenges with the use of data analytics in the intersection between real-world domains and research. My initial focus was on developing visual analytics methods that enable domain experts to conduct exploratory analysis of large data collections. This work resulted in novel methods and tools for analyzing real-world data in domains such as production, healthcare, and public business administration. Through repeated interaction with domain experts and development of interactive prototypes, it became apparent that the adoption of data analytics within an organisational context relies on the ability to exploit existing knowledge capital—knowledge which is often distributed across multiple people. Thus, my focus evolved to include expanding aspects of collaboration in visual analytics and the design of systems that inherently support collaborative workflows. As stated in the introduction, this dissertation embodies the sentiment that it is rarely a single person’s job to be an expert on all areas related to data utilization, thus the adoption of advanced data analysis methods depends on the degree to which our technologies afford collaboration. My work highlights the need to put a stronger emphasis on collaboration in visual analytics research as leveraging mixed expertise in data analytics remains a central challenge for real users. Furthermore, my work shows how careful system design can support several of the collaborative challenges that real users currently face in a way that augments existing analysis approaches. This includes utilizing domain expertise and supporting shareability, replicability, and explainability in data analytics.

To summarize, this dissertation has presented both conceptual, empirical, and technical contributions that aim to bridge the gap between the opportunities advanced data analytics promise and the reality domain experts currently face. In Chapter 4, I presented novel analysis approaches and visual analytics prototypes that have the potential to enable domain experts analyze large and complex data. Among others, the contributions of this work include sequence aggregation using composite event learning and the probabilistic event tree visualization. In Chapter 5, I presented empirical findings from our interview study focusing on collaboration in data work. Besides extending current knowledge of data utilization, the contributions of this work include potential implications for visual analytics research—such as the importance of mixed-expertise collaboration and the need to focus on multiple levels of data analytics integration. In Chapter 6, I presented both conceptual and technical contributions in the form of design con-
cepts and interactive systems that address the challenges found in Chapter 5. The Vistrates platform presents a component model for data analytics that supports both synchronous collaboration on multiple abstraction levels, sharing and re-composition of atomic analysis parts, and support for heterogeneous devices. The InsideInsights system bridges data analytics with elements from literate computing, provenance tracking, and data-driven storytelling in order to further support sharing and involvement of people with mixed expertise in the analysis process. These systems present general approaches for supporting collaboration that can effectively encapsulate existing analytics solutions. Thus, the proposed approaches have the potential to bridge the gap between the advanced data analysis methods commonly found in research and the needs for tool integration and people involvement found in real-world organisational contexts. As presented in Chapter 7, the work presented in this dissertation also opens several avenues of future research. This includes research into multiple heterogeneous devices in data analytics, the concept of literate analytics, and how to further support mixed expertise collaboration. In general, my work suggests that visual analytics research should go beyond the design of specific analysis tools targeted at a single human. In addition, the presented systems also directly support future research. Our own work on distributing visualization interfaces in multi-device scenarios is an example of how research can directly utilize and extend the work presented here.
Part II

Publications
Clear Visual Separation of Temporal Event Sequences

Andreas Mathisen, & Kaj Grønbæk

Department of Computer Science, Aarhus University, Denmark

Abstract

Extracting and visualizing informative insights from temporal event sequences becomes increasingly difficult when data volume and variety increase. Besides dealing with high event type cardinality and many distinct sequences, it can be difficult to tell whether it is appropriate to combine multiple events into one or utilize additional information about event attributes. Existing approaches often make use of frequent sequential patterns extracted from the dataset, however, these patterns are limited in terms of interpretability and utility. In addition, it is difficult to assess the role of absolute and relative time when using pattern mining techniques.

In this paper, we present methods that address these challenges by automatically learning composite events which enables better aggregation of multiple event sequences. By leveraging event sequence outcomes, we present appropriate linked visualizations that allow domain experts to identify critical flows, to assess validity and to understand the role of time. Furthermore, we explore information gain and visual complexity metrics to identify the most relevant visual patterns. We compare composite event learning with two approaches for extracting event patterns using real world company event data from an ongoing project with the Danish Business Authority.

9.1 Introduction

Gaining informative insights from temporal event sequences is a challenging task in many real world domains. As shown in Figure 9.1(b), data volume and variety render most sequences unique as evident in a large collection of company event data. While the analysis of temporal event sequences is well-studied within both the visualization and data mining communities [121, 179, 200, 238] many challenges persist. Existing visualization techniques are often inadequate without appropriate aggregation of the data, since simply visualizing multiple raw event sequences will not provide interpretable information. However, defining meaningful aggregations is also difficult as described in recent work by Liu et al. [199]. A common approach is to extract frequent sequential patterns, but these techniques often yield an overwhelming number of subsequences on real world datasets. This makes the results unsuitable for manual inspection and it can be difficult to assess
9.1. INTRODUCTION

The relevance of a derived pattern when inspecting it in isolation, Liu et al. [199] therefore propose an algorithm to compute branching patterns that describe the most common event sequence flows. This automatic search shares a similar goal to the manual approach of Monroe et al. [219], where a series of user-specified simplifications lets the analyst arrive at a simpler representation of the major flows in the data. While the manual approach does not scale very well, the automatic search suffers in terms of interpretability, since it is hard to assess the quality of the derived patterns. Previous work [109, 318] have therefore shown promising results when using outcomes to analyze temporal event sequences. In this work we also use sequence outcomes to both reason about pattern quality and to define visualizations that allow the analyst to identify critical flows and to assess pattern quality. To efficiently use temporal event sequences in decision making processes, it is important to leverage both data mining and visualization techniques.

The motivating use cases for the work presented in this paper comes from an ongoing project with the Danish Business Authority (DBA), which maintains historical registration data and financial statements for more than 1.5 million Danish companies. The registration data is essentially temporal event sequences containing events such as changes to business type, name, accountants and board members. The domain experts at the DBA are not data scientists and therefore they lack automated tools to systematically gain insights from their data. Currently their search for suspicious behavior is started from previous cases, hence they often have outcome labels for a subset of the companies, i.e. outcomes describing bankruptcy or fraudulent behavior. Using our methods, the aggregation in Figure 9.1(b) is automatically transformed to the arguably better aggregation in Figure 9.1(c). Analysts can now separate the critical flows (dark) from the less critical flows (light) w.r.t. to the outcome and get insights into how different events affect the outcome probability.

Pattern mining algorithms are often ill suited when the order in which certain event types occur is irrelevant. In such scenarios, it can be advantageous to group events into a single event prior to the application of any aggregation algorithm. In our work, we automatically group events into higher level composite events since this drastically reduces the number of unique event sequences and thereby enables the simpler aggregation in Figure 9.1(c). Our approach is inspired from the work of Du et al. [85], who recently presented a series of strategies for sharpening analytic focus when analyzing temporal event sequences. In this paper, we use clustering to find similar event sequence windows and thereby replace the old event types with new composite events.

The event tree visualization (Figure 9.1) is based on the visualization presented in [219, 319]. However, given sequence outcomes, we also augment the various paths with outcome probabilities, which allows the analyst to assess the relevance of a pattern. Furthermore, we empirically validate our approach using information gain and visual complexity metrics. These metrics can also be used to automatically search for interesting views since they describe visual separa-
Figure 9.1: An event tree with few event types and few event sequences (a) that shows how to read the visualization. The event tree in (b) shows a large event sequence collection without any simplification and the event tree in (c) shows an event tree of the same dataset using composite events and outcome percentages encoded in the transitions between events. The y-axis is percentage of the total number of sequences and the x-axis is time.

Section w.r.t. to the outcomes. We compare our approach with two versions of the branching pattern mining technique presented in [199] - the original version and a modified version where the outcome is used to rank the events. Finally, we propose several future research directions for supporting the understanding and assessment of patterns derived from temporal event sequences.
9.2 Related Work

Visualizing and retrieving insights from temporal event sequences is not a new research focus due to its various application domains such as electronic health records \[109, 219, 319\], different types of clickstreams \[180, 200\] and company behavior, as presented in this paper. More domains will likely surface in the future as the methods to analyze event sequences improve. Currently, the major challenge in utilizing temporal event sequences for decision making is to cope with volume and variety, which is especially difficult to cope with when the data does not fit into standard tables formats as used in most machine learning scenarios.

For these reasons, Wongsuphasawat et al. \[319\] and Monroe et al. \[219\] have explored event sequence simplification based on user specified queries together with a custom visualization based on the icicle plot \[177\] to display common event paths and average timespans. Zhao et al. \[330\] proposes a matrix based visualization organized in a wave to provide overview of different web traffic patterns. These methods show promising results when the number of unique event types are manageable to an analyst. In our work, we use the same visualization as in \[319\] to display common event paths.

Several alternatives to user driven simplifications have been proposed to support the analysis of temporal event sequences. Prominent among these methods is frequent sequential pattern mining \[179, 200, 238\], where common subpatterns with a minimum level of support are extracted and visualized to the user. However, Liu et al. \[199\] recently pointed out several limitations with these approaches, including limited interpretability and utility. Another approach is to cluster common sequences \[305, 308\] or search for similar sequences based on a reference sequence \[84\]. While complete unsupervised clustering of entire sequences have shown promise in certain domains, it can be difficult to assess the relevance of a clustering result and unimportant events might have an overly large impact.

In the work by Du et al. \[84\], the goal is to arrive at a given outcome by identifying appropriate next actions. In general, sequences outcomes are a powerful tool for narrowing down relevant patterns and provide context to the user. In the work by Gotz et al. \[109\], users continuously pick event milestones and review how they affect the sequence outcome in the visualization. In our work, we use a similar approach to visualize the outcome probabilities within the different event flows. In the work by Wongsuphasawat et al. \[318\] event sequences are summarized to provide all possible paths to a given outcome and Lam et al. \[180\] use session outcomes to provide the user with context. Related to outcome analysis is cohort comparison and in many cases the two problems can be modelled in the same way. Krause et al. \[174\] provides a visual tool to efficiently extract relevant cohorts from a database. Malik et al. \[205\] proposes a system where multiple statistical significance scores for all subpatterns of a certain length are computed and visualized to the user, but their approach does not provide overview of multiple flows.

Event sequences are especially difficult to analyze when events are not only
ordered but also timestamped, since it can be a complex task to assess the role of time. Numerous approaches have been proposed to visualize events over time, e.g., in the form of cloudlines \cite{176}, temporal summaries \cite{202} or visualizations to explore timespans in known processes \cite{202}. In our work, time is mainly used to reason about which events are close enough to be combined into a composite event. Furthermore, Wongsuphasawat et al. \cite{320} and Monroe et al. \cite{220} also report on challenges with similarity measures for querying event sequences and specifying time interval and absences queries.

Liu et al. \cite{199} recently presented the notion of branching patterns (Core-Flow), which are extracted patterns that describe all event sequence flows instead of only frequent subsets of the event sequences. In our work, the goal is also to extract relevant branching patterns w.r.t. to sequences outcome. For this reason, we present and investigate a modified version of the algorithm presented in \cite{199} that uses information gain, traditionally used to infer decision trees \cite{247}, to reason about appropriate milestones, i.e. to generate an event decision tree. The composite event learning method is inspired from the work of Du et al. \cite{85}, who recently presented a series of strategies for sharpening analytic focus when analyzing temporal event sequences. Among others, they mention that it can be beneficial to group events based on time windows and replace all events within a given window with, e.g., the most frequent event type.

\section{Company Investigation}

The motivating domain for the methods presented in this paper is company investigation in collaboration with the DBA. While the methods we present generalize to other types of temporal event sequences, we will in this section briefly describe this domain in order to provide intuition prior to the technical details. The DBA maintains several types of data about more than 1.5 million companies in Denmark. This include a registration database, where companies report changes to important information about their business. This include changes in important relations, like accountants and board members, and changes to basic company information, like business type and name. In this work, we model the registration database as temporal event sequences, since all changes are timestamped. In total the database contains more then 50 million events. Removing all sequence information gives little insight, since looking at a single change, e.g., in the board, can both be a positive or a negative change. However, the order of events is sometimes also irrelevant in shorter time windows while major time gaps are still informative. Furthermore, the quality of the data is unknown, since certain changes are not mandatory to report immediately, which means sometimes the order of events introduce more noise than clarity. For these reasons, we introduce the concept of composite event learning in the following section.

Domain experts at the DBA usually start their investigations based on knowledge from previous cases, hence relevant labels can be defined for a subset of the
companies. These labels include first of all whether a company went bankrupt or otherwise have been forced to close, but also different types of fraud. In this paper, we model this information as a sequence outcome, which will be the driving factor in the analysis scenario we are trying to support. The interactive system we designed, enables domain experts to efficiently visualize relevant event flows w.r.t. to outcome in large sequence collections. For experimental purposes we only use event data prior to 2014 in the evaluation in order to reason about future outcomes, i.e. which companies will close in the period of 2014-2017. Such experiments will also be important in future use scenarios, when domain experts are using the system for decision making.

9.4 Learning Composite Events

In this section, we define the notion of composite events and describe a concept to automatically learn these. As described previously, the main motivation behind generating composite events is to reduce the variety of temporal event sequences which makes it possible to aggregate sequences that would otherwise be unique. The idea behind composite events is therefore to find collections of similar sub-sequences and replace these with new high-level events. An example from the DBA data is the beginning of a new company. Since new companies usually go through very similar initial processes, their beginning can often be replaced with a single event, which, e.g., includes the addition of several board members and accountants as well as updates to core company information like name and main business type. This furthermore allows for the identification of different types of beginnings where, e.g., there size of the initial board can be important. The following notions will be used throughout the paper:

**Temporal Event Sequence:** A sequence of event tuples \((e_i, t_i)\), where \(i\) is the positional index in the sequence, \(e_i\) is the event type and \(t_i\) is the timestamp. In certain scenarios the sequence contains event triples \((e_i, a_i, t_i)\), where an event also have an attribute \(a_i\).

**Composite Event:** A grouping of several events from a sequence into a single complex event.

Note, that in our definition of temporal event sequences, two events from the same sequence are allowed to share the same timestamp. There exist numerous approaches for retrieving common patterns in temporal event sequences that can be used to find composite events including frequent pattern mining [121], temporal abstractions [275] or user-specified find-and-replace methods [219]. However, not all methods can deal with events that share the same timestamp. In this paper, we use a simple bucketing by time period approach, coined as a strategy to simplify temporal event sequences in [85], as the foundation for defining composite events. The sequences are divided into equal time segments that each will constitute a
composite event. By counting event type occurrences within each segment (and potentially attribute occurrences) conventional clustering methods can be used to define similar segments and thereby find composite events. Concretely, for window size \( w \) and number of clusters \( k \) we do the following:

1. **Segmentation**: Divide each temporal event sequence into equal time segments of size \( w \).

2. **Feature Generation**: Count event type occurrences in each segment generating a feature for each event type, effectively ignoring sequence order within the window segments.

3. **Clustering**: Partition all segments into \( k \) groups using the k-means clustering algorithm \[13\]. The \( k \) groups constitute the composite event types.

This realization of the three steps in the composite event learning concept introduces the challenge of finding the optimal window size \( w \) and number of clusters \( k \) for some notion of optimal, which in our use case is an event tree with good separation of the outcomes and good predictive power beyond the dataset. If an overly large \( w \) and a small \( k \) is used, the resulting aggregation will potentially be an oversimplification, which can result in an event tree with low separation of the outcomes. On the other hand, if an overly small \( w \) and a large \( k \) is used, the following aggregation will potentially be overfitted to the dataset and have little predictive power. We will describe sequence aggregation methods and quality metrics in the subsequent section. The exact choice of \( w \) and \( k \) depends on the use case. The time between events is usually larger in the company event data compared to, e.g., weblogs or medical records, hence appropriate choices of \( w \) are likely also larger for this use case.

The proposed realization is only one way to find composite events and there exist multiple methods for implementing the three steps. Dynamic window sizes or the addition of more features, such as average time between events or features based on event attributes, could be immediate extensions of our realization. When introducing increasing complexity in the mining approach, though, complexity is also introduced in the visual interface, since additional information about feature types and varying window sizes needs to be conveyed to the user. Different clustering algorithms or alternative similarity measures can also be used in step 3. A study on similarity measures for text document clustering using k-means showed that the euclidean distance can be outperformed by alternative measures in this domain \[140\]. In text document clustering features are usually also frequency-based, hence it seems worthwhile to investigate different similarity measures for event frequencies in future work. However, it is not the main focus of this paper to pick the optimal segmentation, feature generation or clustering methods, but rather how to utilize the concept of composite event learning to generate visualizations that provide relevant insights about temporal event sequences.
9.5 Sequence Outcome, Aggregation & Quality

In the following, we will describe the three different temporal event sequence aggregation methods investigated in this paper. Furthermore, we will define how sequence outcomes are used to encode flow probabilities and to score the overall pattern. Sequence outcomes are important in many event analysis scenarios and is usually the occurrence of a certain event type. An example is health outcome analysis as described in [109], where analysts and epidemiologists study data to understand what factors influence certain health outcomes. As previously described, outcome is also the driving factor behind the motivating analysis tasks at the DBA. In this paper, we define an outcome as a special event occurrence.

**Sequence Outcome:** An event sequence outcome is a special event tuple \((o_i, t_i)\), where \(i\) is the sequence position, \(o_i\) is the outcome type and \(t_i\) is the timestamp of the outcome.

Note, that any event type can therefore be thought of as an outcome and in the DBA case the relevant outcomes include for instance bankruptcy and different types of fraud. Given an outcome, the sequences can be divided into two groups - the sequences that include the outcome and those that do not. While the basic analysis scenario constitute a binary distinction, the methods presented in this paper can be extended to multiclass or numeric outcome scenarios.

9.5.1 Sequence Aggregation

We consider three different methods to aggregate temporal event sequences that are suitable for visualization - one method where the simplified sequence collection is aggregated and two methods where descriptive patterns are extracted from the raw event sequences. The first method computes a hierarchical structure of all unique event paths. The second method is the branching pattern algorithm (CoreFlow) proposed by Liu et al. [199], which computes the most common flows. The last method is a modified version of the branching pattern algorithm, where we use entropy based information gain using the sequence outcomes to rank the events instead of using the overall frequency. Entropy is often used in decision tree algorithms to greedily choose the best attributes to branch on. Entropy and information gain will be explained in section 9.5.2. The result of all approaches is a hierarchical tree structure as described in [313].

**Simplified Aggregation (SA):** First, all sequences are simplified by computing composite events as described in Section 9.4. Sequences with the same prefix will follow the same path in the data structure, and when two sequences no longer consist of the same series of events the data structure will branch. The hierarchical structure contains a path for each unique event sequence, hence in cases with high variety there will be little to no aggregation. In our approach we root the aggregation at the first event in each sequence, however, it is also possible to
root the aggregation on, e.g., the first occurrence of a certain event type or other user-specified alignment points [219].

**Most Common Pattern (MCP, CoreFlow [199]):** The Rank-Divide-Trim algorithm presented in [199] recursively ranks all events w.r.t. frequency (the number of sequences an event occurs in), divides the sequences based on the highest ranking event type (where ties are broken by minimum average sequence index) and then trims the sequences up to the first occurrences of the chosen event. This approach creates the same hierarchical tree structure as the full aggregation, however, only including the most frequent occurring events.

**Most Separating Pattern (MSP):** This method uses the same rank-divide-trim procedure as for the most common pattern but with a different ranking function. By using entropy-based information gain with the sequence outcomes to rank the events, the resulting hierarchy will include the most separating events w.r.t. the outcome. This method is similar to building a decision tree.

All aggregation methods can be used either with or without the event sequence simplification described in section 9.4. In section 9.7 we will provide a comparative evaluation of the SA method, which uses sequence simplification, and the two pattern extraction methods on the raw event sequences, with data from the DBA use case. The evaluation also serves as an example of how a potential evaluation can be done in other domains in order to find the best combination of event sequence simplification and aggregation method.

**9.5.2 Quality Metrics**

Sequence outcomes provide a way to compute pattern quality empirically. Since the outcome of interest is often heavily outnumbered in real world datasets, i.e. only a subset of the sequences contain the outcome of interest to the analyst, it can be difficult to argue about pattern quality for two reasons. First, if the dataset is used to predict outcome, you get a very high accuracy simply by repeatedly guessing on the dominant outcome, which in the DBA domain is the same as always predicting no bankruptcy. Second, the goal of an analyst is not only to get a list of the next instances that will have the interesting outcome, but also to investigate which events or combinations of events that have an influence on the outcome in order to limit the number of instances in an investigation and later fuse the gained knowledge with other sources of data. For these reasons, we use entropy-based information gain to measure pattern quality which is commonly used to reason about appropriate splits when building decision trees. Furthermore, we also use two metrics describing visual complexity presented by Monroe et al. [219].

**Entropy:** Entropy is a measure of sample homogeneity. If a sample is completely homogeneous, i.e. all sequences lead to the same outcome, the entropy is zero and
if the sample is equally divided, i.e. contains the same number sequences leading
to each of the outcomes, the entropy is one. The entropy $E$ of a sample $S$ is
calculated as

$$E(S) = -\sum_{i=1}^{e} p_i \log_2 p_i$$

where $p_i$ is the probability of outcome $i$.

**Information Gain:** The gain in information is the decrease in entropy after a
dataset is split into a number of samples. The information gain $IG$ of a sam-
ples divided into the partitioning $A$, where $A$ in our case describes the resulting
partitioning given by the extracted event paths, is calculated as

$$IG(S, A) = E(S) - \sum_{v \in A} \frac{|S_v|}{|S|} E(S_v)$$

where $v$ is a unique path to the point of splitting in the event tree. Effectively, we
will use $IG$ to reason about the initial outcome distribution versus the outcome
distributions of the chosen samples in our event tree. The samples can be defined
by choosing a minimum level of support, which we denote as the point of splitting.
This can, e.g., be the subgroups just before the samples become smaller than 5%
of the total number of records. This notion is similar to the concept of minimum
level of support from frequent sequential pattern mining. If the minimum level of
support is simply set to 0%, the leafs of the event tree will be used as the chosen
samples.

**Visual Complexity:** We compute visual complexity using two measures pro-
posed by Monroe et al. [219]. (1) The average height of the vertical elements
as percentage of the display height, i.e. the size vertical bars describing the dif-
ferent events in the event tree as percentage of the total number of records. (2)
The number of elements in the event tree. These measures builds on two central
notions of visual complexity; separability and information density. Few visual
elements means low information density and large visual elements are easier to
distinguish from each other, hence reduce perceived complexity.

Both information gain and visual complexity are important metrics. High
information gain alone does not necessarily imply good generalization beyond the
dataset, since the simple split into raw event sequences, i.e. multiple samples of
size 1, will give the maximum information gain. Lowering the visual complexity
is therefore important for both interpretability as well as generalization purposes.
Visual complexity is similar to the generalization heuristics of decision trees, where
smaller trees should in theory generalize better.
Figure 9.2: The screenshot shows our prototype tool with dataset 1 (cf. Table 9.1). The main view (a) shows the probabilistic event tree with composite events and outcome percentage encoded in the transitions between events — white means that zero companies have closed and black means that all companies taking this subpath have closed cf. (e). Hovering a composite event in the hierarchy will show the corresponding outcome statistics (b), highlight the composite event in (c) and display a time histogram (g) that shows when the companies went through this event (red are closed and blue are normal). The components of the composite events can be inspected using the aster charts (c), which show the composite events that the user is currently interested in, as well as the corresponding event legend (d). Users can also specify the size of the segmentation window, the number of composite events to compute and limit which event types to use (f) as well as load previous results.

9.6 Visualizations & Interactions

We designed a prototype system that is able to visualize probabilistic event flows of large data collections using any of the aggregation methods described in section 9.5.1. When the SA method is applied, users can currently define the segmentation window size ($w$), the number of composite event types ($k$) (cf. Section 9.4) and limit which raw event types to use (Figure 9.2(f)). While this allows the analyst to compute any desired aggregation, it provides little support in choosing suitable parameters for a given dataset, hence we discuss how this interaction can be improved in Section 9.8. The prototype has two central views, one for the event tree, visualizing also outcome probabilities at the event transitions, and one to investigate the composite events. In the following, both views will be described in detail as well as central interactions.
9.6. VISUALIZATIONS & INTERACTIONS

9.6.1 Probabilistic Event Tree

Most tree visualizations can be applied to display aggregated event patterns. In our prototype, we use the hierarchical time visualization presented in [319], including also outcome probabilities at each event transition (Figure 9.2(a)). The x-axis represents time, i.e. average time between events and average time to the root, which in this case is the start of the extracted sequence time periods. The y-axis represents percentage of the total number of sequences. Colored vertical elements represent a specific type of event or a composite event, i.e. the large yellow element shows that almost 60% of the sequences start with composite event 3 and then around 30% of those sequences continue with composite event 1, 2 or 14, while the remaining sequences end. The composite events can be inspected using the aster charts (Figure 9.2(c)) which will be described in the following section, i.e. the color of the vertical element corresponds to the color of the composite event number. Relevant composite event descriptors (aster charts) can be added by clicking the composite events in the event tree.

Outcome percentages of the individual subpaths are encoded with a continuous black and white scale in the transitions between events, where 0% is white and 100% is black (Figure 9.2(e)). This allows the users to quickly identify the relevant subpaths for a given analysis task and see how the outcome probability changes with each event on a path. Furthermore, the event tree visualization includes two simple but powerful interactions. First of all, users can zoom on both axes with mouse scrolling, which allows the analyst to focus on relevant subpaths. Secondly, users are provided with a tooltip when hovering an event bar with basic statistical information about the outcome (Figure 9.2(b)). Currently, we also report on future outcome, i.e. outcome beyond the current data extract used to generate the event tree, which allows the analyst to assess whether the current subpath generalizes. The data extracts will be explained in Section 9.7. We will also discuss how to extend the interaction opportunities such that users can experiment with different time periods to investigate how well certain patterns explain future outcomes.

When an event in the event tree is hovered, users are also provided with a time histogram (Figure 9.2(g)) that summarizes the timestamps of the events in the chosen subgroup, i.e. when each sequence went through the hovered event in absolute time. Blue summarizes one outcome (in this case companies that are still alive) and red summarizes the other outcome (closed companies) – the histogram then shows both blue and red bars as two overlays for easy height comparisons. First of all, the time histogram allows the analyst to identify whether certain events occurred at certain points in time and, secondly, it allows the analyst to identify whether the time distribution is different for the two outcomes. In cases where the distribution is different, the time histogram can potentially help the analyst narrow down interesting sequences even further by, e.g., first investigating the blue instances that occur closest to the red instances.
9.6.2 Composite Event Inspection

The components of the composite event types can be inspected using (1) the aster charts (Figure 9.2(c)) and (2) the event legend (Figure 9.2(d)). Each aster chart represents one composite event type, which in our case is a resulting cluster of the k-means algorithm. A slice represents one original event type and the color of the slice matches the event legend. The width of a slice is the feature mean, i.e. average number of event occurrences within each segment, proportional to the sum of all feature means within a cluster. The height of a slice is the feature mean proportional to the means of the same feature in the other clusters. These compact visualizations of the cluster centers are suitable in this scenario since several feature means (event counts) are zero, i.e. those events did not occur within the segments and are therefore irrelevant in order to understand a certain composite event. To make the circle charts even simpler, we combine features with very low means into one category, which can then later be inspected if desired, e.g., to search for outliers. Users can quickly compare multiple composite events for common event types and average number of occurrences by looking at both color, height and width of the slices. As an example, if a slice has a height of 50% it means that there is a composite event where this type of event occurs twice as frequently. Hovering the aster charts provides the user with statistical information about the individual clusters.

The event legend (Figure 9.2(d)) describes actual event types and is linked with the other views, i.e. it shows only the event types of the composite events that the user is currently investigating. Hovering the event legend provides the user with overall statistical information about the individual event types, which can be used to provide context to the statistical information about the composite events. The composite event inspection is work in progress and we aim to do a user study to evaluate how to optimally visualize these (cf. Section 9.8).

9.7 Evaluation

In this section, we will first present a comparative evaluation of the different aggregation methods presented in Section 9.5.1 on three different datasets from the DBA using the quality metrics described in Section 9.5.2. We will also describe a use case example to illustrate the practical implications for domain experts at the DBA. Statistics for the chosen datasets are shown in Table 9.1. The datasets represent three different business types (IT-service, Building Entrepreneurs and Retail). Before analyzing the event sequences, several choices have to be made regarding time. In this evaluation scenario, we use only the last 10 years of the event history for each company, since otherwise the event tree would be highly biased by the initial updates of a company. The investigated outcome of this evaluation is risk of bankruptcy, which describes that the company have been flagged in the database as either bankrupt, forced to closed or in a pre-state for either of the two. Table 9.1 also show the overall percentage of companies that
have experienced risk of bankruptcy. For the companies that do experience risk of bankruptcy, we use only the events one year prior to the outcome, since in a real analysis scenario domain experts want to identify companies worth further inspection some time before the outcome actually occurs. Furthermore, we only use data prior to 2014 in general, which allows us to reason about how extracted patterns generalize to the following years. For reference, Table 9.1 also show the general chance of picking a company that will experience risk of bankruptcy after 2014. The different choices regarding dataset preparation means that multiple parameters can be tuned and tested, which we will discuss further in section 9.8. The following results have been computed with a one week window size and 25 composite events for the simplified aggregation method.

### 9.7.1 Comparative Evaluation of Aggregation Methods

Table 9.2 shows the evaluation results for the three different aggregation methods on the three datasets with a minimum support of 1% for the subgroups. The MCP method provides the least information gain w.r.t. the outcome, which is not surprising since there is no guarantee that the most common event flows will also describe relevant flows w.r.t. outcome. However, the MCP method is the best for reducing visual complexity when measured using average height of

<table>
<thead>
<tr>
<th>Dataset 1</th>
<th>Dataset 2</th>
<th>Dataset 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of sequences</td>
<td>3475</td>
<td>4008</td>
</tr>
<tr>
<td>No. of events</td>
<td>30494</td>
<td>41151</td>
</tr>
<tr>
<td>No. of unique events</td>
<td>42</td>
<td>59</td>
</tr>
<tr>
<td>Business type</td>
<td>IT-Service</td>
<td>Building Contractors</td>
</tr>
<tr>
<td>Risk of bankruptcy (%)</td>
<td>8</td>
<td>58</td>
</tr>
<tr>
<td>Future pred. prec. (%)</td>
<td>3.5</td>
<td>2.6</td>
</tr>
</tbody>
</table>

Table 9.1: Dataset characteristics

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MCP</th>
<th>MSP</th>
<th>SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information Gain</td>
<td>0.003</td>
<td>0.151</td>
<td>0.153</td>
</tr>
<tr>
<td>Average Height</td>
<td>1</td>
<td>6.71</td>
<td>5.33</td>
</tr>
<tr>
<td>Number of Elements</td>
<td>58</td>
<td>67</td>
<td>36</td>
</tr>
<tr>
<td>Information Gain</td>
<td>0.431</td>
<td>0.713</td>
<td>0.744</td>
</tr>
<tr>
<td>Average Height</td>
<td>2</td>
<td>6.04</td>
<td>3.95</td>
</tr>
<tr>
<td>Number of Elements</td>
<td>69</td>
<td>84</td>
<td>55</td>
</tr>
<tr>
<td>Information Gain</td>
<td>0.037</td>
<td>0.262</td>
<td>0.246</td>
</tr>
<tr>
<td>Average Height</td>
<td>3</td>
<td>6.37</td>
<td>4.85</td>
</tr>
<tr>
<td>Number of Elements</td>
<td>57</td>
<td>66</td>
<td>37</td>
</tr>
</tbody>
</table>

Table 9.2: Evaluation results for the three different aggregation methods on the three datasets with a minimum support of 1%.
the visual elements. The MSP and SA methods provide similar information gains w.r.t. the outcome despite that the MSP method is using the outcome to generate the aggregation and the SA method is complete unsupervised. The SA method computes the fewest visual elements in the event tree. However, these elements are composite events that need further investigation and the SA method is therefore not necessarily superior in terms of visual complexity, especially since the MSP method computes larger visual elements. While the MCP method is superior in terms of visual complexity, the method does not let the analyst gain insights into how the outcome flows differ. Since the MSP and SA methods provide similar information gain w.r.t. the outcome, we will in the next section also explore how an event tree can assist the analyst narrow the search for companies that will experience the negative outcome in the future.

9.7.2 Use Case Example

In this section, we will describe insights an analyst can get when using the prototype tool with the SA method and dataset 1 as presented in Figure 9.2. First of all, the event tree shows that several companies perform very few updates in the DBA database. However, it is not simply the well-functioning companies that update their information since there exist both dark and light paths. Some of the most common composite events are 1, 2 and 14, which mainly consist of updates to business type, name and contact information. Furthermore, the two most common sequence beginnings are the composite events 3 and 0, which describes the start of a very light path and the start of a very dark path, respectively. The event tree allows an analyst to efficiently identify the most common negative and positive paths. By zooming and hovering the event tree and the composite event glyphs, the analyst can inspect exactly what the composite events are made of and identify related composite events as well as relevant subpaths. The time histogram shows that a lot of updates happens around 2008, which we later found out is because the DBA introduced new regulations at that time.

Besides summarizing how companies have behaved so far, the prototype tool also allows the analyst to reason about future outcomes. In the most troublesome event paths in Figure 9.2, at least 30% of the companies have experienced risk of bankruptcy. If an analyst employs this heuristic to narrow down the search for future troublesome companies, i.e. companies going through subpaths where at least 30% have had the negative outcome and with 1% minimum support, the resulting subgroups are presented in Table 9.3 for all three aggregation methods on the three datasets. The table includes both size of the subgroups, risk of bankruptcy percentage within the subgroup as well as the chance of picking a company from the current normal companies that will experience risk of bankruptcy in the future, i.e. prediction precision for the period 2014-2017. Note that the table does not say anything about prediction recall. Both the MSP and SA methods find larger interesting subgroups compared to the MCP method and they have higher prediction precision. While the subgroups of the MSP method are slightly
9.8. DISCUSSION

Table 9.3: Statistics when looking at subgroups with 1% minimum support and at least 30% with the **towards closure** outcome

<table>
<thead>
<tr>
<th>Subgroup</th>
<th>MCP</th>
<th>MSP</th>
<th>SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of sequences</td>
<td>469</td>
<td>745</td>
<td>617</td>
</tr>
<tr>
<td>Risk of bankruptcy (%)</td>
<td>1</td>
<td>34</td>
<td>30</td>
</tr>
<tr>
<td>Future pred. prec. (%)</td>
<td>12</td>
<td>13.9</td>
<td>17.2</td>
</tr>
<tr>
<td>No of sequences</td>
<td>3994</td>
<td>2576</td>
<td>2300</td>
</tr>
<tr>
<td>Risk of bankruptcy (%)</td>
<td>2</td>
<td>58</td>
<td>88</td>
</tr>
<tr>
<td>Future pred. prec. (%)</td>
<td>2.6</td>
<td>7.6</td>
<td>10.3</td>
</tr>
<tr>
<td>No of sequences</td>
<td>5544</td>
<td>7264</td>
<td>6425</td>
</tr>
<tr>
<td>Risk of bankruptcy (%)</td>
<td>3</td>
<td>43</td>
<td>42</td>
</tr>
<tr>
<td>Future pred. prec. (%)</td>
<td>17.2</td>
<td>18.9</td>
<td>20.5</td>
</tr>
</tbody>
</table>

larger, the SA method have higher prediction precision and thus generalizes better beyond the datasets even though the two methods provided similar information gains in Section 9.7.1.

Another interesting observation from Figure 9.2 is the growing group of companies starting with the composite event 17, which also can be categorized as a troublesome path. Several of the companies who also goes through this event in the light paths will experience the **risk of bankruptcy** outcome after 2014, i.e. the visualization can also be used to identify interesting subpaths that might hint future outcomes. The composite event 17 can best be described as a major overhaul of the company, since it includes both chairman, board member, business type, name and company form updates as its main components. We will discuss how time related experiments can be incorporated in the user interface in the subsequent section, such that an analyst can better identify which paths are currently the interesting ones.

9.8 Discussion

We have shown that combining events into composite events prior to sequence aggregation can provide a better separation of temporal event sequences w.r.t. sequence outcome in the real world domain of business investigation. While the data quality is unknown, the results suggest that the data can still be used for initial separation of the sequences, i.e. the companies, such that guesses about future outcomes are more informed and significantly better than random guessing. Even if an event sequence dataset is not rich enough to provide perfect outcome predictions, the proposed system can show what the most critical flows are and how well they generalize.

Immediate future work includes evaluations and improvements of the user interface with domain experts at the DBA. While the underlying algorithms generalize to multiclass and numeric outcomes, the current visual system is designed
for binary outcomes. It is therefore also worthwhile to investigate how the visualizations can be extended to multiclass and numeric outcomes. We believe the presented system will be very powerful as a complement to existing approaches, since the domain experts currently lack holistic views on the registration data. Furthermore, it will be interesting to either compare the results from this type of analysis with an analysis based on other datasources, e.g., financial statements, or fuse with other datasources in the feature generation for the composite events.

In general, composite event learning opens new possibilities but also poses several challenges. Understanding clustering results is inherently difficult, hence the average user will probably find it difficult to make sense of the high level events that are formed from several event types. Further investigations into how best to convey the components of a composite event are therefore necessary, i.e. the visualizations in Figure 9.2(c). Basic interactions like manual updates to a composite event or labeling of an event with a user-friendly name for future reference could be incorporated in future iterations of the system. Choosing appropriate parameters for the composite event learning, i.e. segmentation window size and number of clusters, is also a difficult task for the average user, thus it becomes important to show users how varying parameters affect both the resulting composite events and the quality metrics. Additionally, automatic suggestions for parameters that score well on the different quality metrics can be included in the user interface to support users that are less familiar with parameter tuning. Furthermore, we want to investigate how best to combine the different aggregations methods with composite event learning. Currently, we compute the full event hierarchy after the sequence simplification, but any of the other methods for extracting patterns can also be used after sequence simplification. This opens for the opportunity to compute multiple composite event candidates and afterwards reason about which ones are appropriate for a given analysis task by, e.g., extracting the most separating pattern and scoring it using the quality metrics. Multiple composite event candidates can, e.g., be based on different feature subsets, in cases where not all event types in a given window should influence the clustering, or a combination of both raw event types and composite events. If event types also have attributes – as in the domain of business investigation where, e.g., the event business type change also includes an attribute with the new business type – multiple candidates can be computed by replacing an event type with the corresponding attribute, if it is categorical, or include numeric attributes as features in the clustering.

When event sequences are not only ordered but also timestamped, several choices have to be made regarding proper data extracts for both pattern extraction and evaluation as described in Section 9.7. Future work about how different choices to these parameters can seamlessly be incorporated into the user interface is also interesting, such that history is continuously used to assess the relevance of the computed patterns. This also means that timely insights, i.e. information about absolute time as in Figure 9.2(g), should be incorporated at opportune steps in the overall analysis flow. For instance, in the business investigation domain, using information about when certain updates happened can narrow
down the subset of interesting companies even further compared to only using the event tree. We also believe that users should be able to fluently shift between different levels of abstraction, since an analyst then will be able to browse the overall patterns using the composite events to identify interesting subgroups and afterwards easily zoom in on the actual event sequences of the chosen subgroup.

Effective solutions to much of the future work discussed in this section are most likely domain specific, hence we would also like to apply our methods to different domains. While the exact match between simplification and aggregation method might change from domain to domain, or even from case to case, we believe the approaches presented in this paper generalize to other domains.

9.9 Conclusion

In this paper, we present the idea of composite event learning to simplify large collections of temporal event sequences prior to pattern extraction or aggregation. We compare our approach with a recent branching pattern algorithm [199] that computes the most common event flow as well as a modified version, where we use theory from decision tree construction to compute the most separating pattern w.r.t. sequence outcome. All methods are able to compute event hierarchies that are suitable for visualization. Evaluation results, using relevant pattern quality metrics, show that computing composite events is useful in the real world domain of company investigation and that the unsupervised aggregation method based on composite event learning is better for future outcome prediction compared to the supervised pattern extraction method of the raw event sequences.

We have also designed a visual analytics system prototype that incorporates the simplification and aggregation methods. The goal is to support domain experts at the Danish Business Authority identify critical event flows w.r.t. chosen sequence outcomes, such as bankruptcy. The system allows analysts to efficiently visualize separating flows, and thereby gain insights into how different composite events affect outcome probabilities, as well as inspect the components of the composite events. We also present a use case example that shows how the learning algorithms and visualizations combined can assist the domain expert in gaining relevant insights. Future work include user studies and ways to assist experiments with time in the user interface, which is important when the goal is to reason about the future.

Acknowledgements

This work was conducted in the DABAI project (IFD-5153-00004B) supported by the Innovation Fund Denmark.
Understanding Cooperation in Data Work -
Towards Computer Support for Cooperative
Data Work

Andreas Mathisen, Henrik Korsgaard, & Kaj Grønbæk

Department of Computer Science, Aarhus University, Denmark

Abstract

Retrieving and analyzing complex data to support decision-making is often associated with the work of data scientists and analysts. However, the contemporary push toward more data-driven approaches across multiple domains, means that knowledge workers are increasingly faced with the task of integrating elements of data analysis into their existing work practices. In this paper, we examine this as data work and explore how cooperation is central to success. We do this through an interview study of knowledge workers within the Danish Business Authorities. Based on the empirical work, we present findings on practices and challenges in cooperative data work. Among other, we report on how the additional dependencies that shared data repositories introduce results in an increased need for articulation and translation work as well as mixed expertise cooperation. We summarize the insights of our analysis in a discussion of potential implications for computer support. The presented work offers a much needed focus on data work in the wild and the important role of cooperation.

10.1 Introduction

Driven by recent advances in, and availability of, analysis and visualization tools society at large is rapidly becoming more data-driven [96, 237]. Organizations generate and collect more data than ever and the promises of value in big data and machine learning are motivating organizations to become more data-driven. As a consequence, knowledge workers with little to no formal training in data science increasingly face situations where they have to deal with topics from and integrate elements of data science into their everyday work activities. They collect, explore and use data generated across the organization, work with data outside familiar templates and applications, deal with data management, quality and model updates, collaborate with other parts of the organization and disciplines around data, contribute to the development of analytical models, and finally, anchor decisions and outcomes in data and data analyses. Along with others [40, 43, 95], we characterize this kind of work as data work.

As a result of this development, previous work has, among other, tried to characterize collaboration in open data analysis [62], uncover issues in enterprise and
intelligence data analysis and visualization [61, 159, 236, e.g.], examine how data transforms scientific collaboration [35, 72, e.g.], and understand how data workers cope with uncertainty in practise [43]. Parallel work discusses and explores the implication of the ubiquity of data and how we increasingly interact, not with applications and computers, but with data itself under the heading of human-data interaction (HDI) [71, 137, 222] and the related area of human computation, see [248]. Other work explores the large implications data-driven approaches have on work [192, e.g.] and the new kinds of work it might generate in the future [40]. While these studies have provided valuable insights, existing research has mainly focused on studying and developing tools for trained enterprise and/or research analysts and data scientists, or investigated the issues as a part of developing and evaluating tools and information visualizations with a focus on specific analytic tasks. Studies focusing on collaboration in data work are rare [128], with few exceptions [62, 159]; so is research that investigate how multiple actors coordinate, cooperate and collaborate in and around data analysis [236]. Crawford and Calo [73] point out that a current lack of empirical research on how AI and data science are applied and their impact on practices represents a significant blind spot.

Adopting data-driven approaches not only affects IT-departments and dedicated data analysts, but the organization and knowledge workers in general. Thus, making collaboration and cooperation central aspects of data work in the real world. Therefore, understanding cooperative data work in practice is crucial for the design of future computer support. To provide some answers and fill out the gaps in existing work, we conducted a case study at Danish Business Authorities (DBA). The DBA serves as an ideal case study since the organization presents multiple aspects of cooperative data work, that are also common in several other public and private organizations. These aspects include for instance employees with mixed backgrounds, several work tasks involving data and several forms of cooperation and collaboration between both employees and other organizations. Thus, our research goal is to uncover practices and challenges in cooperative data work as well as, to identity key elements of data work that affect cooperation.

In this paper, we present the findings of our study at DBA and discuss potential implications for computer support in light of related work on cooperation and data science. While a single study cannot uncover all aspects, we see our study as a first step towards a more complete picture of general cooperative data work, beyond the existing data science focus. Among other, we confirm conjectures from related work about the necessity of mixed expertise cooperation, the often non-linear process of working with data and the challenge of changing data. In addition, we also present findings regarding the challenge of translating between high-level concepts and concrete data when sharing and discussing data work, and on the dependencies that shared data repositories introduce. The insights presented here contribute to the recent interest in exploring data work from multiple perspectives in HCI, CSCW and HDI. It offers a much needed focus on how data work happens in practice.
10.2 Background

We start by looking at the concept of cooperative data work and data workers within the recent literature. Then, we present related work on collaboration in data analytics.

10.2.1 Conceptualizing Cooperative Data Work

Several works make distinctions related to the kind of tasks and the role that data play in general work activities. Furthermore, level of data science expertise and training in programming techniques is often highlighted when data work is described. Boukhelfa et al. [43] define data workers as non-professionals engaging in activities typically associated with data science: Acquiring, manipulating, characterizing, reasoning about, and presenting data. In their definition, the key differences between data work and data science lies in the lack of formal training and the frequency of the analysis activities, i.e. not being a primary task. Kandel et al. [159] distinguish between three analyst archetypes in their study of enterprise data analysis: Hackers, scripters and application users, suggesting that it is a matter of programming expertise. Thus, data workers can be defined as knowledge workers who from time to time engage in data science tasks, typically using applications and data sets prepared by other roles (e.g. hackers and scripters, and/or colleagues within the IT-department [51]). However, studies examining the role of automation and data-driven work offer a different perspective. These suggest that data workers are people who are embedded within larger socio-technical systems, where data and automated analysis plays a key role in shaping tasks and working conditions. This is discussed by Lee et al. [192] in their study of how drivers working for ride-sharing services experience automated task assignment and performance rating. Similarly, Irani and Silberman [146] describe people working on Amazon’s crowd-sourcing platform Mechanical Turk as a human computation resource, that can be integrated into other computational systems and solve sub-tasks allocated by Amazon’s platform. When characterizing what data work encompass, related work presents three different levels or relations between work and data:

First, working on data where data is the object of interest, attention and actions are directed at data, and activities produce data to be collected and analyzed. This is the focus of multiple studies on how analysts work and collaborate [62, 159, 236, e.g.], and in works proposing tools and process-models to support data-centric tasks [128 see]. The study of Boukhelifa et al. [43] on how researchers and analysts work with data and deal with uncertainty represents such a focus. The authors analyze data work activities, compare these to familiar data science tasks and processes, and propose a flow diagram that captures the dynamic nature of data work. Cabitza and Locoro [51] discuss two sides to working on data, primary use and analysis, and then processing and storage, tasks often handled by IT specialists in large organizations.
Second, Cabitza and Locoro [51] point to the notion of working by data, as when activities depend on (accurate) data and/or play an organizing role in work. For example, when data is handed over by another actor (human colleague or automated system) and to a significant degree shape how the task should be performed and when. We see the discussions by Lee et al. [192] and Irani and Silberman [146] as a broader extension of working by data insofar multiple systems can assign tasks and direct work through data. Here, data work is closely related to work that is dependent on or organized around ordering mechanisms [271], e.g. forms, protocols and manuals, but with an explicit data-centric focus.

Third, several contributions have pointed to the additional work associated with working on and becoming increasingly dependent on data in existing work activities. Cabitza and Locoro [51] discuss the additional effort and articulation work that health care workers put into making a record a working record (see also Fitzpatrick [97] for additional details). Fischer et al. [95] discuss data work based on studying energy advisors and their practices around installing sensing kits, analyzing data and giving advice. Their notion of data work is closely related to articulation work as “[...] the work to make the sensor data work in support of the advisors’ advice giving practice” [95, p.5936].

The perspectives and topics covered above draw upon and reflect strands of work within CSCW. Cabitza and Locoro [51] criticize (but define and use) the concept of data work as on the level of paperwork, in that it is both abstract and vague and suggests an instrumental working on data understanding. Similar observations can be made in relation to knowledge work and have strong links to previous works within CSCW on knowledge- and expertise sharing [1, 2], as well as studies of how people use spreadsheets and similar [223, e.g.]. The three levels of data work share concepts with, e.g discussions on the relationship between work and articulation work [271] and the levels of collaborative activities discussed by Bardram [22]. Still, the data part of data work introduces a few challenges with respect to articulation work and collaborative activities. The concept of coordination mechanisms as developed by Schmidt and Simonee [271] focus on how different external representation and artefacts can support articulation work in cooperative activities, e.g. forms, protocols, task overviews [271]. While coordination mechanism such as bug reports and work forms reduce the need for articulation work, they do not remove it. Additional work is still necessary to make forms and templated data views workable as Fitzpatrick have shown in a detailed analysis of care workers use of medical records [97]. This is an important observation when considering why data sets (in spreadsheets or “raw” formats such as JSON or XML) cannot act as coordination mechanisms and introduce an additional need for articulation work (without additional constructs). When moving from templated views and protocols, to displaying data as comma separated values in a spreadsheet or “raw” data formats, the coordination aspects are stripped away. When a shared dataset is used across different activities, it is
tempting to approach it analytically as having the qualities of a boundary object in that it “have different meanings in different social worlds but their structure is common enough to more than one world to make them recognizable.” However, Lee discuss how so-called boundary objects (Lee’s analysis) often fail to satisfy the informational needs of collaborative practices and still require considerable additional explanation and discussion to be intelligible. Thus, when knowledge workers engage collaborative processes and activities that mix working on and by data, data itself serve as a poor boundary object and coordination mechanism. It often represented outside the systems and templates typically mitigating the need for articulation work and negotiation. We will return to this discussion in section 10.5.

10.2.2 Cooperation and Collaboration in Data Analytics

Supporting cooperation and collaboration is an important topic in visual analytics and data analysis. A few works have begun to focus on supporting collaboration in data and visual analysis, exploring collaborative sense-making and sharing, primarily within the sciences. Recent studies are touching upon how analysts collaborate. Heer and Agrawala discuss opportunities for collaboration in the data visualization process, and high-level design considerations when designing tool support for asynchronous collaboration within visual analytics. They discuss division of work in the analytical process (based on the information visualization reference model and sense-making model), see below. Among other, they stress the important role of awareness of the activities others and the importance of supporting group consensus. Isenberg et al. discuss collaborative visualization based on the collaboration time/space matrix of Ellis et al. They apply a tool- and visualization-centric perspective and discuss application support for collaborative visual analytics, without touching upon the complex dynamics of activities. Although rich on design considerations, the conceptual work is not developed (directly) based on empirical insights and studies of collaborative data analysis in practice.

A few studies have touched upon collaboration in data science and analytics from an empirical vantage point. In their study of enterprise analysts, Kandel et al. employ a simple view on collaboration as knowledge sharing among the analysts and across the different archetypes (hacker, scripter and application users). They found that analysts rarely interact and collaborate around a specific task, but share resources: data, scripts, results, and documentation. A recent study by Passi and Jackson examines common tensions on enterprise data science work. While they study data scientists and how they collaborate in their work, they describe how the tensions shape work and collaboration, and how the tensions are mitigated (or not) with different strategies, “/.../ in service of imperfect but ultimately pragmatic and workable forms of analysis.”

1 an analytical perspective suggested by reviewers and readers of an early draft of this paper
Choi and Tausczik [62] examine the characteristics of collaboration in open data project and analysis. They examine team composition, group dynamics and expertise (communities of practice). They found that expertise (technical or domain) is crucial in team composition and when motivating how projects and people seek out collaborators.

This work represents, at the one hand propositions about collaborative process models and the need for systems support, and on the other hand, cases that exemplify the dynamics and diversity of collaborations around data analysis. The case presented below adds to this body of work by providing empirical grounds for continued explorations into computer support of cooperative data work and in discussing some of the collaborative aspects associated with data work.

10.3 Case Study: Danish Business Authorities

DBA is a public agency within the Ministry of Business and Growth with a wide range of responsibilities and tasks, including company and public audit oversight, maintaining the digital records of Danish business, policy development and support for municipalities. Here we focus on two main activities. First, DBA is responsible for the oversight and approval of financial statements, auditors and audit firms. This is to ensure quality and accuracy of company audits and annual financial statements from Danish companies. The audit firms are selected for a quality inspection based on a risk-based assessment and the frequency is regulated by law. Second, DBA is responsible for and maintains the Danish Business Register, a database containing information on Danish companies. As part of this responsibility, they provide an online platform where companies can register and change their records, and provide legally required information and financial statements.

10.3.1 Study

The study was conducted throughout the fall of 2017, as part of a larger research collaboration with several research, public and private stakeholders, including DBA, on visual analytics, data analysis and big data. The study was developed based on issues emerging as part of ongoing meetings, workshops and design work, e.g. collaborating across a large shared data repository, engaging in risk-modelling across departments and groups, and accountability and transparency of business records available to the public. Thus, besides the empirical data from the interviews, the insights of this paper have also been developed through our experiences from ongoing field work and development with the data, which include concrete examples of the risk indicators in use.

2Accessible through virk.dk (in Danish)
3Link to research project omitted for review.
Table 10.1: Interview participants, length, primary area, role, background at DBA, and archetype (cf Kandel et al., 2012). ID is used as reference in quotes. Dash (-) letter indicate gender (Female/Male)

<table>
<thead>
<tr>
<th>ID</th>
<th>Background</th>
<th>Primary Area</th>
<th>Role(s)</th>
<th>Archetype</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBA-F1</td>
<td>Law</td>
<td>Business records</td>
<td>Caseworker</td>
<td>Application user</td>
<td>00:53:32</td>
</tr>
<tr>
<td>DBA-M2</td>
<td>Undisclosed</td>
<td>Spatial planning</td>
<td>Manager, Data Analyst</td>
<td>Scripter</td>
<td>00:40:32</td>
</tr>
<tr>
<td>DBA-M3</td>
<td>Geoinformatics</td>
<td>Spatial planning</td>
<td>Data Analyst</td>
<td>Hacker</td>
<td>01:02:17</td>
</tr>
<tr>
<td>DBA-M4</td>
<td>Business economics</td>
<td>Business records</td>
<td>Caseworker, Data Analyst</td>
<td>Scripter</td>
<td>01:03:52</td>
</tr>
<tr>
<td>DBA-M4</td>
<td>Political science</td>
<td>Business records</td>
<td>Data Analyst</td>
<td>Scripter</td>
<td>00:24:10</td>
</tr>
<tr>
<td>DBA-F6</td>
<td>Business economics &amp; Law</td>
<td>Audit oversight</td>
<td>Caseworker</td>
<td>Application user</td>
<td>00:49:32</td>
</tr>
<tr>
<td>DBA-M7</td>
<td>Undisclosed</td>
<td>Business records</td>
<td>Caseworker</td>
<td>Application user</td>
<td>00:50:25</td>
</tr>
<tr>
<td>DBA-F8</td>
<td>Public Accountant (auditor)</td>
<td>Audit oversight</td>
<td>Caseworker</td>
<td>Application user</td>
<td>01:00:18</td>
</tr>
<tr>
<td>DBA-M9</td>
<td>Law</td>
<td>Business records</td>
<td>Manager, Caseworker</td>
<td>Application user</td>
<td>01:14:10</td>
</tr>
</tbody>
</table>

We recruited the participants through the contacts at DBA involved with the larger research project. Based on our requests and an initial workshop, they selected participants within the organization connected to some of the areas of interest and/or perceived as relevant to the discussed focus. The participants all have different roles in their daily work with varying degrees of work with data and data analysis. Several of the respondents also participate in activities related to developing different intermediate risk models and heuristics in the organization and in activities related to infrastructure and policy development.

Data collection and analysis

We collected data through interviews following a semi-structured format. This was chosen to maintain a level of flexibility in the interview to accommodate for variations in analyses-tasks and project involvement [98]. The interviews were conducted at the Copenhagen office in 3 rounds, spanning 3 days in total. In each session two of the authors interviewed the participants individually following a semi-structured interview guide. The interview focused on their work, everyday tasks, work process, data and tools, and the role of collaboration and expertise. Following the first round of interviews, data quality and change was emphasized in the latter rounds. One interview (DBA-M4) was interrupted by an urgent task and not fully completed.

The interviews were recorded and each respondent was assigned a pseudonym (see table 10.1) and the interview data was subsequently transcribed. The interview data was analyzed with four stages of meaning condensations in order to develop the categories and themes presented here [178]. In the first stage, two
of the authors did a coded analysis of the interviews individually. This involved identifying categories and themes, such as data quality, analysis process, collaboration, knowledge management, tools and systems, data examples, challenges etc. Following this, we compared the coded analyses and discussed the categories and keywords with the aim of combining and clustering keywords under common categories. Then the interview data was re-analyzed using the selected categories. This was done by selecting quotes from each interview and organizing these in a spreadsheet. In the presented analysis we use representative quotes that convey the aspects of data work found within the interview data.

**Study Limitations**

There are several considerations imposing limitations to the study. First, recruiting the participants was done through our contact at DBA. While we rely on their judgment in selecting participants fitting the study focus, it introduces potential gate keeping and limited access to verify insights emerging through the interviews. This is a common challenge in organizational research [315]. Second, as a single case study, the direct findings offer limited opportunities for generalization without juxtaposition with existing empirical work.

In selecting the present case, we considered not the generalisability of the findings, but how the case could stand as an empirical example, highlighting characteristics of data work not previously studied in depth (see section 10.2).

10.3.2 Activities at Danish Business Authorities

Before moving on to the analysis and findings in the upcoming sections, it is necessary to provide information on the specific activities that the case workers engage in at DBA. We have grouped the activities within two areas.

**Financial Audit oversight**

The first area in focus is casework and data analysis related to selecting and inspecting accounting firms which are handling oversight of Danish business and their annual financial reports. The oversight activities are carried out by DBA on an annual basis. The oversight activity is divided into three overall steps: Selecting auditing firms for inspection, on-site inspection, and taking action based on the documents from the inspection and additional data drawn from the DBA systems. The work is divided between the case workers sitting in the Copenhagen office and on-site inspectors. The case workers curate the selection process and send the necessary documentation to the inspectors, who then, upon completion of the inspection, send the collected documentation and their assessment back to the case-worker, who subsequently analyze the data and take the necessary action. The case may be closed without further remarks, is maybe subject to further inspections, or DBA may forward the case to the Danish Disciplinary Board on Auditors or other appropriate authorities (e.g. tax or police). In any
case, the case worker prepares the documentation (in PDFs) and send a formal response to the auditing firm. The auditing firm can then issue different responses accepting or protesting the decision. This may add additional time (typically 4 weeks) to the case. As part of the process of moving toward becoming more data-driven, the selection process now includes integrating data from multiple sources, e.g. historical data, client company data, geographical data, and working with a risk-model in automating parts of the process.

Maintaining Business Records

DBA is responsible for and maintains the Danish Business Register, a database containing information on Danish companies. A large part of the dataset is open and freely accessible to the public to increase transparency and support activities by multiple stakeholders. The business records are used internally across the departments, e.g. for fraud investigation, geographic and economic analysis and in servicing companies. Outside DBA, it is used by other authorities (e.g. municipalities, tax authorities, ministries), analysis institutes and data-brokers, audit companies, companies, media, research and others. As part of this, DBA continuously monitors and manages the ongoing registrations from companies, changes made to the records and their yearly financial statements. In 2017 approximately 500,000 registrations and changes were made to the Danish Business Register.

DBA frequently initiates investigations based on changes and discrepancies in the business records. A substantial part of this happens as part of internal tasks and automated analysis that monitors changes in the data and checks if certain requirements are fulfilled by the registered companies. Certain events recorded in the data (changes in management, missing financial statements etc.) will cause the system to automatically instantiate processes that can lead to sending automated request to specific companies, prompt records for manual investigation, and in some cases initiate processes toward the termination of companies not upholding their responsibilities. Frequently, internal processes at DBA and external stakeholders will report discrepancies in the data, which in turn leads to investigations and quality management issues. DBA also conducts regular large-scale analysis-driven inspections based on known issues, strategic focal areas (e.g. known fraud schemes or common registration errors) or sudden emerging problems.

10.4 Findings: Cooperation in Data Work

In this section, we present the findings of our thematic analysis. We present both aspects of working with data and larger themes that influence collaboration and cooperation in data work.
10.4.1 Data-driven Work and Automated Analysis

Multiple processes within DBA increasingly rely on data and elements of automated analysis. Since 2017, DBA has moved towards a more data-driven approach where case workers collect and use data from multiple sources throughout their existing work processes. Moreover, they increasingly collaborate with the IT-department in integrating and running automated analysis based on risk-indicators from a shared model (cf section 10.4.4). Previously, selecting instances for inspection (e.g. auditing firms or financial statements) was primarily based on fixed regulatory rules (e.g. all firms within a 3 year cycle), strategic and political foci (e.g. specific company constructs and sectors) or re-checks of past neglects. Now, the process involves a higher degree of both manual and automated analysis. Case workers will develop and define rough criteria for the selection process based on the risk indicators, statistical analysis of prior years inspections, and additional new data sources (e.g. geographical data, company registration and fiscal data). This is a week long task involving all the members of the group. They start by collecting data from internal systems as generated spreadsheets (CSV) and run tailored queries in collaboration with the IT-department (generating JSON/XML). Following this, the team go through the spreadsheets to identify additional criteria for the automated selection process. This include student workers that are tasked with cleaning and combining the many spreadsheets. As the risk patterns are moving targets, the case workers will continuously update the selection criteria with new insights and ideas. In cooperation with programmers, the indicators are programmed into an automated process that goes through all business records to identify and generate a list for inspection. In the case of audit oversight, the output of the process is a large spreadsheet containing risk scores of each auditing firm, whether they have been selected or not, and individual sheets with additional information about the auditing firms.

"The machines processing of the data, whatever that’s called, these different algorithms, they took around five days to generate the data. And then there has been a tremendous amount of manual work in modelling the data that came out of the process. [...] Aside from the automated analysis in the spreadsheet, we also enrich the analysis with manual data. For instance, we collect input from other areas [departments], e.g. if the people overseeing the business records have encountered issues with auditing firms. It is a terrible mess."(DBA-F8)

As indicated by a respondent, the outcome of the automated analysis involves substantial post-processing before the final list is ready. Case workers and student workers will go through the data manually to integrate relevant data from other areas, such as financial statements from client companies (PDFs). As a final step, the case workers will go through the list to detect potential errors and prioritize the selected instances, as the output from the automated process far exceeds the available resources for inspection. An important final analysis involves trying to establish a sense of why an auditing firm was selected. A case
worker explained how she would keep detailed notes on each step “[...] in case someone subsequently make an inquiry, then we need to be able to account for the reasons motivating the inspection.” (DBA-F8) (see also section [10.4.5). Finally, on-site inspectors will review the selected instances and report back the necessary documentation which then can be incorporated and used in the next selection process.

“We have registered all the results from previous year’s inspections in a few huge spreadsheets, so that is the most data-driven part of our work at the moment.” (DBA-F6)

The increased articulation work serves both as a way to initiate new inspections based on observations in the data and as a mechanism to coordinate assumptions and insights, for example, when describing selecting criteria to programmers. As the use of data analysis introduces an element of uncertainty to the work process, the added communication is also increasingly ad-hoc. To some extent, this echoes observations made by Edwards et al. [87]. They describe how it is often impossible to capture all metadata, thus it is necessary to acknowledge the ad-hoc nature of communicating metadata and view it as a process. Besides the ad-hoc, unstructured and incomplete nature of communicating metadata, our study also shows that it is difficult to ground communication in data. Communicating new selection ideas to the programmers can be challenging since the ideas do not necessarily stem from the data. The case workers often do not have an overview of all variables, and their possible values and distributions. Thus, the programmers are forced to interpret criteria not directly grounded in data. In our work with the business data, we experienced similar challenges when discussing ideas with the personnel at DBA. For example, the concept of bankruptcy is encoded in an enum type variable (company status) that can take one of several values. Each time a company changes status, a time-stamped event with the latest status is added to the data, which also include towards bankruptcy (i.e. when the company is notified by the court) and different types of closure. A company can then alternate between normal and towards closure several times before actually closing, which means that multiple different status event series exist. Therefore, defining groups of interest to an analysis is not trivial. It is not as simple as stating normal vs. bankrupt, but it is also infeasible to state all the unique status series that should be included and how. Thus, it can be challenging to ground communication in data as real world phenomena are often encoded in some form of data abstraction.

The above example illustrates multiple aspects of cooperative data work. During the process, case workers work on data; they collect, clean, select, and reason about data across multiple systems and spreadsheets. The process share the dynamics with the model discuss by Boukhelifa et al. [13], in being iterative and non-linear. Following the automated analysis, the case workers continue working on data, but also start working by data. The selection process (and inspection step) is shaped by the results from the risk model and analysis. All these activities
10.4. FINDINGS: COOPERATION IN DATA WORK

rely on extensive articulation work and coordination – between the case workers within the group, student workers, colleagues from the IT-department and case workers specialized on other parts of the data.

10.4.2 Acquiring and Integrating New Data

One of the reasons for the increase in articulation work is the need to acquire, integrate and understand new data (in unfamiliar formats). This happens for example, when case-workers explore indicators outside already established analysis methods. Until 2016, the team overseeing the inspections of the auditing firms primarily used data on the auditing firms, i.e. historical data on inspections, rulings, errors etc. With the new risk-based approach, the team started looking at geographical and general company registration data as well as fiscal reports from the central business register. This data is collected from other systems than their normal case systems, where the case workers usually access data by downloading spreadsheets (CSV). When accessing these data, they had to deal with different formats, such as JSON\footnote{https://www.json.org/} and XBRL\footnote{https://en.wikipedia.org/wiki/XBRL}:

“For instance, all the accounting data was initially in a format [JSON] that was unintelligible [to us]. This was then converted into another format that is far more usable to for us to access on a system level [CSV]. […] Then we can make the tables we need. And yes, that is our primary way of working [with data].” (DBA-F6)

Several challenges with utilizing new data is therefore also related to technical expertise. Knowledge workers may not have the skills to access, modify and view the data from multiple different angles. Thus, employees at DBA are frequently seeking technical expertise from coworkers. In addition, the knowledge workers are often specialists in a subset of the shared data, which means that when integrating new data it often does not suffice to seek technical expertise. We will return to the topic of cooperation across expertises as a main way to deal with several challenges in section \ref{10.4.7}.

Another challenge with acquiring new data is the team structure of the organization. Traditionally, the organization has been divided into teams based on legal focus areas, and it has not been customary to share data across teams outside of the central business register. Teams have therefore developed various work practices where shared information within teams have been maintained in e.g., different spreadsheets. Finding relevant information about use of certain data variables or other types of internal information can therefore be very challenging. Furthermore, it is difficult to stay consistent and crosscheck across teams, which currently occur only on occasion in an ad-hoc and manual manners. At the same time, sharing data is not a trivial task and the idea of an "ideal" database containing all available data within the organisation may be utopia. Similar thoughts
have been presented by Bietz and Lee [34], when studying data sharing in the context of metagenomics. In our study, it is not clear that a single shared database is sufficient to accommodate the ad-hoc nature of data work and, at the same time, to be reliable and readable in all contexts of use.

10.4.3 Changing Data

Our study showcase that cooperation in data work is not only a product of opportunity (as described in the previous sections), but that the shared data also create dependencies between people. The data at DBA is constantly changing; either the quality and accuracy is improved or new data arrive, in particular when such data only affect your own work but have consequences for data work conducted elsewhere in the organisation. Data quality is a general issue in data work and much effort has been put into streamlining the wrangling process that often constitute the major part of any analysis process [158]. At DBA it is not only important to improve data quality in order to support data analysis, but also to maintain an accurate account of the business landscape. Thus, data quality is a shared goal throughout the organisation.

Issues related to data quality emerge in different ways. First, when case-workers find quality issues while investigating a specific case, they discuss these within the group to identify common issues. Second, changes to the system, migration of data and differences in how the data is registered can introduce quality issues. Third, several external data brokers use the publicly available data provided by DBA. The external data brokers catch errors as they analyse the data and report these back to DBA. A respondent recalled an example of the first category.

“We might discuss it at the weekly meeting, and I say: “I’ve been searching on ten different management statements [data element], and there is something wrong with all of them!” [Colleague] “I can see we have many cases involving the statements.” (DBA-F1)

The dynamics around the shared data repositories are further complicated by the fact that companies themselves are responsible for recording correct information as well as by the fact that the business register is publicly available, and thus used for analytic purposes by companies, public institutions, researchers, etc. Therefore, whenever corrections are introduced these may have direct and visible effects on the companies involved, as they will automatically be notified. But, it can also have indirect effects on case workers elsewhere in the organization, if they are not aware that the data extracts they are analyzing have been modified. Thus, when more and more data from different parts of the organization are used to improve traditional work tasks, new dependencies are also introduced that may not be directly visible. In addition, participants reported that it is not always completely trivial to update data. Sometimes, it is risky to do large corrections, as that can trigger the system to send out too many notifications, causing concern.
10.4. FINDINGS: COOPERATION IN DATA WORK

and noise within the organization. Similarly, correcting information (e.g. company status, time of registration, ownership), can have legal implications and, in the worst case, cause the initial registration to be invalid, meaning that the company was never formed. One of the current methods to deal with changing data is to create snapshots of the data repositories or parts of them. A participant explained that they would sometimes create these snapshots in order to always be able to account for the foundation of their analyses. In general case work, the participants will also sometimes redo work when returning or taking over a case. This is in order to be able to understand and explain the reasoning applied, but also to ensure that the current information is up to date.

10.4.4 A Shared Risk Model

In 2016, DBA started developing risk models containing both known and new indicators based on historical data from past inspections and rulings, as well as from shared experiences discussed in internal meetings:

“The background [for the model] was results from past inspections and identified errors, with a few additions that originated in a brainstorm in the team, where we discussed what could be considered as risk factors [indicators]. We also made them public on a general level and invited the [external] inspectors and auditors to contribute if they saw something else than we did.” (DBA-F6)

Initially, the risk indicators have served as a way to consolidate knowledge about non-compliant behavior within each team. However, recently the indicators have also been used to consolidate knowledge across teams, and thus as a way to share knowledge about the use of different data. While the focuses of the teams vary, the participants reported that several of the parameters were quite similar across the teams; hence the same risk patterns can often be useful in multiple different focus areas. Sharing data is therefore not only about data and meta-data, but also about concrete use and the intuitions behind. Currently, the risk indicators are simply shared through text documents that describe each indicator including motivations and how to calculate it. As part of our study, we got the document of the team that inspect financial statements, and this reveals several interesting observations about sharing of data work in practice:

Firstly, the risk indicators does not conform to a single format, which means that it is not trivial to combine the different types. Some of the parameters are very concrete checks, such as missing or incomplete data; others are quantifiable parameters that require some form of interpretation, like high or low margins; and some can be described as soft parameters, such as frequent changes in board members and accountants, which are very context dependent. Thus, risk indicators are not only simple calculated values that can be summed and averaged; while some can be used as features for machine learning, others also require contextual judgment. Secondly, several of the indicators are described without using actual variable names, meta-data or how to combine several data variables in different
formats. Thus, the descriptions of the risk indicators are sometimes described on a different abstraction level and thereby detached from the actual data. This can be beneficial when discussing data utilization between people that are not all grounded in the actual data, but it can be challenging to realize the concepts afterwards. For example, a programmer can have a difficult time translating the concept of frequent board member changes into model parameters, thus back and forth communication between different expertise groups is inevitable. Thirdly, the initial target group of the indicators are people of similar expertise and grounding in the same work practice; the vocabulary is often specific to law or accounting, whereas the same vocabulary is not always present in the data.

The risk indicators were initially used as guidance to human workers, however, the systematic collection of the risk parameters have been done in order to use them in algorithmic models (cf. section 10.4.1). The long term goal is to use the existing indicators in machine learning and to use machine learning to identify new indicators. This is an ongoing process, hence so far the indicators have mainly been used as checks in automated selection processes in collaboration with the IT department:

“And this is new in 2017. Here we have a system where we input a list of risk factors and then it scores them [auditing firms] based on whether they occur or not.” (DBA-F6)

Thus, current validation of the indicators is based on statistics from previous years as well as domain knowledge of law and regulations. At the time of the study, the team had just initiated a process where they would add additional historical data to validate the model and potentially discover new patterns.

“They [risk indicators] need to be assessed every year, because we get additional inspection data. It is envisioned as a form of loop, where we can analyze the results from the inspections up against the risk factors, and say, we need some other factors or there are other tendencies at play.” (DBA-F8)

Furthermore, participants reported that new types of risks always will emerge, and as such this type of work involves adjusting over time:

“If someone finds a new type of error in a financial statement, I immediately think about whether I can use it in our risk model and whether I can create a operational rule for it.” (DBA-M4)

While the long term goal is to incorporate more automation in the analysis of data at DBA, there are several aspects of the risk model that highlight that also future roles of more automated models will be of advisory nature. The mixed types of rules that make up the risk indicators require situated judgment and an understanding of the context to ensure correct interpretations. In addition, the work involved in aligning the real world to patterns found in the data will continue in the future as new risk patterns emerge, which will require human judgment and design.
10.4.5 Accountability and Transparency

Concerns regarding accountability and transparency were touched upon by multiple participants, and the thematic analysis therefore resulted in several concepts related to this. Prominently, participants were very aware that decisions they made would have to be accounted for and explained to everybody involved, or that results they are passing on can be used for important decisions;

“The numbers you pass on in the system can be used politically, so I am trying carefully to explain the limitations of my work.” (DBA-M2)

Concerns like these motivated a lot of the collaboration within teams, where multiple opinions were used to assure the quality of assumptions, conclusions and decisions. Participants furthermore reported that their domain expertise was important in this process, especially when describing the basis of their decisions to not only peers, but also stakeholders affected by their decisions. Co-collated synchronous activities around the same screen were not uncommon with high-complexity cases or in master-apprentice scenarios to share knowledge and discuss appropriate next steps. An understanding of the definitions, assumptions and analyses were therefore mainly achieved by participating in such processes. This created challenges since everybody cannot be part of the process all the time;

“Typically, they would probably start over, because you need the overview of a case, which is difficult to pass on.” (DBA-F1)

This overhead in data work served as a way to achieve an appropriate level of understanding, and as such it might be impossible to completely avoid this kind of repetitiveness. At the same time, sharing result, information and knowledge is important in cooperative data work, since our study shows that managing accountability and quality involves multiple people. The tool support for shareability was focused on synchronisation of initial data and end results, and not on workflow, intermediate results and contextual information.

10.4.6 Awareness of Activities and Knowledge

A main theme in the analysis was group awareness. Awareness of what coworkers are doing, have done and what knowledge they possess in relation to data collection and usage. Participants frequently responded that people was specialised in different parts of the data or had certain domain, analytic or technical knowledge:

“We are specialized within the team. I have a lot of knowledge about what data we have in-house, plus I have been part of the business registration area as well, so I generally know about the available data in our system, which my team members do not necessarily. But then they know different things.” (DBA-F6)
The importance of awareness in collaborative work has previously been highlighted in related work. “Having an understanding of the activities of others provides a context for your own activity” [81], which ensures that the work of individuals are relevant to the group as a whole. Thus, awareness is important to coordinate activities and it is often a prerequisite for the actual work tasks [55, 81, 128]. In addition, Carroll et al. [55] present four facets of activity awareness: common ground, communities of practices, social capital, and human development, and conjectures that collaborators need to be aware of all facets to effectively work together.

This classical view on awareness was also important to the participants in our study, when discussing data work conducted within their respective teams. Awareness about what coworkers are currently working on or tasked with, and awareness of shared resources that establish common ground, often in the form of spreadsheets. For the most part, activity awareness was maintained through synchronous, co-located work activities with team members. The participants also expressed general awareness of common practices within their teams, and to ensure that data work follows those practices, the participants was regularly discussing cases with team members sitting in their proximity.

“It’s quality insurance, and for the new people training as well. Some times there is a sense of security in seeing the same and confirming it. It’s also because there is a high level of complexity in the data and the cases, especially for the new people. Then there is a need for more eyes.” (DBA-F1)

Activity awareness was rarely expressed as a theme across teams, unless there were large coordinated efforts between multiple teams. Besides these large scale coordinated cases, activity awareness across teams was described as ad-hoc personal interactions, and as a consequence participants rarely took advantage of overlap in data usage or task focus when looking for help or finding important information. However, in the process of utilizing new data driven methods, integrating new data, and relying on changing data, the scope of the awareness that is needed has increased beyond the collaborating teams. The participants frequently expressed that employees with a longer history in the organization possess a higher degree of knowledge about coworkers outside of their teams. Knowledge which is often important in data work. For example, when identifying new useful data, information about data collection or how certain variables can be interpreted. Employees will also exercise this broader sense of awareness when identifying who possess certain technical or analytic expertise within the organization. Thus, our study suggests that the scope of group awareness is expanding when data is shared beyond individual collaborating teams. Furthermore, new types of awareness relating to the knowledge others posses about data, meta-data and past data work, becomes important. This observation is inline with the conjecture that it is important to view meta-data sharing as process [87], and may suggest that awareness about the knowledge of others, beside your direct collaborators, can potentially support the establishment of common ground on an organizational level.
10.4.7 Cooperation Across Expertise

A large theme in the analysis was asymmetric cooperation, which we characterize as the cooperation between people with little overlap in expertise. The participants reported large differences in both domain knowledge, data analysis proficiency and technical expertise. Furthermore, our study suggests that a single employee rarely possesses all skills needed, and since data work often requires multiple expertises, asymmetric cooperation becomes unavoidable. Participants described that cross-expertise cooperation fostered an environment where coworkers challenged each other and thereby improved outcomes. However, at the same time, asymmetric cooperation also introduced certain challenges when extensive knowledge and communication gaps existed. An example of this was in defining the requirements to data extracts between case workers and programmers. If the requirements were described too imprecisely, they could be difficult to translate into code, and as a result back-and-forth interactions were needed in order to reach an understanding. A main way of dealing with these challenges was through so-called ambassadors, who possessed cross-expertise knowledge;

“He is a good example of an ambassador that is capable of having a foot in two camps, and that is really great.” (DBA-M9)

Ambassadors do not necessarily possess maximal expertise in all areas, however, they have sufficient knowledge to mitigate potential misunderstandings and thereby they get an important role in collaborative sense-making. There was no immediate technical support for asymmetric cooperation, but the business registration repository has two interfaces; one is attended for the general public (virk.dk) and one serving employees. Interestingly, case workers would still use the public interface since it was considered to be easier. Similarly, when finding data for externals, the public interface was also occasionally used to ensure only obtainable information was shared. We found only few examples where data visualizations were used for analysis purposes, and one participant explained that they mainly used data visualizations when reporting results to management.

10.4.8 Interdependence, Ambiguity and Non-linearity

In summary, the challenges in cooperative data work as found in our study include additional dependencies between knowledge workers (cf. sections 10.4.2 and 10.4.3), ambiguous communication (cf. sections 10.4.4 and 10.4.7) and non-linear workflows (cf. sections 10.4.1 and 10.4.6). Although knowledge workers are organised in teams and focus areas, the complex dynamics in maintaining and using shared data introduce multiple dependencies and overlapping work tasks. For example, changing data can influence multiple knowledge workers throughout the organization, patterns in data can be relevant outside ones own focus area and limited knowledge about meta-data and utilization force workers to often seek assistance to ensure the quality of their work.
In this dynamic work landscape, communication and articulation work become keys to success; both to improve work tasks with data and to share findings. However, there seems to be a tension between abstract discussion of data utilization and actual concretization of often ambiguous concepts. Discussing the use of data on higher abstraction levels can be useful when workers are not grounded in the data in the same way. Thus, it is a way to bridge expertise levels, or simply a way to avoid communicating too many details about analytic work. At the same time, concrete realizations of analytic ideas are what drives data work. The transition between working on different abstraction levels in data work is challenging, since it can be hard to translate ambiguous ideas to concrete instances and maintain links between high-level discussions and low-level realities. This can seem paradoxical when data ideally should provide solid ground to work tasks and decisions, but nevertheless it is a core problem when data is an abstraction of real world phenomena.

Our study has shown that group awareness and workers with hybrid expertise are some of the concrete strategies for dealing with these challenges. However, this means that cooperative data work inevitably becomes non-linear and iterative as knowledge workers needs to accommodate changing data, changing assumptions and new insights. While it is acknowledged in related work that working with data is often non-linear, our study highlights some of the underlying challenges in the context of cooperative data work.

10.5 Implications for Computer Support

How is cooperation in data work different from other types of knowledge work? And, what can we learn for the design of computer support for cooperative data work? While it is clear that we cannot handle data without computers, it is not clear what role systems should have in cooperative data work. In the following, we discuss these questions in light of related literature.

10.5.1 Data work, Articulation Work and Boundary Objects

As many domains and organizations move towards becoming more data-driven and we see an increased interest in automation, understanding how data shapes and impacts work is crucial. Here, the concept of data work puts an analytical emphasis on how non-data scientists, i.e. a majority of knowledge workers, might face and deal with the increased datafication of work. We agree with Cabitza and Locoro [51], in that the distinction between working on data and working by data is practically inseparable – knowledge work within data-driven areas, like the case presented here, exhibit elements of both. However, keeping the two levels separated offer an analytical perspective on how collaborative data work unfolds. For example, comparing how data science-like tasks as a non-data scientist versus how work is shaped and directed by increasingly complex processes, that combines collaborative and cooperative data work with elements of automation.
10.5. IMPLICATIONS FOR COMPUTER SUPPORT

The third level of data work, i.e., articulation work, is a familiar concept within CSCW. As discussed (cf. 10.2) and presented in our study, when data is separated from systems and templated domain views, e.g., in spreadsheets and raw formats, it requires significantly more articulation work than discussed by Schmidt and Simonee [271] and Fitzpatrick [97] on coordination mechanisms and working with health-care records. This suggest that in data-driven practices and data work, there is an increased need for systems and coordination mechanisms that can support articulation work.

To contextualize cooperation in data work, it seems worthwhile to ask the question: Can data in itself be a boundary object? Boundary objects has been central in CSCW literature since the concept was proposed by Star and Griesemer [284]; and it has been highlighted several times that the construction of boundary objects is important in collaboration between communities of practice [44, 184, 190, 284, 310]. Based on the definition of and the popular use of boundary objects as an analytical objective, it seems a likely conclusion that data itself or shared databases constitute boundary objects, as they effectively bind together intersecting communities of practice and expertise groups in data work. However, our study suggests that data rarely meet the informational requirements of boundary objects, as significant efforts are used on sharing knowledge about the data. For example, in the form of the shared risk models (cf. section 10.4.4) or the articulation work used to align data work practices (cf. sections 10.4.2 and 10.4.6). One of the general critiques of the concept is that some of the things we call boundary objects does not satisfy the informational requirements for each community of practice. Thus, Lee [190] have proposed the notion of boundary negotiating artifacts to describe the kinds of artifacts used to transmit information in non-routine collaborative processes. While boundary objects may be the ideal, the lack of routine (e.g. in certain types of data work) introduces the need for boundary negotiating artifacts.

The text document of the risk model can be thought of as a boundary negotiating artifact, as it is used to consolidate information about potential risks on a high-level. In this way, cooperation and collaboration in data work is no different from other types of knowledge work. Boundary negotiating artifacts are also useful in data work to create shared understandings and provide contextual knowledge, thus fostering collaboration. However, the risk model also shows that there is substantial translation work involved in cooperative data work. Shared understandings are often established based on high-level ideas and real world concepts, hence leaving cooperators to translate these concepts to concrete data and utilization in their own practice. In our study there were few tools or boundary negotiating artifacts in use that could support such translations. The main ways to deal with these challenges were through personnel with hybrid expertise and awareness of what your colleagues know. Thus, the need to talk about data utilization on multiple abstraction levels seems as a thing that sets cooperative data work apart from traditional knowledge work.
10.5.2 Data as a Dependency

As data become a more integral part of traditional knowledge work, our study has shown that additional dependencies are established between workers. As a consequence, cooperation in data work is not only about working together to reach a shared goal, but also to align otherwise separated work tasks, i.e. to align how data is interpreted and used across organizational boundaries. In turn, this type of alignment also involves negotiating how and what data is being collected. As the participants pointed out, the format of the shared data repositories is continuously improved while aiming to maintain the usefulness of legacy data entries. In some sense, the data repositories can also be described as a boundary negotiating artifact as echoed by Bietz and Lee [34]. But, the continuous improvements can also have negative impacts since new formats require new knowledge of context, introduce translation work (e.g. mapping between old and new formats) and new tools. A few of the participants reported that previously they were able to access the data directly in the database, but updates to the database technology forced them to use another query language, leaving them unable to have direct access.

The additional dependencies introduced by relying on shared data repositories also mean that sharing ways of utilizing data and the intuitions behind become increasingly important. The shared risk model is an example of the need to also share motivations behind data work. Several papers have addressed the challenges of sharing and reusing data. For example, Bietz and Lee [34] description of how sharing data requires negotiating common data repositories while reuse often requires data to be reworked and thus how data often does not sit in ready as-is repositories [240]. In addition, sharing of data requires intrinsic knowledge about metadata and context, which is often best shared through articulation processes [87]. Our study indicates that in cooperative data work it is also important to recognize whenever your own data work is affected by data work conducted elsewhere in the organization. Changing data can alter the decisions of knowledge workers, prompt new data work and require algorithms to be recomputed. Furthermore, our study has shown that translation work is critical in cooperative data work; between high-level concepts and concrete combinations of data variables and between new and old data formats. Thus, sharing and reuse of data work also requires workers to share how those translations are made. The observed translation work can be thought of as a consequence of the sentiment that "raw" data is an oxymoron as described by Gitelman [104]. The careful curation and maintenance of data that increasingly transform organizations also means that data is largely a designed abstraction of the "raw" real world, thus any sharing and reuse requires re-articulation and re-contextualization.

Another major factor in sharing and reuse of data work is technical and data science expertise. Our study has also shown that technical expertise varies a lot (cf. sections 10.4.7 and 10.4.6), which means that computational literacy and general understanding of, e.g., machine learning concepts are low with certain job roles, even though they are still required to use data analysis. While this comes
as no surprise, it is still important to acknowledge when trying to support sharing and reuse of data in general knowledge work. In addition, our study suggest that knowledge workers have a lot of important input and ideas on how to use data although they have limited knowledge of data science methods. Thus, manifesting and recording translations between high-level discussion and concrete data work can potentially ease and foster future dialogues.

10.5.3 Mixed Expertise Cooperation

Our study echoes the sentiment that data analytics requires multiple different types of expertise, which is further enhanced as data analysis is adopted in general knowledge work. However, our findings also indicate that it is unlikely that individuals possess high degrees of both technical, analytic and domain expertise, which ultimately creates expertise asymmetries in cooperative data work. In social science, Clark et al. [67] describes the notion of common ground as a prerequisite for successful communication, i.e. the generation of mutual knowledge, mutual beliefs and mutual assumptions. Thus, in cooperative scenarios with low expertise symmetry it can be challenging to establish grounding in the data and in methods of utilization. Currently, so-called ambassadors with a foot in multiple camps are a way to mitigating expertise asymmetry in cooperate data work. While they may lack full knowledge or capabilities within all areas of data work, they are highly valuable in translating between domain knowledge and data specifics and they assist workers in interpreting and using analysis methods.

Current tool support for data analysis often focuses on enhancing the capabilities of its users, so as to mitigate missing expertise in certain technical or analytical aspects. A prominent commercial example is Tableau, which enables its users to easily create various visualizations and this way utilize data. Tableau also supports the creation of story points with major findings from an analysis, and thus supports communication of results. However, in tools like Tableau, there is little focus on continuous sharing and discussion of partial progress as the communication part is mainly a final outcome. However, our study indicates that dialogs around data occurs continuously, thus maintaining motivations, assumptions and interpretations of data work as an integral part of the software could be an exciting avenue of future research to enable easier alignment of ideas.

As expertise asymmetry is an inevitable part of cooperative data work, another potential implication for computer support is to allow individuals to conduct work suited their expertise and then transfer intermediate results to coworkers with different expertise. This would potentially include software that allows workers to interact with the same data and analytic work with different interfaces. Vistrates [19] is an example of a technology that aims to support cooperation between multiple expertise levels. Its component model allows users to share atomic analysis parts that can be recomposed in reactive dataflow pipelines. But more importantly, since the system is built on the concepts of malleability, shareability and distributability [169], it allows for building software where the same underlying
data work can be modified from different interfaces; a potential way to consolidate data work from multiple expertise groups.

10.5.4 System Mediated Awareness

As indicated in section 10.4.6, group awareness in data work is not only important to coordinate activities but also to align knowledge of meta-data and practices around data utilization. As shared data introduce dependencies (cf. section 10.5.2), it is important to be aware of how the data work of others will influence one’s own work. For example, whether certain data quality improvements invalidate any existing analyses? Additionally, our study has shown that being aware of data knowledge and data utilization of others can foster new opportunities and strengthen data work. For example, if one team investigates a specific company and another team is investigating accountants of that company, both parties will likely benefit from collaboration and the exchange of knowledge. While similar observations have also been made for collaborative visual analytics [128], our study shows that awareness is not only important for social navigation and coordination, but also to improve and maintain the integrity of the data work itself.

Awareness in data work is closely coupled with the modification and utilization of the actual data, which introduces interesting opportunities for system mediated awareness. Simply being aware that two otherwise disjoint parties are using or modifying the same data can potentially foster collaboration or support sharing and reuse of data work. Automatic detection of links between concurrent data work or new and old data work can also inform who to ask what. In addition, this can be a way to improve social navigation and coordination, specifically in cooperative data work. We therefore believe that different ways of creating system mediated awareness is an interesting avenue of future research, as our study indicates that data centric types of awareness are important for cooperative data work. This type of future research also involves avoiding the creation of new work practices where knowledge workers suddenly feel monitored or overloaded with information.

10.5.5 System Environments for Cooperative Data Analysis

As indicated by our study, cooperative data work oscillates between routine and chaos. Inspection work at DBA follows an increasingly systematic approach, while new data, insights and quality changes can disrupt current cooperative work. One minute a data worker carry out routine analytic work and the other minute new insights change who they work with or and why. Thus, optimal formats for data analysis in cooperative data work remains an open question.

There are numerous techniques in data science and visual analytics that relate to the challenges presented in this paper. Data wrangling is becoming a core focus area in research and commercial software [157, 158, 291, 298], with an
integrated goal of keeping an accurate record of the data preparation, e.g. in
the form a wrangling recipe [298]. Data pipelines are used to provide a visible
account of how data is processed [19, 291] and notebooks are increasingly used
in data analysis prototyping as well as for explaining data work [170, 250, 289].
For more predictable types of data work tasks, the visual analytics community
has a long tradition for tailored applications that effectively combine information
visualization and computation [88, 89, 162, 205, 209]. In addition, communication
of findings have been in focus as a separate stage following the actual analysis,
e.g. in the form of storytelling [189, 268, 274]. While each technology have key
strengths, it remains challenging to define how they can be effectively combined
to meet the requirements of an organization.

Currently, spreadsheets are heavily in use at DBA. They provide familiar
structures for working with data, while they are flexible enough to accommodate
chaos. They can serve as boundary objects for routine types of tasks and simulta-
neously as boundary negotiating artifacts for ill-defined data work. To better
adopt data-driven methods, it seems that future formats for cooperative data
analysis should have similar qualities, for instance by combining and integrating
several of the technologies presented in this section. Transitions between data
analysis and communication of information is important as manifested by the
oscillation between routine and chaos. In addition, recording your own reason-
ing for later retrieval, accommodating multiple expertise groups, and being aware
of how and when the "solid" ground changes (i.e. shared data repositories) are
increasingly important aspects.

To exemplify how these insights can be manifested in concrete designs of new
environments for cooperate data work, we refer to our system InsideInsights [210]
that builds on top of Vistrates [19]. InsideInsights combines a data flow-based
pipeline model for data analysis with provenance tracking, literate computing and
storytelling elements. The goal is to create a software model where the result-
ning data analysis documents could serve as both boundary objects and boundary
negotiating artifacts while combining state-of-the-art data science methods. Part
of this was to allow users to take on different roles depending on the needs or
their expertise. Users can program new data analysis components from scratch or
compose existing component in the pipeline. The outputs and results of the com-
ponents (e.g. visualizations or calculations) can be combined to application-like
interfaces or narrated with structured literate elements that also allows hiding of
details. This way, users can at any point simultaneously modify the code, modify
the overall structure (i.e. the pipeline), interact with visualizations and analysis
results and review explanations and reasoning as manifested by the narrative.
This multi-role software environment is one way to directly foster a work environ-
ment were the same data work can be used and interpreted by mixed expertise
personnel and different communities of practice. While it is out of the scope of
this paper to fully describe the system, we believe it illustrates how the insights
of this paper can be used to design new types of computer support.
10.6 Conclusion

In this paper, we have presented the results from an interview study with data workers at the Danish Business Authority. We have exemplified practices and challenges in cooperative data work to increase the scope and understanding of data work beyond the existing focus on data science. We argue that this focus has become increasingly important since data analysis is becoming a major part of many knowledge workers’ existing work practices and as such present important real world challenges. Our results both confirm conjectures from related work and present new perspectives on data work. We have described the challenge of translating between high-level concepts and concrete data in asymmetric cooperation, how data work is non-linear and oscillates between routine and chaos, and how shared repositories introduce dependencies in data work. We have consolidated our findings in a discussion of implications for computer support based on related work on cooperation as well as state-of-the-art technologies for data analysis.
Vistrates: A Component Model for Ubiquitous Analytics

Sriram Karthik Badam\textsuperscript{1}, Andreas Mathisen\textsuperscript{2}, Roman Rädle\textsuperscript{3}, Clemens N. Klokmose\textsuperscript{3}, & Niklas Elmqvist\textsuperscript{1}

\textsuperscript{1}College of Information Studies, University of Maryland, College Park, USA
\textsuperscript{2}Department of Computer Science, Aarhus University, Denmark
\textsuperscript{3}Department of Digital Design and Information Studies, Aarhus University, Denmark

Figure 11.1: Vistrates transcend the traditional tool boundaries of analysis activities, analyst expertise, input and output devices, and modes of collaboration to enable a truly ubiquitous visualization and analytics workflow. Each vistrate (center) is a shareable dynamic media \cite{169} where components encapsulating existing web technologies—such as D3 \cite{42}, Vega-Lite \cite{270}, Leaflet, and Plot.ly—can be made to interoperate seamlessly. The document can be accessed from multiple settings, using heterogeneous devices, and by multiple concurrent users in activities ranging from data wrangling and exploration, to development and presentation.

Abstract

Visualization tools are often specialized for specific tasks, which turns the user’s analytical workflow into a fragmented process performed across many tools. In this paper, we present a component model design for data visualization to promote modular designs of visualization tools that enhance their analytical scope. Rather than fragmenting tasks across tools, the component model supports unification, where components—the building blocks of this model—can be assembled...
to support a wide range of tasks. Furthermore, the model also provides additional key properties, such as support for collaboration, sharing across multiple devices, and adaptive usage depending on expertise, from creating visualizations using dropdown menus, through instantiating components, to actually modifying components or creating entirely new ones from scratch using JavaScript or Python source code. To realize our model, we introduce VISTRATES, a literate computing platform for developing, assembling, and sharing visualization components. From a visualization perspective, Vistrates features cross-cutting components for visual representations, interaction, collaboration, and device responsiveness maintained in a component repository. From a development perspective, Vistrates offers a collaborative programming environment where novices and experts alike can compose component pipelines for specific analytical activities. Finally, we present several Vistrates use cases that span the full range of the classic “anytime” and “anywhere” motto for ubiquitous analysis: from mobile and on-the-go usage, through office settings, to collaborative smart environments covering a variety of tasks and devices.

11.1 Introduction

Visualization, for all its success within academia, industry, and practice, is still very much a fragmented area with no common, unified method that applies in all (or even most) situations [18, 24]. For any given visualization project, the choice of tool, technique, and approach depends heavily on the dataset, the goals of the analysis, and the expertise of the analyst and audience. Many additional factors come into play: At what stage is the visualization going to be used: during initial analysis or presentation of results? Is the analyst alone, or is there a team consisting of multiple people, each with their own roles and expertise? Are there special devices or equipment, such as smartphones, tablets, display walls, or tabletops, that should be integrated?

All of these questions give rise to specific choices among the available tools and techniques in the visualization field today. For example, in terms of expertise, a novice may go for a template-based visualization tool such as Excel, a financial analyst may choose a shelf configuration tool such as Tableau [288], and a data scientist may opt for Jupyter Notebooks [239]. Early on in the sense-making process, a developer may choose Observable [231] to interactively code their analyses and see immediate results, whereas a more mature project may call for a custom-designed web visualization built in D3 [42] or Vega [269], and a final report for communication may require a narrative visualization tool such as Graph Comics [14], Data Clips [7], or even Microsoft PowerPoint. Graph data may influence a designer to pick NodeXL [122] or Gephi [23], whereas tabular data may require Spotfire [3], and event sequences may mean using EventFlow [219] or EventPad [52]. Obviously, there are currently few synergies between these five determining criteria that we identify—expertise, analysis stage, data, single/mult-
user, and device—and committing to one typically means disregarding the others. Furthermore, this fragmentation takes its toll on as participants need to make a “collective compromise,” negotiate a common software denominator \[230\], and expend additional effort to share information, import and export artifacts, and work across visualization systems. The core issue is essentially one of interoperability: how to combine functionality from multiple available platforms?

In this paper, we apply the vision of shareable dynamic media \[169\] as well as recent advances in conceptualizing and implementing software as information substrates \[30\] to the field of data analysis. Information substrates blur the traditional distinction between application and document, they embody content, computation and interaction, and can evolve and be repurposed over time. We propose a component model for assembling visualization and analytics pipelines into such information substrates to increase their analytical scope. In this model, a component is a unit module with internal state, inputs, and outputs. Components provide visual analytics functionality and are reusable, replaceable, and extendable. This allows them to become building blocks for data analysis systems. Following the philosophy of information substrates, these systems can be integrated in media such as slideshows, interactive whiteboard canvases, reports, or interactive applications. Designing a comprehensive component framework is outside the scope of this paper, but we propose a basic library of components based on a few examples. While the component design space is huge, our model provides a starting point for future advances.

To demonstrate this model, we introduce VISTRATES (visualization substrates), a web-based collaborative platform for visualizations manifested as dynamic media (Figure 11.1). Vistrates provides a document-based framework of cross-cutting components for data visualization and analysis that transcends (i) both single-user and collaborative work, (ii) the full spectrum of data analysis activities, (iii) all levels of expertise, and (iv) a menagerie of devices. The platform provides a data layer for managing data, a pipeline model for controlling data flow, and a canvas for organizing visualization views. It is well suited to the type of “anywhere” and “anytime” sensemaking characterized by Elmqvist and Irani as ubiquitous analytics \[17, 90, 309\].

Components in Vistrates are maintained in a component repository as prototypes, from where they can be instantiated and composed based on the target analytical activity, or modified, or even refactored into new components. To this end, Vistrates follows a literate computing inspired approach\(^1\) by providing an integrated browser-based programming environment using the Codestrates framework \[250\]. While analysts and developers with programming expertise can instantiate, customize, or build components from scratch, Vistrates also supports a point-and-click interface for novices (or for use on mobile devices with limited screen size), where a pipeline can be assembled using a graphical user interface. Meanwhile, all vistrate documents are collaborative, and can be viewed on any

---
\(^1\) Akin to interactive programming in Jupyter \[239\] and Observable \[231\].
device with a web browser.

In this paper, we contribute the Vistrates platform, which applies the concept of information substrates specifically to the activities in visualization and data analytics, while supporting multiple expertise levels, users, and devices. We also illustrate Vistrates and its capabilities using an in-depth motivating scenario on travel journalism in the next section and the accompanying video. Furthermore, we showcase features of the platform in additional examples on (1) wrapping existing toolkits such a D3, Vega, and Plot.ly as components, (2) offloading heavy computations from mobile devices, and (3) using the Vistrate canvas to present the results of an analysis to an audience.

11.2 Motivating Scenario

Vergil is an experienced freelance travel writer. He has been commissioned by a new internet-based travel guide company called “TraLuver” that is trying to “disrupt” the travel guide industry by providing customized travel plans for their clients. Their business idea is to use data science to find an optimal match. TraLuver is rolling out their service to a select few North American cities, and Vergil has been tasked with curating and preparing the dataset for Toronto, Canada.

Prior to starting his field work, Vergil uses his laptop to familiarize himself with the TraLuver platform, which is built on top of a Vistrates installation. Vergil is not a data scientist, so he connects with Daria, an analyst in the data science team at TraLuver’s headquarters. Using videoconferencing and a single vistrate document, Daria takes Vergil on a tour of the basic datasets available including Yelp! businesses and reviews and open data provided by the City of Toronto. She constructs a simple visualization interface, where a map of businesses in Toronto can be filtered to see their ratings in a bar chart, by putting together available components in the vistrate’s graphical interface without any programming. Since Vergil knows he will be restricted to mobile devices when he is out in the field, he installs a mobile view following Daria’s example, which shows one of the available views at a time.

After learning the system, Vergil heads out to the 553-meter CN Tower, a major landmark of the city. He installs a GPS component to center the map in the vistrate to his location. This helps him visit surrounding restaurants and access their reviews on the TraLuver platform. He realizes that the single map view would be more useful if it also incorporated relevant review keywords. He pulls out his tablet, sketches this idea, and discusses it with Daria, who provides a temporary solution by adding a simple word cloud. Daria gets in touch with Sam, a developer in TraLuver, who starts building the new component in a copy of Vergil’s vistrate. After this, Sam calls Vergil and Daria and adds the new Word Map component to their vistrate.

A month later, after hard work by Vergil and his TraLuver team, Daria can finally present the finished Toronto project to the company’s board. Basing her
presentation on the same vistrate that she and Vergil created weeks back, she has created a slideshow of multiple canvases that show each feature, dataset, and visualization of the final product.

![Diagram of ubiquitous analytics](image)

Figure 11.2: Transcending individual silos of activities, collaboration, devices, and expertise to support the vision for truly ubiquitous analytics.

11.3 Vision for Ubiquitous Analytics

The motivating scenario showcases a unique series of activities across multiple users with different expertise and using different devices. The users—Vergil, Sam, and Daria—seamlessly transition between analytical tasks including exploration of multiple datasets, presentation and sharing of ideas and insights, and development of new visualizations. They performed collaborative activities from distributed locations, with Vergil situated in the field as a mobile user. While doing so, they were even able to involve the expert developer—Sam—to meet an analytical need that arose during mobile use of the platform.

The common thread in the scenario is a notion that we coin *transcendence*. Data analysis in the scenario goes beyond individual contexts and applications to encompass a wide variety of analytical activities (A), modes of collaboration (C), types of devices (D), and levels of expertise (E). Transcendence across all four of these “ACDE axes” captures the essence of *ubiquitous analytics* [90], where people use multiple networked devices to analyze data anytime and anywhere. Central to realizing this vision is the ability to transcend each particular silo and consider the analysis as a whole (Figure 11.2).

However, achieving the vision cannot simply be to build a new tool that incorporates all four of the ACDE components [18]. Consider, for example, if Tableau added robust functionality in support of all four. In doing so, all we are left with is a yet another tool that while it has a wider purview than the others, still is a disconnected island separate from the rest of the data science ecosystem. We be-
lieve that instead of focusing on applications, the solution is to focus on *dynamic media* as the core building block of data analysis. Such “dynamic media” consist of information substrates that are *shareable* in that they intrinsically support collaboration, *distributable* in that they can be accessed using any device, and *malleable* in that they can be fully adapted by the user\[169\]. They transcend the classical application boundaries by allowing the user to mix and match tools and representations that typically are siloed in applications for particular domains\[2\].

How would such a “dynamic media” approach to data visualization and analytics look in practice? First and foremost, such an approach would most likely be *based on the web* (C1), which at its core is the closest we have to dynamic media\[169\]. Second, to capitalize on the large and growing ecosystem of web-based visualization resources, the approach must be *open* (C2) and *extensible* (C3) to allow for integrating existing technologies into the platform without duplicating effort. Third, to enable recycling and magnifying prior efforts, components of the platform should also be *reusable* (C4) and *composable* (C5) so that new ideas can build on existing work. Finally, to truly transcend all contexts of activity, collaboration, device, and expertise, the tool should support *multiple, adaptive, and simultaneous layers of access* (C6), enabling teams consisting of the gamut of novices through programmers to analyze data together anytime and anywhere.

11.4 Background

Here we outline the background work in visualization platforms, collaborative visualization, and visualization across heterogeneous devices to define their scope in terms of our design considerations.

11.4.1 Visualization Toolkits, Platforms, and Languages

Data visualizations can be created with many toolkits and platforms, which mainly differ in terms of their target users. Visualization tools for novice users, such as Excel, support basic charting and data transformations. Shelf-based visualization tools such as Tableau\[290\] support easier configuration of visualizations by drag-and-drop into “shelves” of visual variables. For visualization design, iVoLVER\[215\] uses visual programming to help non-programmers extract data from multiple sources and transform it into animated visualizations. iVisDesigner\[255\] supports creation of visualizations by utilizing template-based mappings. More recently, Data Illustrator\[201\] and DataInk\[324\] explore the concept of lazy data bindings for better expressiveness in visualization design.

---

\[2\]This is consistent with the principles of *instrumental interaction*\[28, 29\] from HCI, where instruments transcend individual applications and instead take a document-centric view of computing as documents manipulated using general instruments. In instrumental interaction, you do not open Microsoft Word to edit a document; instead you simply use the text editing instrument to modify the document object, and you can use the same instrument to edit a webpage or a tweet—activities that would normally require separate applications.
Table 11.1: Classification of existing tools and platforms for supporting data analysis based on the ACDE axes from Figure 11.2. A checkmark (√) signifies that the corresponding tool/system/framework has dedicated support for the dimension. Note that Codestrates is the only framework here not developed for data analysis, but it is added here since Vistrates is built on top of it. (ETL implies Extract, Transform, Load.)

<table>
<thead>
<tr>
<th>Systems/Frameworks</th>
<th>Collaboration</th>
<th>Cross-Expertise</th>
<th>Responsive to Devices</th>
<th>Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jupyter [55]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Observable</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collaboratory</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tableau</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3 [20]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Viz Porter [23]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ManyEyes [75]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PolyChrome [7]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Codestrates [58]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Vistrates</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

for non-programmers. Faceted visualization browsers such as Keshif [325] generate predefined facets representing data to support novice users focus on visual exploration.

For development, visualization toolkits such as Protovis [41] and D3 [42] support web visualization with a data model that maps data items to visual marks for SVG-based interactive graphics. D3 also binds the data to the Document Object Model (DOM) of web browsers and supports basic extract, transform, load (ETL) operations to create data objects for custom visualizations. More recently, high-level visualization specification grammars have been developed, such as Vega [269], Vega-lite [270], and Atom [234]. These platforms are oriented towards analysts with technical expertise in visualization development. They offer preliminary means for reusability by maintaining online examples and promoting open source contributions [38].

11.4.2 Supporting Collaboration

Previous collaborative visualization platforms have covered all quadrants of the collaboration matrix [20, 149]. The asynchronous collaboration model is most common for visual analytics, exemplified by the Sense.us [126] and ManyEyes [302] platforms that leverage crowd intelligence through view sharing, annotation, discussion, and social navigation [82]. Methods for maintaining group awareness [118] understanding of collaborators’ work—and coordination [203, 292] have been considered through history mechanisms and notifications [128] as well as data coverage representations and widgets [166, 267, 311].

Specialized techniques for supporting collaborative visualization have also been explored. Collaborative brushing [119, 148] helps build upon the group’s interactions through highlighting of the user selections. Branch-explore-merge [213] presents a co-located group analytic technique for transition between cou-
pled and decoupled work across devices. For co-located collaboration, Lark [296] allows multiple users to work together by directly manipulating the visualization pipelines. Frameworks have also been created for developing distributed visual analytic interfaces, such as Munin [17] and PolyChrome [15]. These efforts support requirements for networking devices and connecting multiple users in specific scenarios, but do not offer concrete means for transcendence on the ACDE axes described in Section 11.3.

11.4.3 Heterogeneous Devices for Visual Analysis

Another significant effort in visualization has been to utilize heterogeneous devices [187]. SketchStory [188] uses touch input and sketching as a means for storytelling. VisPorter [65] enables cross-device interactions for visual exploration between large displays and smartphones. GraSp [108] promotes flexible workflows involving physical navigation and remote interaction between handheld devices and large displays. VisTiles [182] presents the styles in coupling multiple small-screen devices to explore data. Recently, a conceptual framework for combining multiple devices—large displays and smartwatches—for visual analysis has also been developed [135]. These advances exemplify the fact that visual analysis has outgrown traditional desktop scenarios [256] to support advanced applications involving heterogeneous devices.

11.4.4 Interactive Notebooks

Interactive notebooks have recently gained popularity in data science and visual analytics, as they promote collaboration by sharing. They adopt a literate computing-based approach to programming, where executable code is interweaved with text and images to create interactive narratives. Jupyter Notebook [239] is a web-based interactive notebook that connects to a kernel capable of executing code in languages such as Python, R, Ruby, JavaScript and many more. Jupyter provides integrations with analytics and visualization frameworks such as SciPy [273] for analytics and Altair [4] for declarative visualizations. However, Jupyter does not support real-time collaboration out of the box. Google’s Colaboratory [108] is a Jupyter implementation using the Google Drive backend for real-time collaboration. However, collaboration is on the level of editing and not interaction. Observable [231] is a JavaScript-based interactive notebook primarily designed for creating interactive visualizations. It provides a reactive programming model where re-execution of a code-block will result in a re-execution of any code block that depends on it. While notebooks written in Observable are easy to fork and share, they do not yet support real-time collaboration neither in editing nor interacting with the produced visualizations. Furthermore, they are not focused on analytical work across heterogeneous devices and activities in the sensemaking beyond visualization development.
11.4.5 Webstrates and Codestrates

Vistrates is built on top of Webstrates and Codestrates. Webstrates is a web framework where webpages are made collaboratively editable in real-time. Changes made to the DOM of a webpage (called a webstrate) are persistent and synchronized to all clients of the webstrate. Codestrates is an authoring environment built on top of Webstrates. A codestrate is a webstrate that includes tools for editing its own content, including writing and executing code, following a literate computing approach similar to interactive notebooks. Individual codestrates contain their implementation, which means they can be reprogrammed from within. A codestrate is structured in sections consisting of paragraphs of code, data, styles, and web content. Sections can be turned into packages of functionality that can be shared between codestrates. Similar to Observable, Codestrates is JavaScript-based and all execution happens in the run-time of the browser. This also means that it integrates well with existing web libraries.

11.4.6 Summary of Related Work

Table 11.1 contrasts the related work against our Vistrates platform. Overall, Vistrates is inspired and influenced by many related works. It is capable of leveraging frameworks such as D3, Vega, and Plot.ly, while promoting reusability and extensibility in development. It uses Codestrates and Webstrates as the underlying infrastructure specifically to (1) support interactive programming for development/extension of vistrate components, (2) synchronize documents across devices and users, and (3) utilize package management to maintain reusable components. By doing so, it extends the related work to ubiquitous visual analytic scenarios by defining a visualization component structure and enabling many activities in the sensemaking spectrum.

11.5 Designing Vistrates

Vistrates is a design proposal for a component model for ubiquitous analytics, but also a proof-of-concept implementation. It is a realization of a set of design choices that together form the vision of a holistic and sustainable data analysis and visualization environment. The design of Vistrates is rooted in the principles from Webstrates that software should be malleable, shareable, and distributable. With Vistrates we assume that all software is expressed as (one or more) webstrates, and hereby we inherit the quality of Webstrates that shareability is a fundamental premise; webstrates can be edited in real-time by remote users, and easily shared by passing around a URL. As webstrates are webpages, they can be accessed from any device with a browser, which means that distributability is ensured. To realize a visualization environment that can enable the motivating scenario and adhere to the ideals for ubiquitous analytics, we introduce the following six design principles to support analytics.
Component instantiation: Existing component prototypes are available to be edited, thus, promoting reusability and extensibility. **Pipeline view:** This view supports configuration through interaction—drop-down selection—to create visualizations and interactively explore data without programming. In this example, a crime dataset from Baltimore, MD is visualized through a map and bar charts for crime type and weapons by aggregation. A filter component is added to filter the bar charts based on the selection on the map. **Dashboard view:** This vistrate view creates a grid layout for visual exploration of the data and annotation using rich text. **Development view:** The lowest level of abstraction for a vistrate in which a programmer can edit the code and create new visual analytic components.

### 11.5.1 Component-based Pipeline Architecture

The typical architecture to go from data to visualization is through a visualization pipeline [60]. We propose a component-based architecture, where components (Figure 11.3(a)) are connected together in reconfigurable pipelines (Figure 11.3(b)). A component can be a data source (e.g., serving a file, connecting to a database or API, or providing coordinates from a phone’s GPS module), a computation on data (e.g., filtering, aggregating, or analyzing), or a visualization (e.g., a bar chart, scatterplot, heatmap, etc.). Visualizations do not have to be end points in the pipeline, but can be interactive and hereby serve as data sources as well. Components should be executable blocks of code with an optional input, output, state, and view (Figure 11.3(d)). The pipeline should be reactive, so when the output of a source component changes, it will trigger updates of components that have the output of the source component as input. Components should adhere to a minimalistic interface for connecting them together, and what a component does and what third-party software libraries it uses should be up to the developer.

### 11.5.2 Collaborative Pipeline

It should be possible to modify, run, and interact with components in the pipeline collaboratively from different clients. However, for most computations it is faster and simpler to execute them locally than to distribute data across a potentially high-latency network. We therefore propose a design where each client executes its own instance of a pipeline, but synchronizes state locally to components between
them. State includes the configuration of a component and any application state that should be synchronized or persisted, e.g., interactions. An example of the latter could be a rectangular selection on a map-based visualization, or a URL to a data file that a data source component should load into memory. It should be up to the developer to specify what application state is synchronized, allowing, e.g., the developer of the aforementioned map component to specify that selections should be synchronized but not, e.g., zoom levels. Components should synchronize execution between clients, i.e., rerunning a component on one client should trigger reruns on all other clients as well. In other words, the collaborative pipeline principle consists of (1) a reactive data flow, (2) a shared execution flow, and (3) shared component state between clients of the same vistrate.

11.5.3 Prototype-based Components

Components should be prototype-based, and components should be instantiated by copying from a prototype and configuring the instance into the pipeline. An instance of a component should contain its own implementation and its state. This is a deliberate violation of the software architectural principle to avoid code duplication. However, by having components contain their own code they become directly reprogrammable, allowing a user to reprogram a single component and potentially turn it into a prototype for new components.

11.5.4 Multiple Levels of Abstraction

Users should be able to work on multiple levels of abstraction: from programming components, to configuring components in a pipeline, to creating presentations of the visualizations and to interact with said visualizations. At the lowest level of abstraction, all aspects of components should be manipulable as code. At a higher level of abstraction, components and their pipeline should be reconfigurable in an interactive fashion, allowing for even non-programmers to reconfigure without programming (Figure 11.3(b)). At an even higher level of abstraction, visualizations should be treated as content that can be composed, e.g., in the form of a slide deck, a document, or a dashboard (Figure 11.3(c)). Collaboration should be possible on each level of abstraction—from writing the code to interacting with the visualizations—but it should also be possible to collaborate on different levels of abstraction at the same time. That is, while one user is interacting with a component, it should in principle be possible for another user to reprogram it and re-execute it without requiring the first user to restart their client.

11.5.5 Component Repository

It should be possible for a “programming literate” user to develop their own components or redevelop other people’s components. However, we also wish for a non-programming-savvy user to be able to construct a visualization pipeline using
components made by others. Components should, therefore, be shareable through a common component repository where users can publish their components, as well as retrieve components made by others.

11.5.6 Transcending Application Boundaries

It should be possible to integrate visualizations directly into other types of media (e.g., presentations or reports). The components and pipeline should therefore co-exist in an open-ended software ecology with tools not only designed for analytics and visualization work.

11.6 Implementation

Vistrates is implemented using standard modern web technologies—JavaScript, HTML, and CSS—as well as Codestrates and Webstrates. It uses Codestrates’ literate computing and package management features. Vistrates consists of a core framework package and individual components implemented as packages (Figure 11.4).

In Codestrates, software is implemented in paragraphs grouped into sections. There are four basic types of paragraphs: code paragraphs containing JavaScript that can manually be executed by the user, toggled to run on page load, or be imported into other code paragraphs; style paragraphs containing CSS rules; body paragraphs that can contain any web content expressible as a DOM subtree; and data paragraphs containing JSON formatted data that can manually be edited by users, or programmatically through JavaScript. Every paragraph can be given

Figure 11.4: Vistrates architecture and relation to Codestrates/Webstrates.
a human readable name, a unique identifier, and a list of class names. Every paragraph can be toggled to be shown in full-screen either only local to a particular client, or for all clients. Vistrates utilizes these abstractions—paragraphs and sections—and defines an update logic and data flow between them, turning them into building blocks for visualization components and analytical activities.

11.6.1 The Core Framework

At its core, the Vistrates framework governs the control flow through component pipelines using the principles of inversion of control and dependency injection of components. The backbone of Vistrates is a singleton that registers all components in a vistrate, a component class that implements an observer pattern for connecting the input and output of components together, and an execution model for executing user-provided component code.

On load, the Vistrates singleton registers all existing components and registers observers between them. When components are updated or new components are created, the singleton also updates observers accordingly. All components have a controller that implements an observer pattern, such that the appropriate components in the pipeline are notified when the output of a component changes. A component consists of three paragraphs grouped in a section: a code paragraph, a data paragraph, and an optional view paragraph (web content paragraph).

The code paragraph of a component includes the definition of specific methods and properties following the format shown in Listing 11.1, which include the fields `data`, `src`, `props`, and `libs` as well as the methods `init`, `destroy`, and `update`. All fields and methods in a component are optional, but defines how a certain component can function within the pipeline. Vistrates uses regular DOM IDs as references, and the `data` property contains the ID of the component’s data paragraph containing configuration and stored state. The `src` property defines named sources that can be referred in its methods, and `props` refers to named properties of the sources that can be remapped dynamically through its configuration data. If a component does not have source references, it can only function as an entry node in the pipeline, which is typical for components that load data into the vistrate. `libs` is a list of references to JavaScript libraries that the component depends on. The references can either be URLs to external files, or file names of files uploaded as assets to the webstrate.

The first time the code paragraph is executed, a controller object is instantiated from the controller class, and the properties and methods defined in the code paragraph are evaluated and copied to the controller object. If the controller references anything in `libs`, these are downloaded and evaluated before the `init` method is run. After `init` the `update` method is called, and it is subsequently called any time any of the component’s sources update their output. Code paragraphs can be rerun by pressing the play button, and whenever this happens, the previous `destroy` method is executed (to, e.g., remove event listeners) and the newly defined properties and methods are hotswapped on the controller object.
### 11.6. IMPLEMENTATION

| vc = {
| data: "id-of-vis-data",
| src: ["mySourceName_1", ..., "mySourceName_n"],
| props: ["myProp_1", ..., "myProp_n"],
| libs: ["myLibraryStoredAsAsset.js", "https://somecdn.com/anotherLibrary.js"],
| init: function() { /* code goes here */ },
| destroy: function() { /* code goes here */ },
| update: function(source) { /* code goes here */ }
| }

Listing 11.1: The code paragraph template.

| { config: {
| src: {"mySourceName_1": "source_1_id", ..., "mySourceName_n": "source_n_id"},
| props: {
| "myProp_1": { "src": "mySourceName_1", "prop": "somePropOnSource"},
| ..., "myProp_n": { "src": "mySourceName_n", "prop": "someOtherPropOnSource"}
| },
| view: "id-of-vis-view"
| },
| data: { /* The data field of the component for storing state */ } |

Listing 11.2: The data paragraph template.

Updating the output property of a component will trigger the update method on any observing components.

The **data paragraph** contains the configuration of the component and the shared state, encoded as JSON. The data paragraph template has the format shown in Listing 11.2. Besides observing the sources of a component, a Vistrates controller also observes its data paragraph and changes to the configuration will trigger changing dependencies to be hotswapped and changes to the state will trigger the update function, and thereby immediately be reflected in the component views on all clients. The chosen configuration of a component includes the mapping between source/property reference names and the actual ids in the vistrate, which allows users to change the mapping on the fly. This format was chosen to be able to reference specific data items in the output of a source component without forcing developers to follow a rigid output convention. As an example, the source with reference name mySourceName_1 that is currently mapped to source id source_1_id in Listing 11.2 can be changed to refer to another source id simply by changing this mapping. Similarly, the property myProp_1 is mapped to a specific data item in mySourceName_1. The configuration also includes a reference to the view paragraph. The shared state of a component can be encoded in the data field of the data paragraph, which for instance can be the interaction state of a visualization. Webstrates will by default synchronize the DOM elements outside transient HTML tags, hence the state encoding will also be synchronized across clients. We have chosen an open format for the declarative state specification, which means that it is left to the developer to define this encoding and how to behave accordingly in the update method. The data paragraph can be edited by the user, or updated programmatically, e.g., by the pipeline view described below.

The **view paragraph** is a body paragraph containing the visual output of a
component. The content can be any standard web content, which is then wrapped in a transient element. Transient elements are Webstrates-specific DOM elements that do not have their state synchronized nor persisted. This means that the content of views are not shared across clients. Clients share code and data paragraphs, but clients are executing their own pipeline and thereby creating the content of their own views. This makes it possible to have views where not all interactions are shared between clients. As an example, a map component can share area selections by writing those selections to the data paragraph, while at the same time allowing each user to define their own viewbox and zoom level.

In the controller code, the view can be referred through the view property and its content can be replaced by setting the view.content property either to an HTML string or a DOM node reference, or by referring to the root DOM element of the view using the view.element property. Finally, style paragraphs can be added to define the appearance of a vistrate view.

Component updates in Vistrates are triggered in two ways: (1) when the output of a source is updated, or (2) when the configuration or the state in the data paragraph is updated. The cause of an update is encoded in the first argument in the update method, which can be a specific source from the src list, the configuration, or the state. We chose this design as it allows the developer to update a visualization differently based on the type of update; if the data input changes the visualization needs to be redrawn, but if only the interaction state changes the visualization can be updated in a different manner: say, by highlighting specific visual marks. It is possible to create update cycles between components, but it is up to the developer to ensure that these cycles are finite. For instance, such update cycles are currently used to develop coordinated multiple views with brushing-and-linking [257].

In essence, Vistrates adapts these paragraphs to visualization and analytics. In contrast to Codestrates, paragraph definitions in Vistrates have an analytical value—the code captures the underlying logic for processing and visualizing data, the data paragraph captures the declarative specification to map properties to visual variables, and the view contains the analytical outcome of the component. Furthermore, Vistrates explicitly defines the update logic and control flow across components made from these paragraphs. Figure 11.5 shows an example component including controller (code), data, and view paragraphs that calculates the average of a data column and views the result.

### 11.6.2 The Pipeline

Vistrates components are composable through the configuration specification in the data paragraph. The pipeline view is an abstraction layer on top of the textual specification, which provides graphical access to the configuration and composition of the components in a vistrate. In the pipeline view, components can be reconfigured and recomposed at any time, and changes are immediately reflected in their output, which also triggers updates of connected components.
Figure 11.5: A simple Vistrates component that calculates the average of a single numeric data column. The component is easily reconfigurable by changing the mappings in the data paragraph. Other components can observe the output and act accordingly when it changes.

The components’ views can be inspected within the pipeline view to immediately observe the effects of a reconfiguration or recomposition. The pipeline view is itself a component that observes the state of the pipeline through an observer installed on the Vistrate singleton. This means that the core of the pipeline view also follows the standard component template with a code paragraph, a data paragraph, and a view paragraph. The pipeline is easily reprogrammable or replaceable with another abstraction layer. Our current pipeline view is a basic graph implemented in D3 with unfoldable nodes that can either contain the view or the configuration of a component. The pipeline can be accessed in a vistrate through a keyboard shortcut or by pressing the pipeline button in the global toolbar.
11.6.3 The Component Repository

The component repository is implemented using Codestrates’ package management features [39]. Prototype components can be pushed to or installed from a repository. New instances of an installed component can be created through the “Create new Vistrate Component” dialog accessible through the global toolbar (Figure 11.3(a)). Instantiating a component will copy the selected prototype, insert the given name and ids, and add it to the vistrate document.

Any component can be turned into a reusable prototype by making it a package and pushing it to the repository. This adds metadata to the component including a short description, a list of assets (e.g., images, JavaScript libraries, or CSS files), dependencies to other packages, and a changelog. This approach allows for reappropriation and customization of existing components. Components in the repository are also ready to use, meaning that an instance can immediately be configured using the pipeline view and, therefore, allows users to create visualization pipelines without programming. The current Vistrates component repository contains components for standard visualizations such as the bar chart, pie chart, line chart, geographical map, scatterplot, parallel coordinates, etc., components to analyze, transform, combine, and filter data, as well as utility components to load data, spawn headless browsers, and offload heavy parts of the pipeline to strong peers.

11.6.4 Component Canvas

Space is an important cognitive resource; we think and work in physical space [8]. Our implementation of the “Vistrates Component Canvas” package, therefore, allows for such spatial arrangements of component views on a 2D canvas. This can facilitate the sensemaking process to externalize thoughts and for distributed cognition during collaborative work, or it can become a dashboard for interacting with the visualizations (Figure 11.3(c)). In addition, users can add rich text and other media supported by HTML5 and annotate the canvas with a digital pen. Any content on the canvas—including component views—can be moved, scaled, and rotated. When installed, the Vistrates interface has a button in the global toolbar to add a component canvas paragraph. A canvas paragraph is styled to look like a whiteboard.

11.6.5 Mobile List View

In contrast to the component canvas, the “Vistrates Component List View” displays a single (selected) component view at a time. It provides a responsive component container that scales component views according to a device’s available screen real-estate, e.g., to fully show a component view on a smartphone (Figure 11.7). Any available component view can be picked from an action menu.
11.7 Vistrates In Action

Here we explain the workflow when using Vistrates and describe how it can be used to realize the motivating scenario (Section 11.2).

11.7.1 The Workflow

A fresh Vistrates workflow begins with a user creating a codestrate with the Vistrates package installed. It can be copied by appending ?copy to the URL. Then different visualization and analytics components can be installed by accessing the Vistrates component repository.

The first component a user will need is a component for loading data. The user will, e.g., create an instance of the CSV loader component from the component menu (Figure 11.3(a)). The user can now use its user interface (in its view) to upload a CSV file that will be stored as an asset on the webstrate layer. For larger datasets, a database component can be used that uses Webstrates’ searchable CSV asset API to store data [74]. Additional visualization and analytics components can be instantiated and connected in the pipeline view (Figure 11.3(b)), or by manually editing the configuration in their data paragraphs.

To create a new component from scratch, the user instantiates a blank component and edits the code paragraph. A new component can also be created by modifying an instance of an existing component. To share a new component, it has to be turned into a package. Finally, there is no clear boundary between development and deployment in Vistrates since all changes are immediately reflected in the document. To get an application-like state, developers can simply use the persisted fullscreen functionality. For instance, to create a dashboard, the user can install the canvas package, create a canvas, expand it to fullscreen, and add components to it through a context menu.

11.7.2 Realizing the Scenario

In the following, we describe how Vistrates can realize the motivating scenario (Section 11.2). The accompanying video to this paper showcases a realization of the motivating scenario built with Vistrates.

Starting Out

To begin with, Daria can use a CSV loader component as a prototype for creating components containing the datasets needed by Vergil, and add these datasets to the component repository. Daria can create a blank codestrate with the Vistrates package installed and share its URL with Vergil. She and Vergil can use a WebRTC-based communication and remote pointers from Codestrates to communicate while Daria explains how Vistrates works. Together they can add existing components from the Vistrates component repository, instantiate them, and use the pipeline view to connect them together to, say, connect a ratings bar chart to
a map component, so that Vergil can filter and choose businesses based on their ratings. The pipeline view helps transcend user expertise as pipelines can be configured by non-programmers such as Vergil through a graphical user interface.

**Extension**

The developer, Sam, can create a new component for the custom visualization (a Word Map) following the component structure (Listings 11.1 and 11.2) and add it to the component repository to enhance the analytical tools available to Vergil. Sam can integrate external libraries such as Leaflet [185] for drawing maps. Other components are based on graphing libraries such as Plot.ly [244] and Vega-Lite [270] (see details in Section [11.8]). To test his component in Vergil’s pipeline, Sam creates a copy of the Vergil’s vistrate and integrates his component.

**Mobile Use**

Vergil can install a list view that allows for navigating the views of components on a mobile device. He can also connect the GPS data from his phone—provided by the JavaScript geolocation API—to a map component to center the map on his current geographic location. The Word Map that Sam has developed relies on computing tf-idf scoring [282] and word embedding, which is too heavyweight for a mobile device. Daria can add a component to the beginning and end of the heavy part of the pipeline so that mobile devices will offload that part of the computation to a desktop client or the server (see Section [11.8.1]).

**Transcending Activities**

Vergil can use Codestrates’ rich-text editing capabilities to write reviews and take notes. He can use the canvas component for sketching, and also arrange selected component views in a 2D layout. This allows Vergil to explore the data in a dashboard on his personal computer. Finally, the view paragraphs can be turned into slides allowing Daria to create a presentation (see Section [11.8.3]).

**Shareability and Collaboration**

Vistrates is developed for collaborative analytical workflows. Vergil, Daria, and Sam utilize the video conferencing components to talk to each other during the analytical process. Interaction is synchronized across instances of a vistrate, while presenting remote pointers for collaborators, to support synchronous collaboration between the actors. For instance, using this, Vergil is able highlight a region of Toronto to Daria as a developing area that can be explored. Vistrates applies a relaxed WYSIWIS model [285] to collaboration, which means a client can have, e.g., a canvas in full screen while another shows the code or pipeline view while their content and execution is kept synchronized.
11.8 Additional Examples

In this section, we will elaborate on a few examples that showcase the expressive power of the Vistrates platform.

11.8.1 Computation Offloading

When all clients of the same vistrate execute their own code, it is possible for weaker peers to offload heavy computations to stronger peers. This can even be an entire subpart of the pipeline, as the example in Figure 11.6 shows, where the highlighted part of the pipeline is offloaded to stronger peers. Two components called Heavy Start and Heavy End handle the offloading. The start node will signal help to other clients if it is executed on a mobile device and pick one of the stronger clients that offers help. The communication between clients is realized using the Webstrates signaling API [281]. The chosen client will then execute their pipeline using the input of the weak client. When the heavy end node is reached, the strong client will provide the weak client with the result, and the weak client can then continue its own execution. This principle also works for multiple heavy start and end nodes. If no strong peer is available, a service implemented on the Webstrates server can be called through an HTTP request to spawn a headless browser instance pointing to the given vistrate. Beyond this, a Heavy Data component is also available to avoid attempting to load large datasets on weaker clients. This way, a client can present interactive visualizations without having to load the dataset.

This approach to computation offloading is implemented purely as new components without any changes to the core of Vistrates. This means that components
for different approaches to distributed computing could be created, e.g., to support the kind of peer-to-peer distributed computation provided by VisHive [76].

11.8.2 Cross-Device Visualization

A vistrate can be opened on any device with a web browser. This provides an opportunity to create physical dashboards across multiple devices and for mobile use of vistrates. In Figure 11.7, we showcase a physical dashboard with two mobile phones and a tablet (inspired by VisTiles [182]). This dashboard is created by installing a Mobile List View on the vistrate from Figure 11.3 and selecting a single component view on each device. The phones show visualizations of aggregated crime data—crime type and weapon used—and the tablet show a geographical map of all the crimes in Baltimore. Filtering a view by interaction will trigger updates to synchronize views on other devices owing to the collaborative pipeline of Vistrates. By introducing heavy nodes, the phones never execute any aggregation, filtering, or analysis, but they can show the visualizations and support interaction.

11.8.3 Integrating Visualizations in a Slideshow

As Vistrates is built on top of Codestrates, it is out-of-the-box compatible with, e.g., a slideshow package from Codestrates. This package wraps a view paragraph into a slide as part of a slideshow—including cross-device presenter notes and remotepointing (Figure 11.1). Each slideshow can be styled by creating a theme—a set of stylesheets—that can specify how the visualizations and text appear in the presentation. Visualizations in a vistrate, when wrapped in a slide, are still
interactive as their event handlers are retained, and this enables interactive and
dynamic presentations when utilizing Vistrates.

11.8.4 Creating Vistrate Components from Existing Libraries

With Vistrates, it is easy to create new components by wrapping visualizations
built in existing libraries such as Plot.ly, Vega, or D3 (e.g., Figure 11.1). This
provides developers with a powerful set of tools to reuse developments not made
within Vistrates, but wrap these in the Vistrates data and synchronization flow. It
further enables developers to utilize existing charting and interaction primitives in
these libraries. As an example, Vistrates allows for making Plot.ly visualizations
collaborative so that they can be used to create filters on the data, which can be
pipet to other analyses. The code in Listing 11.3 showcases how Plot.ly can be
used to initialize a line chart, save the line chart selection in the data paragraph,
and update selections across clients.

11.9 Discussion

A unified approach to transcend analytical activities, device ecologies, levels of
expertise, and modes of collaboration has the potential to bridge the gap between
data science and visualization [2] as well as the gap between research and real-
world adoption. Vistrates is a proof-of-concept implementation that shows the
technical feasibility of such an approach. While it is a step towards fulfilling our
vision for ubiquitous analytics [90], the vision will not be accomplished by a single
group of researchers, and as such the current framework and implementation also
comes with limitations.

Listing 11.3: Creating a Vistrate component from a Plot.ly line chart.

```javascript
vc = {
  ...  
  libs: ["plotly.min.js"],
  init: function() {
    this.plotDiv = document.createElement("div");
    this.view.content = this.plotDiv;
    this.update_view = () => {
      let layout = {"xaxis.range": this.data};
      this.Plotly.relayout(this.plotDiv, layout);
    };
    ...  
    this.draw_plot = () => {
      ...  
      this.plotDiv.on("plotly_relayout", (eventdata) => {
        this.data = [eventdata["xaxis.range[0]"], eventdata["xaxis.range[1]"));//
      });//
    },
    update: function(source) {
      ...  
      this.update_view();
    }
  }
};
```
11.9.1 Limitations and Future Work

**Scalability.** All code in Vistrates is currently executed within a browser, which has at least two limitations: (1) reliance on JavaScript and (2) the available computational power. Data scientists often use languages such as Python and R that provide multiple efficient libraries for data transformation and machine learning, which JavaScript does not provide to a similar extent. As a next step, we are making Vistrates a mixed environment, where, e.g., data analysis components developed in Python can be executed in the cloud and interleaved with visualization components developed in JavaScript. Beyond this, it can be challenging to deal with large datasets and execute complex computations driving the data analysis in a web browser. Our computational offloading functionality showcases a first step to support mobile devices. In fact, this feature enables viewing the word cloud and Word Map visualizations on mobile devices by offloading the tf-idf analysis and layout computation on the large Toronto datasets to a computer (see supplementary video). However, this offloading is still limited since it just executes the code on a different device with better capabilities. A future work is to extend Vistrates to support distributed and parallel methods (cf. imMens [198]) and progressive visual analytics [287] for better scalability. Finally, the downside of a vistrate containing its entire source code is that the web browser needs to download all the required resources on page load to execute the vistrate. This can be slow on low bandwidth connections. In the future, we plan to add a lightweight mode to Vistrates where resources are loaded on demand.

**Usability.** Another limitation of the current implementation is its usability. Vistrates supports multiple levels of abstraction through development, pipeline, and canvas views. However, the current proof-of-concept is centered on a linear development view based on literate computing. More abstraction levels should be available to support certain activities and users: (1) shelf-based configuration such as in Tableau and Polestar to assemble components, (2) provenance tracking using interaction and insight histories [130] for visual exploration, and (3) better mobile interfaces driven by responsive visualization [10, 16].

**Flexibility.** The core framework in Vistrates is designed to be flexible, with only a few constraints such as the usage of predefined `src`, `props`, and `output` properties described in Section [11.6.1](#). Therefore it is difficult at this stage to provide guidelines for a good component design since the components can be defined at multiple granularities. For instance, an aggregation algorithm can be developed to be a standalone component as in our current approach or integrated into a visualization component. The former is better suited for extensibility and recomposability, while the latter leads to a simpler pipeline view that is easier to use. This tradeoff between flexibility and usability exists when creating new components within Vistrates. To answer this challenge, we are currently developing visual analytic templates—groups of components for specific data types and tasks—that are more meaningful and easily extensible by target users.

Another key property of the Vistrates framework is that once users start to
gain expertise, the environment allows them to exercise that expertise by transcend- ing abstraction levels. This property along with the component repository offers flexibility to develop new abstraction levels. However, leveraging this flexibility can be challenging, especially for non-programmers. To answer this, solutions at multiple levels need to be investigated including documentation standards, design patterns, and community support to share knowledge in Vistrates.

11.9.2 Implications

With Vistrates, we hope to advance a trend of creating software environments that can support computational thinking [313]. We believe that there is not only a pedagogical and educational challenge to heighten computational literacy in our society, but also a need for tools that make it easy to exploit the power of programming and bridge the skills of experts and novices. Vistrates can also support the design of learning environments for analytics and visualization, where the complexity level can be increased gradually as students improve their skills. Vistrates continues a trend set by the work on interactive notebooks for creating reproducible data analysis and science [277]. The principles applied in the design of Vistrates allows for the creation of digital artifacts that can include interactive visualizations and allow the user or reader to experiment with the data, but also to peek behind the scenes and tinker with the implementation.

11.9.3 Universal Component Model for Ubilytics

Our work on Vistrates have given us a unique perspective on how future component models for ubiquitous analytics should be designed:

**Standardized Integration and Embedding.** Vistrates provides a framework to make most visualizations collaborative and composable, but existing libraries should also have some key functionality to make them easy to integrate. All libraries already have methods to (1) translate data to visualization, and (2) capture user interactions. But they should also contain methods to (3) extract the predicates for user interactions [269], and, most importantly, methods to (4) update visualizations based on the interaction predicates, such that they can be synchronized across clients. The Plot.ly line chart is an example of a specific visualization that fulfills all these requirements.

**Declarative State Specification.** The Vega [269] grammar showcased the power of declarative specifications for creating interactive graphics. Similar ideas are needed to specify the state of a visualization, such that the state can be synchronized across clients. In Vistrates, this is currently left entirely up to the developer, which has its merits and drawbacks. It would be interesting to study whether Vistrates could support developers in creating declarative specifications of visualization state and interpret them again.
11.10 Conclusion

We have presented Vistrates, a holistic and sustainable data analysis and visualization environment designed using the principles of malleability, shareability, and distributability. Built on top of the Webstrates [169] and Codestrates [250] platforms, Vistrates provides a collaborative pipeline that supports the full range of visualization and analysis activities—including data management (ETL), exploration, sensemaking, and presentation—for teams of collaborators of diverse expertise—including developers, analysts, and laypersons. Using the basic Vistrate platform, we have built an initial component model that enables bundling disparate visualization and analytics functionality into reusable prototype-based components. These components can easily be instantiated and plugged into the data flow pipeline, even using drag-and-drop on a mobile device, and can be optionally customized down to the actual source code itself. We have demonstrated the utility of Vistrates in a scenario involving data-driven travel planning, as well as in three examples involving server-side computation, wrapping existing web components, and cross-device visualization.

In the future, we envision using Vistrates as a platform for a multitude of visualization projects. The fact of the matter is that the core Vistrate features are simply too convenient to give up, and they come at minimal cost; building a Vistrate component instead of freestanding D3 or Vega code will make the result collaborative, shareable, and persistent. However, beyond such convenience arguments, an important future research activity is to look deeper into the broad topic of component models for visualization and analytics. The component framework we have built using the basic Vistrate platform for this paper is merely a suggestion, and we claim by no means that it is a final, definite component model. More work on this is necessary to realize the grand vision of a universal visualization platform [18].

Acknowledgments

We are grateful to the anonymous reviewers for their feedback. We also thank Brian Ondov for the voice-over of the accompanying video. This work was supported by the U.S. NSF award IIS-1539534, the DABAI project (IFD-5153-00004B), and the Aarhus University Research Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the respective funding agencies.
InsideInsights: Integrating Data-Driven Reporting in Collaborative Visual Analytics

Andreas Mathisen\(^1\), Tom Horak\(^2\), Clemens N. Klokmose\(^3\), Kaj Grønbæk\(^1\), & Niklas Elmqvist\(^4\)

\(^1\)Department of Computer Science, Aarhus University, Denmark
\(^2\)Interactive Media Lab, Technische Universität Dresden, Germany
\(^3\)Department of Digital Design and Information Studies, Aarhus University, Denmark
\(^4\)College of Information Studies, University of Maryland, College Park, USA

Abstract

Analyzing complex data is a non-linear process that alternates between identifying discrete facts and developing overall assessments and conclusions. In addition, data analysis rarely occurs in solitude; multiple collaborators can be engaged in the same analysis, or intermediate results can be reported to stakeholders. However, current data-driven communication tools are detached from the analysis process and promote linear stories that forego the hierarchical and branching nature of data analysis, which leads to either too much or too little detail in the final report. We propose a conceptual design for integrated data-driven reporting that allows for iterative structuring of insights into hierarchies linked to analytic provenance and chosen analysis views. The hierarchies become dynamic and interactive reports where collaborators can review and modify the analysis at a desired level of detail. Our web-based INSIDEINSIGHTS system provides interaction techniques.
12.1. INTRODUCTION

Modern society—from business and journalism to medicine and policy—is increasingly data-driven. Data analysis itself is iterative, non-linear, and fragmented with both top-down and bottom-up characteristics [159, 241, 299]: sometimes the analyst lets the data itself inform assessments in an exploratory fashion, and sometimes the analyst will formulate priori hypotheses that are tested in a confirmatory fashion. Furthermore, data analysis is seldom confined to a single person; multiple collaborators will often contribute to the same analysis, or intermediate results can be shared with stakeholders. Thus, an important aspect of collaborative data analysis is to enable communication of insights even before the analysis has concluded [203, 267, 331]. However, transferring knowledge is especially difficult for non-routine and ill-defined tasks [276], hence effective communication requires a correspondingly rich mechanism to match the complexity of exploratory data analysis. In the context of reporting insights, a proven communication method is data-driven storytelling [103, 274], but such mechanisms are not well integrated within the iterative and branching nature of sensemaking. For example, Tableau [290] allows users to build stories to convey their insights, but this is done as a separate task once data analysis has concluded. Similarly, computational notebooks [165, 262, 300] (such as Observable and Jupyter) have limitations when used for exploration and presentation simultaneously. Furthermore, slideshows and notebooks alike force analysts to linearize their findings and fixate on the level of detail presented to the audience, which again is not commensurate with its incremental nature.

In this paper, we present INSIDEINSIGHTS, a conceptual design and a web-based system for capturing both low-level insights and high-level abstractions during data analysis. The idea is to support data-driven reporting as an integral part of the analysis process by bridging proven concepts from literate computing, provenance tracking and storytelling. In essence, InsideInsights is a hierarchical insight management system allowing analysts to interchangeably (1) coalesce findings into higher-level abstractions, and (2) subdivide composite assumptions into items that can eventually be validated. The result is a multi-level report supporting hierarchical (pyramidal) thinking [218] that a collaborator or stakeholder can expand to the desired level of detail in order to understand the findings and the process. Our system (Figure 12.1) provides a structured data-flow environment where analysts can load, transform, and visualize data on an interactive canvas. Thus, InsideInsights supports the entire sensemaking loop [241], from information foraging and marshalling, to schematizing and presentation.

The proposed concepts are a step on the way to what we call literate analyt-
ics, where the goal is to document not only the final outcomes, but also provide a narrative of the actual analysis process itself. We envision that by unifying the analysis and communication components of sensemaking, our tools can better support the dynamic data analysis scenarios that organizations increasingly face. The contributions of our work are the following: (1) a conceptual design that address core challenges for integrating data-driven reporting by bridging literate computing, provenance tracking, and presentation techniques. (2) the InsideInsights system, a proof-of-concept implementation using modern web technologies; (3) application scenarios of the system to two real world datasets; and (4) findings from an expert review indicating the soundness of our approach.

12.2 Background

In this section, we review existing work on supporting exploratory data analysis and data-driven communication.

12.2.1 Tracking Exploratory Data Analysis

Exploratory data analysis (EDA) is typically an iterative process that involves multiple cycles of, e.g., cleaning, modeling, visualizing, and interpreting data. Consequently, analysts will often try multiple analysis avenues, of which several will be dead ends, until they arrive at usable insights and conclusions. Furthermore, exploration often proceeds iteratively through several levels of abstraction: sometimes from low-level insights to high-level explanations, and sometimes from general schemas to concrete data. In this work, we are interested in the entire data analysis pipeline, from cleaning and modeling to visualization and interpretation.

Maintaining an accurate record of this non-linear process is challenging, and has led to a large body of research focused on provenance tracking to support recall, replication, communication, and presentation. Interactive visualization is particularly useful for communicating provenance, and several systems thus incorporate provenance tracking. For example, GRASPARC tracks exploration histories as a branching hierarchy, which was later extended by Shrinivasan and van Wijk by allowing users to append annotations. VisTrails was an early provenance management system for scientific visualization and analysis workflows. Similarly, the Burrito system automatically captures low-level computational activities to allow researchers to compare different analysis stages. Gotz et al. instead used such low-level user events to infer high-level semantic meanings for the analytic process. Similar to our approach, this results in a hierarchical structure but is fixed to predefined task classes and probably is not able to adequately represent the analysts mental model. Further, as with all approaches of fully automated tracking of user interactions, it remains challenging to make sense of the resulting, often large, provenance graphs.
Recently Kery et al. [164] described how analysts rarely use conventional version control, such as Git, thus motivating their design of a lightweight inline versioning method for programming environments to support EDA. In our work, we also employ provenance tracking of the entire analysis pipeline. However, the tracking is triggered by user annotations to support lightweight versioning, and to avoid cluttering the provenance history.

12.2.2 Data-Driven Storytelling

Data analysis is arguably meaningless if its findings cannot be effectively communicated to stakeholders, other analysts, or the general population [294]. One way to convey data analysis results is through data-driven storytelling [189], where annotations, rich media (videos, images, sound, tables, etc), and visualizations are combined into a narrative [103, 141, 274]. The increased focus on creating interactive presentations to communicate findings has led to several related concepts. Examples include active reading with live documents [307], literate hierarchies using supplemental materials [114], interactive narratives [68], and external representations [167, 258]. However, because design choices for visual communication impact comprehension [142, 167], there often is no single presentation that fits all audiences. For example, a C-level executive may only need the top three takeaways from a quarterly sales report, whereas a strategic account manager will want to understand the provenance and details of the same analysis.

Current tool support for data-driven storytelling mostly consists of linearizing analytical findings into a sequence [268], which requires the designer to fixate on a specific abstraction level in their narrative. This discards the richness of non-linear analyses and makes it difficult to review the findings at the desired level of detail. Separating analysis from presentation also fragments the sensemaking process further, often making the communication of analytic products an afterthought instead of a mechanism to support collaboration throughout the exploration [165, 262, 301]. For this reason, Gratzl et al. [113] created a method for seamlessly switching between visual exploration, authoring, and presentation specifically for interactive visualizations using provenance tracking of user interaction. In our work, we extend previous work by focusing on the entire analysis pipeline, i.e., the provenance of cleaning and modelling parts as well as user interaction in visualization parts. Furthermore, our hierarchical insight structure allow users to build interactive reports targeting multiple user types while still preserving the provenance in full detail.

12.2.3 Notebooks and Toolkits

In recent years, literate computing in the form of computational notebooks [108, 170, 231, 250, 289] have become an essential part of data science [163, 262], even of entire organizations [300]. Notebooks combine executable code, their output, and text in a single document, which has proven to be very useful for quick pro-
totyping and exploration. Thus, notebooks already possess qualities suitable for data-driven communication and replicability. For these reasons, recent efforts have tried to enhance their functionality; for example, synchronous collaboration in notebooks is becoming the de facto standard [108, 250], and reactive execution flows are also being supported [19, 231]. Nevertheless, the linear document nature that is fundamental to all computational notebooks yields a tension between exploration and presentation [158, 262], making it difficult to support non-linear workflows that alternate between generating and presenting insights.

Beyond notebooks, several toolkits exist for quickly creating visualizations for exploratory data analysis [213, 269, 270, 325]. However, these systems generally do not provide support for insight management during EDA. A prominent example is the Tableau [290] suite that takes analysts from data preparation, to visual exploration, and finally to presentation. However, narration is typically a separate step in current tools, or done in completely different tools [23, 230]. In our work, we build on the success of collaborative notebooks and pipeline-based methods for data analysis by implementing our system on top of Codestrates [250] and Vistrates [19] (cf. Section 12.4.4).

12.3 Design Framework: Data-Driven Reporting

The sensemaking loop [241] is an iterative process for foraging data, building schemas from specific findings, and creating presentations from the schemas. In this way, the analyst engages in a gradual, bottom-up, and data-driven refinement that slowly increases in abstraction level. The whole process is iterative, meandering, and prone to branching [48, 279]. Sometimes, the analyst will backtrack to explore a different avenue of inquiry; sometimes entire paths will be abandoned; and sometimes the current strand of investigation proves to be the right one. Supporting this type of workflow is a challenge in itself, and in collaborative scenarios it is also a challenge to share, hand off, and communicate insights from such a process, as amply pointed out in past work [128, 159, 203, 262, 276, 331]. In addition, there is no watertight boundary between analysis and presentation [241]; sometimes a presentation may even return back to exploration. Thus, as described already in the initial research agenda for visual analytics [294], all this indicates an explicit need for integrating presentation in the analytic process.

For example, consider a team of analysts tasked with finding corrupt companies in financial data. In order to identify potential violators, they may analyze known cases and identify typical conducts, e.g., specific registration changes, financial statements, or shareholder changes. Within the team, the members have varying competences; some are expert programmers, some have knowledge of data science methods, and others possess domain knowledge. Sharing data work and insights in this environment is a challenge as highlighted above, and especially when sharing partial progress. In addition, the team occasionally needs to consult with their stakeholders to discuss next steps as well as share their work with
other teams within the organization to support reuse and to align knowledge about risk patterns. This scenario is based on data from the Danish Business Authorities, indicating that blending analysis and reporting is an intrinsic challenge within larger organizations (we expand on this scenario in Section 12.5.2). In this section, we first explore core challenges for integrating data-driven reporting and then describe design concepts that address them.

12.3.1 Challenges for Integrating Data-Driven Reporting

Through an analysis of related work and systems, we have identified three core challenges for integrating data-driven reporting:

C1 – Annotations & External Representations. Insights are pieces of information relevant to the analysis that can be characterized in manifold ways—complex, deep, qualitative, or even unexpected—and refer to different artifacts, e.g., single data points, patterns, or the overall analysis structure. Capturing and presenting such insights is a challenge and often require rich mechanisms for creating interactive narratives, annotating visualization states, and constructing external representations. Still, few tools incorporate such methods as part of the analysis environment. For example, in notebooks, it is mainly the layout that specifies the relation between annotations and computations, which means that describing the iterative nature of an analysis is challenging. This challenge therefore includes maintaining exactly what parts of the analysis annotations refer to and simultaneously supporting interactive representations.

C2 – Adaptive Details. Expertise and level of interest vary in collaborative workflows, which means that every detail is not always beneficial, but should be readily available for review when required. This challenge has been highlighted in various ways in related work. It has been described how preparing a notebook for presentation often requires the analyst to delete parts—parts that may be of benefit at a later stage. Rule et al. have subsequently motivated how dynamic restructuring of notebooks can support sharing. Research has also shown how hierarchical structuring of supplemental materials can benefit comprehension and how analysis history can support collaboration and task handoff. Thus, it is not only a challenge to dynamically maintain details of an analysis, but also in a way that indicates how insights are related. Current tools provide little support for such multi-level and broad-audience analysis products that can capture the full richness of the data analysis process, including not just findings, but also dead ends and failed hypotheses.

C3 – Fragmented Workflows. An overarching challenge is the non-linear nature of collaborative data analysis. An analysis frequently alternates between exploration and reporting, with the same analysis parts often
revisited several times. However, in most current visual analytics tools—with a few exceptions [113]—analysis and presentation are separate stages that are poorly integrated [105, 202]. By forcing the user to create presentations as a separate activity, the rich provenance of the analysis is easily lost and it becomes non-trivial to go back to the analysis stage or reuse analysis parts.

12.3.2 Linking Annotations to Analysis States

To address C1, we adopt the concept of linking annotations to visualization states [131, 302], but we expand it to incorporate the full provenance of the analysis pipeline. Conceptually, an annotation can be linked to a snapshot of the analysis parts it describes (Figure 12.2a). For example, the states can capture selections in visualizations, input values of UI elements, or computational output. The subset of the analysis which an annotation depends on is defined by how various parts are linked. For example, annotating the first cleaning step does not depend on a later visualization, but annotating the visualization depends on the first cleaning step. While the provenance information is automatically maintained, the user defines which states should be captured by triggering the tracking through annotating insights. As this force the user to actively create annotations throughout an analysis, it addresses the issue of making sense of a potentially inflated provenance graph after EDA.

This concept also addresses C3 in that it is possible to return to old stages of an analysis and use the annotations to recall any reasoning. However, changing the analysis state during EDA can have the side effect of negating certain annotations as they no longer describe the active analysis states. We therefore introduce the concept of active and inactive annotations. If the current states matches the linked states of an annotation, it is active; otherwise it is inactive. Analogously, manually activating the annotation restores the snapshot of the analysis system. This enables bi-directional interaction for exploring annotations, contrary to the one-directional interaction in, e.g., Idyll [68]. But, it also introduces design challenges for adequately indicating when annotations become active or inactive. Finally, annotations can be grouped based on whether they describe, and thus depend, on the same analysis parts (represented as stacks in Figure 12.2a). If each annotation in a stack describes a different state, only one annotation can be active.

12.3.3 Dynamic Insight Hierarchies

For tackling challenge C2 and supporting the flexible nature of the sensemaking loop, we propose to maintain all annotations in a hierarchical structure. This is motivated by the observation that people tend to organize information in hierarchies, in order to break down ideas and make them easier to capture [218]. The natural alternative to a hierarchical information structure is a graph structure. However, it has been shown that hierarchical structures can better support
Figure 12.2: Bridging narration and analysis: (a) the dynamic annotation hierarchy ties low-level insights to analysis states and links them to higher-order schemas; further, (b) the annotations can be linked to specific presentation views.

overview and comprehension, especially for readers with low prior knowledge of the content [5, 260]. In contrast, graph structures are better for certain information seeking tasks for readers with higher prior knowledge, but this structure imposes a more demanding process on the reader [5, 260]. To build up such a hierarchy, analysts can capture an annotation in a cell (as in computational note-books). Similar to the concept of structured writing [136], these annotation cells allow the analyst to organize the analysis details and abstract multiple specific insights into generalized structures that we call dynamic insight hierarchies.

As annotations can be grouped based on the same analysis parts they describe, related annotation cells can automatically be kept as groups within the hierarchy. The active annotation cell can then be shown as the top (visible) cell in a group and the inactive annotations can be thought of as tabs with alternative explanations. By supporting this as the default behavior, users are assisted in adding new annotations to relevant places of the hierarchy. This way, the annotations can capture the alternative decisions that were considered (and thus which ones were omitted), kept in a semantically sound location of the hierarchy. For example, there is typically a specific model or visualization selection that proves to be the ideal choice for a given analysis. However, this behavior does not fit all analysis types, and if multiple annotations about the same visualization state are created there is more than one active annotation. Thus, it should be possible to split up annotation groups to have annotations about the same analysis parts in different places of the hierarchy. Note, all annotation cells in a hierarchy does not have to depend on analysis states. It should also be possible to create normal cells as in the literate computing paradigm [170] within the hierarchy that e.g. describe the logic behind an algorithm.

The insight hierarchies can be constructed during the analysis process and at
any point be restructured, thus supporting challenge C3. Conceptually speaking, annotation cells can either be *merged* by introducing a parent cell, which summarizes the high-level meaning or interpretation of the underlying cells, or *split* by adding children to an existing cell, and thereby subdividing the parent into multiple supporting pieces of evidence (Figure 12.2). Supporting both types of operations interchangeable is critical to support the non-linear nature of EDA. By structuring insights during the analysis, the author is already creating a narrative similar to data-driven storytelling [103, 141, 274], i.e. a guided way of understanding the current content. In addition, the dynamic insight hierarchy supports precisely the adaptive level-of-detail outlined in challenge C2. A viewer presented with an insight hierarchy will initially only see the top-level element (or potentially only a few levels down). Depending on their expertise, available time, and interest, the viewer can then expand annotations with children as far as desired, potentially down to the atomic annotations at the bottom of the hierarchy. Readers can also choose to stay on a single level-of-detail and thus use the report as a linear story.

12.3.4 Hierarchical Presentation Views

To complement the annotations and address C1, we propose the concept of *presentation views*. By default, annotation cells remember the analysis view during their creation, allowing to restore it later along with the analysis state. Presentation views are an addition that allow to define a different representation. For example, a user can manually create a canvas containing visualizations alongside with hand-drawn elements. This flexible creation of external representations supports the comprehension of insights in the same way visual elements are used in data-driven storytelling [189]. Similar to linking annotations to analysis states, annotation cells can be linked to presentation views (Figure 12.2b). When browsing the annotation cells, the linked view is automatically shown to the user and the analysis state restored. By supporting easy creation of such views throughout the analysis process, the gap between the analysis and the presentation (C3) is effectively minimized.

Furthermore, traditional presentation tools such as Microsoft PowerPoint, or even Tableau’s story points, force the analyst to linearize a rich and branching analysis process into a flat sequence. Linking the information cells of the dynamic insight hierarchy to presentation views effectually creates *hierarchical presentations*, further supporting C2. In our concept, it is simply the insight hierarchy that defines the hierarchy of the report. Instead of merely being able to go backwards and forwards in the presentation sequence, our hierarchical approach also provides “up” and “down” operations that will *roll-up* and *drill-down* into the hierarchy, respectively (Figure 12.3). This way, the reader can lower or raise the abstraction level as needed. To support interactive reading, presentation views remain interactive such that readers can explore the visualizations, and the annotation cells will then adapt to user interaction by changing to *active* or *inactive*. Hierarchi-
cal presentations can be explored directly in the workspace interface where the insight hierarchy is constructed, but it is also possible to create dedicated presentation interfaces that support different types of review experiences and controls (Figure 12.3).

Figure 12.3: The hierarchical presentation concept. Navigating the report entails not just visiting annotations on one level, but also ascending and descending the hierarchy.

12.4 The InsideInsights System

The InsideInsights system is a proof-of-concept of the design described above, implemented as a collaborative web-based solution (the source code will be made available online at the time of publication). The system is built on top of the computational notebook Codestrates [250] and the Vistrates component model [19]. The novel combination of literate computing, hierarchical structure, provenance tracking, and presentation techniques allows for an iterative workflow interleaving composing analysis parts, capturing insights, and sharing the current progress. The workflow is therefore similar to the open-ended workflow of computational notebooks. In this section, we will explain the system with its three main parts: (A) compose analysis, (B) develop insight hierarchy, and (C) share and review.

12.4.1 Part A: Compose Analysis

The InsideInsights system supports a visual data-flow approach, where a user performs sensemaking by assembling components to transform and visualize the data on an interactive canvas (Figure 12.1c). For this purpose, Vistrates [19] provides a component template and a reactive data-flow based execution model within a notebook environment. A component in Vistrates is essentially a piece of code that consumes one or multiple input sources, process the data (e.g., to create a view), and eventually provide an output. The component observes the data output of its sources and reacts on any updates, causing a recalculation of any views, data, or filters. A wide range of existing components (e.g., aggregation methods, visualizations, input controls) can be instantiated from a global repository, or new
ones programmed and shared, either by inheriting from a prototype or written from scratch. Instantiating a component will insert the prototype code in the notebook. This way, components can be edited or reconfigured in the notebook anytime. The configuration of a component contains input mappings as well as the current state of the component.

The component model allows for the interactive pipeline abstraction in our system (Figure 12.1) showing the data flow between components. The pipeline supports both component configuration and inspection of views (cf. Figure 12.9). With InsideInsights, we also extend the core functionality of Vistrates by introducing composite components, which allow users to build hierarchical pipeline parts that can be collapsed or open as seen in Figure 12.4. Besides helping to structure the pipeline in a meaningful way, these composite components also allow users to share and instantiate a set of pre-configured components. We will return to the implementation of composite components in section 12.4.4. To ease the understanding of the pipeline, we also automatically distinguish between four high-level component types. Dark components are data sources, which include both data as well as parameter inputs. Green components are visualizations, and light components are computations with no view, such as a filter or a $k$-means clustering algorithm. The unfilled components are composites. A key strength of our system is the ability to seamlessly switch between working in the InsideInsights interface and developing components with the underlying notebook interface. It is the individual components that decide what is piped downstream, but a typical visualization component will output the current selection and store this as its state. The generality of our approach is inherited from the generality of the underlying component model. In essence, there is no restriction on what a component can do as long as it adheres to the general template. Still, the integrated data-driven reporting design can also be implemented with other analysis systems.

![Diagram of pipeline components](image)

Figure 12.4: Part of a pipeline showing a clustering component that takes as input a parameter $K$. The resulting centroids are visualized in Centroid Vis. The components have been grouped into two (currently open) composites.

At any point when constructing the pipeline, the analyst can capture important insights in the data by creating an annotation cell to describe specific component. The full provenance of the analysis, i.e., the states of the previ-
ous components in the pipeline, is automatically maintained in the background. We describe this in detail in Section 12.4.4. Annotation cells can contain any HTML content, similar to text cells in, e.g., Observable [231]. As mentioned in Section 12.3.3, annotations about the same component are by default grouped together in the hierarchy, and the different annotations are then represented as dots (see Figure 12.5). Orange dots are active annotations and grey are inactive. Unless the annotations describe the exact same set of states (cf. subsection 12.4.4), usually only one annotation will be active at the time, and thus be the visible one. Otherwise, the latest active annotation will be visible as default, and the user can browse both active and inactive annotations. When the component states change, the active and inactive annotations will update accordingly, surfacing the current active annotation. While it is not possible to split up these groups in the current prototype, it is trivial to implement. Alternatively, annotation cells without links to any component states can also be created in the hierarchy.

Figure 12.5: Creating an annotation cell linked to a component state can be done simply using the pen icon on the specific component. Whenever the state of the component changes active (orange dots) and inactive (grey dots) annotations update accordingly.

12.4.2 Part B: Develop Insight Hierarchy

The dynamic insight hierarchy is always visible, even during analysis, on the left side of the screen in our prototype (Figure 12.1a). This hierarchy contains all annotation cells, both those that depend on the analysis state (cells with dots above) and cells with general descriptions. Composing a data-driven report in InsideInsights can be performed in both top-down and bottom-up fashion (Figure 12.6). For top-down, the user may create annotation cells with merely textual or rich media content that are not yet linked to any analysis state or presentation view. In the confirmatory fashion of a top-down approach, such cells would describe high-level hypotheses about the data. A series of conceptual split operations would populate these hypotheses with annotation cells as children, and this process would be repeated until the analyst can anchor each claim in actual data. For a bottom-up, exploratory approach, the analyst would instead let the data itself guide them, surfacing specific low-level insights in the dataset and then letting these be gradually aggregated into higher-level findings using annotation cells in conceptual merge operations. Specifically, the hierarchy can be extended
Figure 12.6: Illustration of bottom-up (a) vs. top-down (b) analysis workflows as supported by InsideInsights. In (a) multiple annotations are summarized in a new parent, and in (b) a high-level goal is further specified with a new child.

by selecting an existing cell and append a child (Figure 12.6b), or by selecting multiple cells and append a parent (Figure 12.6a). Alternatively, the pen icon on a component is a shortcut for creating an annotation cell that is linked to that component (cf. Figure 12.5). We expect that a practical analysis and authoring session in InsideInsights will alternate between both of these modes. Thus, an existing cell can at any point be selected and moved around in the hierarchy.

Analysts can also link a presentation view to an annotation cell. We currently support two view types: (1) the analysis pipeline itself, configured with certain composites open or closed, or (2) a canvas with selected component views, such as a single full-screen visualization or a dashboard of multiple visualizations, rich media and hand-drawn elements (Figure 12.7). When an annotation cell has a linked presentation view, that view will be shown instead of the full pipeline when reviewing the cell later on (see Part C). This allows the analyst to author external representations that show specific items of interest and hide the complexity of the full pipeline (Figure 12.7). To attach a specific cell to a presentation view, users will first create an empty slide (or canvas) and populate it with items from the pipeline as well as rich media (text, images, video, etc). They can then use the link icon next to the selected cell to connect the presentation view to the cell.

12.4.3 Part C: Share and Review

Sharing and reviewing a report in InsideInsights is not a separate mode; at any point during the exploration process, a user can traverse the growing insight hierarchy, either by clicking on cells or by using the arrow keys. Sharing an analysis is simply done by sharing a link to the web-based document—multiple
12.4. THE INSIDEINSIGHTS SYSTEM

Company Event Analysis

The goal of this analysis is to identify companies for further inspection based on whether they have behaved similar to companies that was forced to close.

Based on the underlying analysis, the listed companies are worth a further inspection. Click to investigate a particular one.

This analysis consists of 3 parts:

1. The first part is to define the outcome of a company, pinpointing the time of occurrence suitable for this analysis and filtering the data. The definition is based on the company status event updates.

2. The second part is to generate a model that can effectively summarize the event sequence data in a reusable way.

3. The last part is to use a visualization of the aggregated sequences to first reason about what patterns might lead to a negative outcome and afterwards inspect the remaining alive companies.

Figure 12.7: A presentation slide with two freely arranged views; an event flow visualization and a prediction accuracy view. The presentation slide is linked to the bottom narration cell as indicated by the link icon next to the cell.

Collaborators can even modify the same document simultaneously. Having said that, InsideInsights also supports a dedicated presentation mode, where the report can be navigated similar to a traditional slideshow and where edit controls are removed. The reviewer can choose to stay on a single level in the hierarchy, and thus read a linear story, but a reviewer can also choose to traverse the hierarchy to learn additional details or abstractions. As mentioned previously, InsideInsights supports bi-directional interaction [Figure 12.5]. Activating an annotation can be done by clicking the appropriate dot, which will restore the analysis state of that annotation. Vice versa, modifying the analysis state (e.g., interacting with a visualization) will update active and inactive annotations, and surface the active ones. This way, the visible narrative always match the current analysis state. Thus, if a reviewer modifies the analysis such that it violates a given reasoning, the insight hierarchy will reflect this by making the involved annotations inactive.

12.4.4 Implementation Details

The InsideInsights system is implemented with standard modern web technologies (JavaScript, HTML, CSS, etc.) as a top layer on the existing technology stack consisting of Webstrates [169], Codestrates [35, 250], and Vistrates [19]. Codestrates is a JavaScript-based computational notebook that provides collaborative editing of code through the DOM synchronization mechanism of Webstrates. On top of this, Vistrates provides a general component model, a data-flow based execution
model, and view abstractions such as the pipeline and visualization dashboards. In theory, any computational or visualization component can be programmed using Vistrates. The InsideInsights system extends the existing technologies with composite components, annotation-driven provenance tracking, and interactive hierarchical reports.

Composites components are implemented with the classic composite software pattern [4] applied to the existing component model. Thus, a composite component is a component itself that contains a list of child components and specifications of their configurations. This way, a composite component can be instantiated as any other component; which also instantiates child components with the specified configurations. Already instantiated components can be combined to form a new composite by the user. Subsequently, the user can share the new type of composite through the component repository.

As components store their configuration and state encoded as JSON [19], the provenance tracking in InsideInsights is implemented using these specifications. The analysis state of component X is therefore defined by the state specification of X itself along with the states of the previous components in the pipeline that X depend on. This way, the pipeline defines a state dependency graph which is automatically maintained by the system. But, as mentioned previously, the provenance tracking in InsideInsights is triggered by user annotations. Thus, not all state changes will result in an expansion of the state dependency graph; whenever a user links an annotation to a component, only the relevant states for that insight are appended to the graph. Restoring an analysis state then simply consists of traversing the state dependency graph and update the relevant components. The insight management and provenance tracking—forming the InsideInsights system—is implemented as a meta-package for Vistrates. Along with the state dependency graph, this package maintains annotation and state links along with mappings between text cells in the underlying notebook and the annotation cells of the InsideInsights system. This way, there is a one-to-one mapping between text cells in the two interfaces.

12.5 Usage Scenarios

There are many possible applications for a data-driven reporting system such as InsideInsights. To showcase the breadth of our system, we provide two conceptually different application examples:

- **Crime data analysis**: Baltimore crimes data analysis, where findings inform the next steps in the analysis, yielding a shareable data-driven report. This conceptual scenario illustrates how a basic workflow can unfold.

- **Company event analysis**: The development of a method for finding companies that may face imminent bankruptcy. The method is based on prior
development of new analysis methods in collaboration with the Danish Business Authorities (DBA) [209]. This scenario is in part informed by the existing collaborative practices at DBA, which involve people with varying expertise, and in part by how InsideInsights can foster new collaborative work practices.

12.5.1 Baltimore Crime Peaks

Let us follow a fictional crime analyst (John) who has been tasked with analyzing crime peaks in the city of Baltimore, MD. John loads a dataset of 110,074 crimes (data.baltimorecity.gov) from the period of 2012 to 2015 into the InsideInsights system with a CSV component. He generates a line chart of crimes over time by first instantiating an existing composite component from the repository that consists of three internal components; time-based aggregation, a line chart visualization and a data filter. He then configures the composite component in the pipeline by selecting the CSV component as the data source and by specifying property mappings, i.e., the names of the time and crime count variables. Upon completion, he identifies two large peaks warranting further investigation. He therefore creates two state annotations on the line chart using the pen icon, one for each crime peak. This way he can focus his analysis on one peak without having to remember the other.

The first crime peak occurs on April 27, 2015. John creates a bar chart of the crime type distribution for the selected date, which shows large peaks for aggravated assaults and burglaries. He quickly annotates both selections. To confirm that the peak is not only caused by one of the individual crime types, he creates a line chart of crimes over time separated by crime type. Then he merges the state annotations for the crime type visualizations into a high-level annotation that describes the intuition and purpose behind this additional comparison. He also links the high-level annotation to a presentation view with the two crime type visualizations that support this comparison, by organizing the views on a canvas and pressing the link icon next to the annotation cell.

John now creates a map visualization where the selected crimes are shown. Interestingly, almost all aggravated assaults occur at the same location, while burglaries are scattered throughout town. John finds this pattern curious and annotates both states of the map. Since the states of the map are linked to previous components, it is easy for John to retrace how he arrived at these findings and understand exactly how the visible crimes have been selected. As his analysis is set to April 27, John googles this date and finds multiple articles about the funeral of Freddie Gray, a young African-American man killed by Baltimore police. John then notes that the location of the aggravated assaults coincides with the protests that followed the funeral. Therefore, he merges the underlying annotations into a high-level interpretation, noting that the increase in burglaries was likely caused by looters taking advantage of the confusion of the riots. The resulting analysis is shown in Figure 12.8.
12.5.2 Danish Company Event Analysis

Elina is a fictional analyst at the DBA. The DBA maintains information about Danish companies, performing analyses to aid political decisions and to catch non-compliant behavior. Elina has been tasked with finding early indicators of involuntary closure. She starts by loading a dataset of 10,005 events into InsideInsights. Her goal is to develop an analysis method that subsequently can be used on other subsets of the database that consist of more than 50 million events (datacvr.virk.dk/data). Events include business type updates, board member changes, accountant replacements, etc.

Elina quickly realizes that given the status registration event in the data, it is not straightforward to define what qualifies as an involuntary closure. Using a bar chart of the different status updates and their frequency, she creates an initial definition. She adds annotations to the visualization selections motivating why these have been chosen. Further, she merges these into annotations describing how the definition is currently calculated. Elina then links the high-level annotations to presentation views with the appropriate visualizations. To check the validity of this definition, Elina shares her current work with a domain expert on the registration data. The expert browses the insight hierarchy aided by the presentations views to understand the current definition. To improve the definition, he modifies the visualization selections and adjusts the annotations to also include companies that have closed due to other involuntary closure types than just bankruptcy. Further, he selects the date of the outcome to be the first time

![Crime Peaks](image)

**Crime Peaks**

The main chart shows that burglars occasionally take advantage of chance situations elsewhere in the city. Likely because they know this will draw much attention from the police.

Occasionally, there is a strong correlation between the crime peak and the occurrence of an event that required a lot of police attention.

The crime location map is one way to identify this.

The data is loaded in the first component named Baltimore Crimes.

The first part of the analysis is to identify crime peaks. Crime Over Time (the yellow peaks) indicates peaks and selects the appropriate time period for further analysis.

Next step is to identify what types of crimes occurred in the selected time period using the Crime Type Distribution chart. In this step, computers of interest can be selected for further investigation.

Burglaries is the most frequent crime type in the selected time period.

It is always ideal to confirm that the peaks in individual crime types are in fact also out of the ordinary in general. This can be confirmed using the Daily Crime Avg chart, that shows the average number of crimes per day of the particular type.

The location of the selected crimes might even win the peak in crime occurred. The locations of the selected crimes can be viewed in the Crime Location chart.

The burglaries are scattered throughout the city.

Figure 12.8: Crime peaks analysis document after John’s initial analysis session.
the companies were warned by the DBA. Since both the calculation and motivation have been captured in the InsideInsights system, future investigations into troublesome companies can now easily be performed with the same definition.

Elina continues her analysis by envisioning the computational parts she needs: (1) outcome definition and event filtering, (2) event aggregation, and (3) event hierarchy and prediction potential visualizations. She creates high-level annotations describing her envisioned analysis, and then repeatedly splits this description into multiple children that eventually can be realized in the pipeline. After composing the pipeline for the second and third part of the analysis, Elina realizes that the initially envisioned hierarchical aggregation of the raw event sequences is insufficient. The event sequences vary too much to be efficiently aggregated. Elina therefore develops another version; however, she keeps the first attempt as a subtree to motivate why the new approach was needed. As the new approach requires a component not already in the repository, she uses her current report to coordinate with an internal programmer. The programmer then implements components that cluster the sequences and computes high-level events. Elina now uses these components in her pipeline prior to the hierarchical aggregation. She stores her entire exploration using state annotations that she links to each part of the planned analysis. The resulting event flow visualization of the new model can be seen in Figure 12.9.

One year later, the DBA has started to collect additional registration events, and Jenna is accordingly tasked with revising the analysis. She goes through Elina’s annotations, motivations, and descriptions to gain an understanding of
the current method. She quickly identifies where to modify the analysis and begins her work. Because the narrative is linked to the program, she can easily update old motivations that no longer hold true for the updated analysis.

12.6 Expert Review

We conducted an expert review of the InsideInsights system with two experienced visualization researchers: an associate professor of computer science with 15 years of experience (P1), and a visual analytics specialist employed in industry for the past 3 years following his Ph.D. (P2). Our goal was to gather early feedback and insights on the potential use of the integrated data-driven reporting functionalities. We chose an expert review because qualitative methods are generally better suited to capturing the form of high-level sensemaking tasks typically performed using exploratory data analysis [54, 242], and expert reviews in particular have been shown to be effective for visualization evaluation [297].

12.6.1 Method

Each session was structured in two phases. In Phase 1, participants were given the full report of the company event analysis scenario, and in Phase 2, they were given only the analysis pipeline of the crime peaks scenario without annotations. For the full report (Phase 1), the experts were tasked with explaining the analysis and the interplay between the involved components, modify parameters to generate a better model, and subsequently summarize what the most important events were. For the analysis pipeline in Phase 2, the experts were tasked with exploring the data and building a narration hierarchy of their insights from scratch. Prior to beginning, participants were given an introduction to the purpose of our research and a tour of the main features in our prototype. During the phases, the participants were instructed to follow a think-aloud protocol. Sessions lasted around an hour and were screen-captured and voice-recorded. Also, observation notes were collected.

12.6.2 Expert Feedback & Findings

The tasks were designed to be open-ended since we wanted the participants to explore all parts of the system and get their feedback on its usefulness. Both participants were able to solve the tasks; however, the open-ended nature of the evaluation resulted in them employing different approaches in Phase 1. P1 mainly followed the narrative and observed the attached views and how the pipeline unfolded, while P2 instead investigated the pipeline and consulted the narrative only when something was unclear.

In general, both participants were engaged by the dynamic workflow of the prototype, and they both felt that there was “great potential” in the concept. P2 explicitly mentioned that they would like to spend more time using the prototype.
The participant specifically emphasized that gradually diving into further detail was a nice way to understand an analysis, either by going down the narration hierarchy or by browsing the pipeline hierarchy. Both participants pronounced the close link between narrative and analysis useful.

P1 noted that in our prototype, it is hard to navigate both hierarchies (narration and pipeline) at once since they do not necessarily align. While this provides freedom to develop many complex types of data reports, it is “freedom under responsibility,” as P2 said. This complexity also manifested itself when the participants found a lack of additional explanations in certain parts of the analysis in Phase 1. In addition, P1 found it confusing when the narration changed perspective between method annotations (how) and analysis result annotations (what). The participant suggested to explicitly provide how, what, and why annotations—similar to the structured tagging in CommentSpace [312]—and allow the user to explicitly choose a certain perspective to read, or to have different narratives for different audiences. Having explicit annotation types could also support authors in developing the narration, as it would highlight what has already been described and what is missing.

Overall, the participants also noted that the current InsideInsights prototype has usability limitations. Some interactions still have latency due to the size of the datasets and the complexity of the pipeline. However, these issues can be resolved by optimizing the underlying analysis system, as well as by offloading computation onto the server. Other usability issues arise when the analytical components in InsideInsights did not offer functionality desired by the participants. P1 also suggested providing a more restricted (or guided) interaction to assist novices. Once more experience is gained, novices can then move on to the full system.

12.7 Discussion & Future Work

We believe that our proposed system is a first step to better support existing analysis and reporting workflows. In the following, we will discuss current limitations of our implementation as well as multiple interesting avenues of future research.

12.7.1 Limitations

While the contribution of our work resides within data-driven reporting, our work requires the scaffolding of an analytical system to provide the necessary functionality. We chose to build on the existing Vistrates [19] and Codestrates [250] frameworks for this purpose, but our efforts understandably still fall short in certain aspects compared to mature data analysis systems such as Tableau. Specifically, user-friendliness of the analysis system was touched upon by the expert reviewers (Section 12.6). This is a limitation that could be remedied by integrating our data-driven reporting method in other analytical environments in the future. In addition, InsideInsights provides visual feedback about how an annotation connects to the pipeline, but the visual feedback in the reverse direction can be
improved. While annotations update when the user interacts with pipeline components, this can happen out of sight of the user when the narration hierarchy becomes sufficiently large. To overcome this issue, improved visual indicators for occurring changes could make such connections apparent.

Another challenge is how to help non-experts navigate the dynamic insight hierarchy. We currently represent the insight hierarchy as indented text cells, but this design may have certain limitations with respect to scalability. Although collapsing branches of the hierarchy allow users to keep cells of interest within view, indenting cells exceedingly have visual limitations. Thus, exploring alternative cell layouts or interaction methods becomes important. On a conceptual level, the grouping of related annotations supports the user in keeping the hierarchy of a manageable size, but keeping entirely different analysis paths as subbranches may make the hierarchy too deep. Supporting the user to maintain different variations of a story may be a way to address this limitation, e.g., by having several hierarchies or utilize different cell types (cf. Section 12.6.2). In other words, our novel integration of data-driven reporting is also yielding interesting design challenges for future work.

Finally, our experts also noted that the freedom afforded by InsideInsights can be a burden to the analyst, and that some structured guidance would be helpful. Supporting insight generation [75, 317] and automatically inferring visual analytic activities from user interactions [110] are already active research focuses. Combining these methods with support for generating comprehensive annotations could be interesting.

12.7.2 Implications for Visual Analytics

Recent developments on interactive notebooks [277] have set a trend towards reproducible and shareable data analytics. InsideInsights continues this trend by allowing the creation of data-driven reports that can be accessed by many different users, thereby making data analytics accessible to a wider audience. The interaction methodology we propose can essentially change how analysts and stakeholders collaborate by supporting the creation of common ground, which is a vital part of collaborative data analysis [128].

While our current prototype is built on top of a collaborative notebook [250] and a visual analytics component model [19], the design itself is independent of any data analysis system that exposes its internal state. This can be achieved in at least two general ways: either by directly supporting provenance tracking through an API to navigate and reactivate states, or by exposing declarative specifications for the internal state, such as as in Vega [269] and Vega-lite [270]. The latter is exactly the type of system our current prototype is built upon. By exposing their internal state in this way, future data analytics applications can more easily become part of larger data analysis ecologies.
12.7.3 Towards Literate Analytics

The key contribution of InsideInsights is to bridge data-driven narration, provenance, and collaboration to assist users in organizing and understanding findings at variable levels of detail. The long-term goal behind our work is to empower users with the ability to dynamically structure their analysis to promote comprehension of increasingly complex data and algorithms. Similar to how the motivation behind literate programming was to write programs not only for the sake of the computer but also to promote human understanding [171], there is an increasing focus to promote human understanding of data analytics [68, 173, 307]. Accordingly, we think the method presented in this paper is a step on the way to a new paradigm we tentatively call literate analytics.

The descendants of literate programming [217, 321] are a testament to the unique way narration and annotation can support human comprehension. Literate computing [217] extended the literate programming concept by combining narrative with executable code. However, current literate computing solutions (e.g., notebooks [108, 170, 231, 250]) do not support the creation of hierarchical structures within the document—an important concept of literate programming. Our work combines hierarchies with interactive analytics within the same document.

The goal in our notion of literate analytics is to document the entire data analysis rationale, including insights about the method and the data as well as interpretations. While our method support the capturing of such documentation, our expert review revealed that it is also important to support completeness of the documentation. In general, we characterize literate analytics as integrated narration and analysis. This integration is manifested in the concept of live documentation, where annotations always match the state of the analysis they aim to describe, and thus enable interactive reading of data analytics. To support this, it becomes important that users understand how the documentation changes when interacting with the analysis, which is another aspect where our current approach can be extended. In addition, literate analytics aims to support documentation throughout the process such that the current progress at any time can be shared and replicated. Although this paper is only an initial step, we hope that our work will help spark future discussions about what literate analytics should incorporate.

12.8 Conclusion

We have proposed InsideInsights, an exploration of the design for integrated data-driven reporting, where insights are organized into an information hierarchy linked to analytic provenance and presentation views. Our system allows analysts to tag views, items, and entire states of a visualization and computational components with annotations as part of their analysis. We have also presented two scenarios and results from an expert review to demonstrate the validity of the idea. The
novel combination of literate computing, provenance tracking, and storytelling elements have the potential to bridge the gap between data analysis and reporting, thus pointing towards a new paradigm we tentatively call “literate analytics.”

Acknowledgments

This work was supported by the DABAI project (IFD-5153-00004B), the Stibo travel grant, and the DFG grant DA 1319/3-3.
**Vistribute: Distributing Interactive Visualizations in Dynamic Multi-Device Setups**

Tom Horak\textsuperscript{1}, Andreas Mathisen\textsuperscript{2}, Clemens N. Klokmose\textsuperscript{3}, Raimund Dachselt\textsuperscript{1}, & Niklas Elmqvist\textsuperscript{4}

\textsuperscript{1}Interactive Media Lab, Technische Universität Dresden, Germany
\textsuperscript{2}Department of Computer Science, Aarhus University, Denmark
\textsuperscript{3}Department of Digital Design and Information Studies, Aarhus University, Denmark
\textsuperscript{4}College of Information Studies, University of Maryland, College Park, USA

Figure 13.1: The Vistribute system: Based on a design space we derived six heuristics that can guide an automatic distribution of visualizations in changing device setups, e.g., (a) dual desktop, (b) laptop and large display, or (c) mobile device ensemble.

**Abstract**

We present *Vistribute*, a framework for the automatic distribution of visualizations and UI components across multiple heterogeneous devices. Our framework consists of three parts: (i) a design space considering properties and relationships of interactive visualizations, devices, and user preferences in multi-display environments; (ii) specific heuristics incorporating these dimensions for guiding the distribution for a given interface and device ensemble; and (iii) a web-based implementation instantiating these heuristics to automatically generate a distribution as well as providing interaction mechanisms for user-defined adaptations. In contrast to existing UI distribution systems, we are able to infer all required information by analyzing the visualizations and devices without relying on additional input provided by users or programmers. In a qualitative study, we let experts create their own distributions and rate both other manual distributions and our automatic ones. We found that all distributions provided comparable quality, hence validating our framework.
13.1 Introduction

Advances in mobile computing have spawned a ubiquity of networked digital devices in our everyday lives [25]. Such devices are increasingly liberating office workers from the bonds of their desks, allowing tasks to be distributed across the day, continued in different contexts with different device setups, and performed with an ever-changing constellation of participants [29, 143, 264]. Data analysis and sensemaking is no different, but current practice rarely exploits the full potential of cross-device interaction, instead merely using additional devices to increase screen real estate [3, 9] or improve visibility in multi-user settings [45]. Fully utilizing these ad-hoc multi-device environments would enable analysts to seamlessly continue their data exploration throughout an entire day across a plethora of devices, settings, and people [90]. For example, consider an oncologist in a hospital using patient tumor data to inform her practice (analyzing, e.g., tumor growth rate, blood levels). The doctor may spend some time on her morning commute to get up to speed (smartphone), in her office to plan treatment (desktop and tablet), continuing during a coffee break (laptop and phone) with a colleague spontaneously joining after a while (adding a tablet), then at a tumor board meeting with other doctors (large displays, laptops, and mobile devices), and finally in a treatment room consulting the patient (tablet and large TV)—all without spending time on manually setting up the interface. To our knowledge, no existing data analysis framework exists that is capable of dynamically and seamlessly adapting to such a multitude of devices, settings, and collaborators.

To address this gap, we propose the Vistrbute framework, an automatic approach for distributing visualization interfaces across dynamic multi-device environments based on view, data, and user properties. Unlike existing automatic distribution mechanisms for general user interfaces, such as AdaM [234], Vistrbute uses in-depth information about views, the data they visualize, and the tasks users want to perform on them to optimize the layout. The framework consists of a design space, a set of heuristics, and an example implementation. Our design space for cross-device visualization draws on the literature as well as an analysis of existing visualization interfaces, and explicitly considers dynamic factors such as view properties and relationships, device properties and the current device ensemble, as well as user preferences. Using this design space, we propose several heuristics as high-level constraints for distributing visualization views. Finally, our web-based implementation automatically collects information about the devices, the dataset, and the visualizations to derive a suitable distribution. In addition to the distribution itself, we enable users to adapt the interface distribution according to their needs and preferences.

In summary, our paper presents the conceptual Vistrbute framework with the following contributions: (1) a design space identifying important properties and

---

1Tumor board meetings convene doctors with different specialties to discuss cancer cases, share knowledge, and plan treatment.
relations for distributed visualization interfaces; (2) six heuristics guiding the distribution process; (3) a web-based implementation as one possible instance of the heuristics; and (4) a qualitative study where experts manually created distributions and rated both other manual distributions and our automatic distribution.

13.2 Related Work

Our work is located in the intersection of human-computer interaction (HCI) and visualization research, straddling two topics specifically: (i) multi-display setups in visual data analysis and (ii) general HCI research on view distribution.

13.2.1 Visual Data Analysis in Multi-Display Environments

In recent years, the visualization community has intensified their efforts in investigating analysis systems and visualizations that go “beyond the desktop” [187, 256]. While incorporating many different aspects (e.g., utilized modalities, display technologies, or interaction styles), using multiple displays in parallel is often one prominent characteristic of these systems. Often, mobile devices are used as movable containers for content, settings, or preferences [65, 135, 213]. This can allow for switching between working alone and in concert [213], or having user-specific tools available on hand [135, 303]. In this context, HCI research has also suggested general cross-device interaction techniques for data transfer [57, 135, 208, 280, 303].

Other literature focuses on specific device combinations. Spindler et al. [283] as well as Kister et al. [168] used hand-held displays in relation to a bigger context display (tabletop and display wall, respectively) in order to show details and alternative representations for a large visualization, while Langner et al. [183] used smartphones to enable co-located remote interaction inside a coordinated multiple views (CMV) application running on a display wall. With Thaddeus, Wozniak et al. [322] applied these principles to a smaller scale by using a spatially-aware smartphone for probing data on a tablet. In VisTiles, Langner et al. [182] focussed on physical ad-hoc layouts with mobile devices (one visualization per device) for data exploration.

While the presented concepts provide useful device combinations for specific visualizations or interaction styles, it remains unclear how these can be generalized for any situation and device constellation. However, as a study by Plank et al. [243] showed, this is a crucial aspect for multi-device setups: they found that participants did not take advantage of having multiple tablets during sensemaking tasks, even with optimized tasks. As their setup had multiple restrictions (e.g., one view per device, fixed device pairs), it appears that having a proactive and flexible interface is a gatekeeper for unleashing the full potential of multi-device setups.

A broader analysis was presented by Chung et al. [66], discussing multiple considerations for data analysis in multi-device setups. For instance, the com-
bination of displays can support different view arrangements (e.g., continuous, CMV, separated instances). Similarly, the way how updates and states are synchronized across devices can promote different interface functionality during the analysis. However, the authors do not touch on how to exactly realize a system supporting these different aspects, and only point out the technical challenges, e.g., system-imposed constraints coming from the device’s hardware or software platform.

13.2.2 Multi-Device Frameworks & View Distribution

Utilizing heterogeneous devices in parallel introduces multiple technical challenges how to coordinate and synchronize the devices. The HCI community has proposed several frameworks to tackle these challenges, often using web technologies as a foundation. Synchronization can happen on multiple levels: For instance, Webstrates [169] operates on the level of the Document Object Model (DOM), effectively maintaining exact copies on different devices. Other frameworks focus on the graphical aspects of an interface from a developer’s perspective, e.g., when spanning one canvas across multiple devices [249, 272]. Also, functionality supporting cross-device interaction techniques can be provided [138]. Designed specifically for visualization, Badam and Elmqvist [15] and Badam et al. [17] presented technical frameworks for synchronizing user interactions as well as application states across devices. While all these frameworks are designed to ease the development of new applications, they require that programmers or users manually arrange interface components.

This gap is partly addressed by research proposing specific distribution algorithms or frameworks, most of which automatically derive a candidate distribution based on interface semantics provided by the developer, and then let the user adjust the result. Panelrama [326] introduced a lightweight specification that allows programmers to provide additional semantics for HTML elements, which are then consumed by an interface optimizer. Park et al. [234] proposed an optimizer called AdaM, which is based on a constraint solver; however, AdaM requires users or developers to provide additional semantics for each interface component, too. The XDBrowser [224, 225] segments web pages and distributes the parts across devices. In all examples, the layout is not guaranteed to be optimal, and serves rather as a starting point.

More specialized applications may allow for automatically determining depen-
dependencies between interface components and how to organize them. As a case in point, recent work by Husmann et al. [143] presented a similar system in the context of an integrated development environment, but applied automatic assignments only for a few selected view constellations. To our knowledge, such an approach has not been proposed for visualization and data analysis yet.

13.3 Design Space: Interactive Visualizations in Multi-Device Environments

The distribution and layout of views in a visualization interface are not arbitrary, but often follow certain patterns. Based on related work, considerations of existing interfaces, and our own experience in cross-device research, we aim to provide a conceptual framework that is able to reproduce these patterns when distributing and arranging views across multiple devices. The framework consists of a design space, distribution heuristics, and a prototype implementation.

In creating our framework, we were guided by multiple considerations. First of all, individual visualizations encode richer semantics compared to other user interface components [246], such as the data being visualized, the visual representation chosen, and the typical tasks supported. By considering these aspects, it is possible to automatically derive properties required for a distribution that otherwise would have to be provided by analysts, designers, or developers.

Second, these semantics also reveal relationships between multiple visualizations [246], which allows for further refining the distribution. In existing interfaces or dashboards (see, e.g., Tableau dashboards\(^1\) or examples analyzed by Sarikaya et al. [263]) it is possible to observe such relationships, e.g., two bar charts are aligned for comparison. Similar aspects can be observed in research focusing explicitly on large displays or multi-device ensembles [182, 183], as well as for the involved devices, where their properties and relationships imply their strengths or possible roles in a distributed interface.

The design space aims to give an overview of interactive visualizations in multi-device setups, considering all relevant properties and relationships occurring (Figure 13.2), which we group and discuss as five dimensions in the following. At the end, this design space will eventually provide a fundamental understanding of the incorporated dimensions. By molding this knowledge into easy-to-apply heuristics, we aim to provide a guidance for new distribution approaches (i.e., specific implementations) for interactive visualizations.

13.3.1 Visualization Properties

In comparison to traditional UI components, visualization views feature a rich body of properties that depends on their configuration, visual representation, or encoded data. These properties can be used to construct visualizations (as in,
e.g., D3 [42], Idyll [68], Vega [269, 270]) as well as to analyze them (essentially the inverse of construction), as in our case.

First, visualizations can be characterized through properties related to their visual appearance: the actual visualization type (i.e., used visual marks), the applied encoding and mapping (i.e., visualized data dimensions), the axis configuration (e.g., orientation, scale, sorting), as well as the default size (and also implicitly the aspect ratio). Although these properties are often defined in the context of the considered data, they do not fully depend on the actual data: two views can have the same visual configuration but show disjoint data subsets. We also consider a visual density property, resulting from the mark size, potentially occurring overlaps, and existing additional elements (e.g., guides). This density can affect the comprehension and supported interaction; e.g., the selection of small marks is more difficult and requires a certain minimum precision (cf. Park et al. [234]).

For data-related properties, we consider the used data source, the data points themselves, as well as the internal state. The data source can describe only the source or the complete data flow prior to the view, i.e., from the dataset through filters or aggregation components. Depending on the visualization system, certain functionality (e.g., aggregation) can be part of the view (e.g., Vega-lite [270]) or a separate component (e.g., Vistrates [19]). Nevertheless, we consider them as pre-processing and not part of the visualization itself. The data points allow comparing the data of two views or analyzing the view regarding the number of visualized marks, e.g., to estimate how dense the visualization is. Finally, visualizations often maintain an internal state, e.g., selected marks or ranges, which can be accessed by other views.

13.3.2 Visualization Relationships

Typical visualization interfaces consist of multiple visualizations (often known as dashboards [265]) where the views complement each other by showing different aspects of the data and, in combination, help the user gain insights. We characterize the interplay between views as one of three relationship types: visual similarity, data similarity, and connectivity (Figure 13.3). These relationships yield patterns for grouping and aligning views common in existing interfaces [265].

Visual similarity considers how similar the two views appear, regardless of the actual encoded data points. We use the visual-appearance properties described above (e.g., type, encoding, axis configuration) to rate the consistency of two views [234]; by comparing the properties, the similarity can range from all different to all same. Similar views can support visual comparison when placed in juxtaposition [106, 107, 133, 240]. For instance, two views with the very same visual configuration is an example of small multiples, where the single instances differ only in the shown data. Slightly weaker relationships can be found in scatterplot matrices, where two plots differ in one dimension. In contrast, dashboards
### Visual Similarity

equality of visual properties

<table>
<thead>
<tr>
<th>Vis. Type</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>all same</td>
<td>all different</td>
</tr>
</tbody>
</table>

### Data Similarity

existence of data points in both subsets

<table>
<thead>
<tr>
<th>Source</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>distinct</td>
<td>overlap</td>
</tr>
</tbody>
</table>

### Connectivity

constellation / interplay defined by the data flow

<table>
<thead>
<tr>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
</tr>
</tbody>
</table>

#### Example: Small multiples

- all same

#### Example: Scatterplot matrix

- all same but one
- same
- bi-directional

#### Example: Dashboard

- all different
- overlap
- uni-directional

Figure 13.3: Three dimensions of visualization relationships based on view properties; many combinations can be useful.

may feature multiple views that are not or partly consistent and, thus, have only a weak visual similarity [265].

The second relationship type is data similarity, and expresses how big the overlap between the visualized data points of two charts is. When the data is exactly the same, this indicates that the two views show different representations for the same data subset. A weaker similarity is a data overlap, and no similarity means the data is distinct. These constellations can indicate certain exploration pattern, e.g., overview+detail (overlap). However, in many situations, data similarity must be considered with respect to visual similarity. For instance, some combinations of the two measures are not practical, e.g., a perfect visual similarity and a perfect data similarity describes the same visualization. In conclusion, data similarity provides an indication which views are related data-wise and, thus, can provide additional insights.

Finally, views can also have a relationship with respect to the data flow, which we define as connectivity. This involves mechanisms such as linked brushing in multiple coordinated views [257], or incorporating a selection in one chart as a filter condition in another. We distinguish between different connectivity levels; the strongest is an exclusive connectivity, where a view receives its data purely from another (e.g., a filter component). Linked brushing, instead, is an example of an additional, supplementary connectivity; however, both views would still be able to display data without this connectivity. Similar rankings of the connectivity can also be found in the literature; for instance, VisTiles [182] encoded this ranking by distinguishing between connections triggered by side-by-side combinations (i.e., stronger ones) and general connections (e.g., selections). Notably, the connectivity extends also to non-visualization components, e.g., UI elements for defining filters or aggregations.
13.3.3 User Preferences

Visualization interfaces are typically flexible and can be adapted to user preferences. We distinguish here between two types of preferences: general and task-specific preferences. General preferences are independent of a specific situation and derive mostly from how a user prefers to arrange things or what overall strategy for device organization he follows [120]. For instance, a user may want to keep a filter component on the right device border, or prefers to have one specific visualization on a specific device. Task-specific preferences emerge during the data exploration [6, 46, 327], and also affect the distribution. This can involve, e.g., aligning views for visual comparison, temporarily enlarging a visualization, or moving a view to another device to simplify interaction.

While multiple distributions of the same quality exist, they may fit analyst’s preferences differently. Thus, considering these user preferences helps to improve the system’s usability. However, retrieving such information automatically is challenging; instead, interfaces should provide adequate functionalities that allow users to express their preferences.

13.3.4 Device Properties

Devices today have a very wide spectrum of distinct characteristics, many of which have already been considered in a multitude of existing research [120, 135, 182, 225, 234, 326]. Likely the most important property is the available screen estate, determining how many visualizations can be displayed at what size. Since pixel density differs between devices, screen resolution should not be a sole measure as the resulting physical size is also important. Further, devices differ in the available input modalities, i.e., no input, touch, pen, mouse, or keyboard, and the resulting input accuracy [234] of these. The device type can also indicate useful information with regards to mobility or computation power. In combination with the ownership, this allows to distinguish between personal smartphones (mostly used by one person) or public large display (shared with multiple users) [135, 168, 213].

Besides these basic properties, further characteristics can be considered. Contextual information about the device’s posture, orientation, and user distance (i.e., user-to-device proximity) provide insights on how the device is used by analysts. For instance, hand-held devices are more likely to be used for input. Similarly, a distant device may require scaling up views for readability reasons. Further, advanced display specifications could be considered (e.g., viewing angles, color accuracy, brightness). However, such properties are hard to access and require external sensors or knowledge.

13.3.5 Device Relationships

Depending on the actual device ensemble, devices can step into different relationships during the interaction. While the theoretically possible combinations are
manifold, we focus here on realistic device combinations. The simplest combination is a two-display desktop setup, where the displays are aligned and form one big surface. However, in a scenario where a laptop is connected to a projector, these two screens act as separate units with different properties. The second case can also be applied to mobile devices (i.e., smartphones and tablets): they can be used in combination with a larger display or a desktop, as well as with multiple other mobiles. In these situations, devices differ regarding their type, size, input modality, posture, and distance, which makes it possible to assign certain device rules to them. For instance, smaller devices in addition to a larger device are most often suitable to host additional details and UI elements, or devices closer to the user can act as remote controls for a more distant device.

13.4 Heuristics for Deriving a View-Sensitive Distribution and Layout

Our design space and its dimensions can be used to both describe and generate layout strategies for cross-device visualization. In our work, we use these dimensions to derive six heuristics for distributing components of a visualization interface across multiple devices. With these heuristics, we aim to provide comprehensible and replicable high-level constraints. We found that formal specification, such as in the AdaM framework, is often costly with little practical gain, and—most importantly—results in definitions that are hard to relate to. In contrast, our heuristics are prescriptive also to human designers and can be used to guide the design of manual distribution, algorithms, or even optimizers.

Each heuristic contributes to different aspects of a distribution, such as view grouping or device assignment, while they also allow for promoting common analysis tasks (e.g., visual similarity supports comparison tasks). Specifically, we

![Heuristics Diagram]

Figure 13.4: The heuristics are incorporated for both the global device assignment and the local view arrangement on devices; in the process, heuristics can contradict each other.
consider the heuristics to be applied in a step-wise process (Figure 13.4), where a later heuristic can contradict earlier assignments. In this process, the heuristics can be detailed, weighted, and transformed into a specific quantification; our Vistribute implementation serves only as one example.

13.4.1 Grouping & Alignment Based on View Relationships

The relationships between visualizations can serve as indicators for which views should be grouped or aligned [216]. Therefore, we introduce three corresponding heuristics.

As pointed out above, views with a high visual similarity promote visual comparison. Based on common practice, such as in small multiple displays and scatterplot matrices, it is beneficial to place these views next to each other. Reducing the screen distance facilitates the user alternating their focus between the two views and, thus, to actually compare them. Aligning the views along a shared axis will further support comparison. Here, we utilize the visual similarity as an indicator if and how well two views are comparable. We consider a high visual similarity as the strongest type of relationship between views that motivates juxtaposing them. However, a lower visual similarity is often not of interest. We define the heuristic as follows:

**Heuristic 1 (Visual Similarity).** If two views are visually very similar, they should be both juxtaposed and aligned.

The second driver for grouping is data similarity. Placing the views with a high data similarity close to each other, i.e., forming view groups, can support the search-related tasks of users [46] as well as focusing on related aspects (cf. the semantic substrate concept by Chung et al. [66]). For instance, if multiple views encode the exact same data subset and are placed next to each other, they will provide different visual representations of the same subset. Similarly, this applies to other constellations, such as overview+detail patterns (i.e., one view shows a subset of the other view). However, this relationship is not as strong as the visual-similarity-based one, and typically does not require an alignment of the views. Further, it may also depend on the type of visual similarity: for example, a subset relationship eventually represents a useful overview+detail pattern if the two views are also of the same type. As a result, this heuristic focuses on data similarity, but also incorporates visual similarity:

**Heuristic 2 (Data Similarity).** If two views have a high degree of data similarity and a corresponding visual similarity, they should be placed close to each other.

As described before, views can consume data from another view and either rely on it exclusively (e.g., filter), or use it as a supplementary input (e.g., linked brushing). In the first case, the component providing the input must be accessible so that the other view can be used. Therefore, it is beneficial to place it close to the affected view, in order to emphasize their dependency. Also, and similar to
visual similarity, proximity helps to reduce the cost of attention switches between the input component and the affected components. This is also true when the connection provides supplementary input. In all cases, a close proximity of the views is desirable:

**Heuristic 3 (Input Connectivity).** *If an interface component serves as data input for others, it should be placed close to the affected components.*

As a result of these heuristics, we expect two types of view groups: (i) strong groups that result in guaranteed alignment, and (ii) weak groups that lead to view proximity, but also can be split up in case of insufficient space.

### 13.4.2 View Adjustments and Device Assignments

The next step towards the distribution is considering the single views with respect to the current device ensemble.

First, it should be identified how much space a view requires: although exceptions may exist [152], generally, the more data points a visualization encodes, the more it benefits from being scaled up [195]. For instance, a bar chart showing three bars requires less space than one with 50 bars. Similarly, a scatterplot encoding hundreds of data points should be allocated more space than one with 10 marks. While the optimal size in relation to the number of data points always depends on the visualization type, it is still a good estimation of relative space requirements. Finally, many visualizations are sensitive to changes in their aspect ratio. Therefore, scaling should be uniform or only slightly alter the aspect ratio to avoid tampering with the original perception.

**Heuristic 4 (Data Density).** *A view should be allocated space proportional to the number of data points it encodes.*

Second, we consider the device suitability, which expresses how well a certain device can fulfill the requirements derived from a view or a group of views. These requirements mainly comprise the space requirement, input accuracy, and relations arising from the connectivity. For instance, views with a high space requirement are likely to be placed on a larger display. However, the suitability has not always an impact, i.e., when all devices are very similar, and, thus, interchangeable. For instance, when only tablets are available, it does not matter which part of the interface is distributed to which device. In contrast, with high diversity in the device ensemble, device suitability can be used for assigning different device roles (see device relationships described in design space). This can lead to exceptions of the grouping, e.g., components serving as an input can be moved to a mobile device and act as a remote control for the larger displays. In summary, device suitability is a main constraint in diverse ensembles:

**Heuristic 5 (Device Suitability).** *If devices are diverse, view assignments should be guided by device suitability.*
13.4.3 User Preferences

No matter how advanced a view distribution system is, users should be able to change the layout based on their preferences or current situation. These preferences can involve, e.g., a fixed placement of some views, an altered alignment, or even the exclusion of certain devices or components. These constraints should always be reflected in the distribution and overwrite the definitions coming from the other heuristics. Furthermore, these preferences should be stored and reapplied automatically, but must be editable by the user.

**Heuristic 6 (User Preferences).** *If user preferences are applicable, they outweigh all other heuristics.*

In the context of analysis tasks [6, 46, 327], i.e., temporary user interests, it could be theoretically possible to infer these automatically based on user interactions. For instance, if a user makes alternating selections in two views, this can express the need to bring the views closer together. As we explicitly left room for weighting the heuristics, this allows for optimizing the distribution for the current task, e.g., emphasizing data similarity (H2) and connectivity (H3) to support investigating related items (*connect* [327]). However, too many (unexpected) interface changes must be avoided.

13.5 The Vistribute System

We implemented a web-based system that is able to (i) extract required properties from visualization/UI components and connected devices, (ii) derive and apply a distribution, and (iii) allow user adaptions via a control panel. The implementation is one of many possible instances of our heuristics; for each feature, we will reference the related heuristic. Stated quantifications/values were determined empirically.

13.5.1 Underlying Systems and Dependencies

Our implementation builds upon three existing system layers: Webstrates, Codestrates, and Vistrates. Webstrates [169] provides the underlying synchronization (of, e.g., states, selections, device information) across devices. Besides an in-browser computing environment, Codestrates [250] provides a package management system based on Webstrates. Vistrates [19] is a visualization layer for Codestrates offering specific visualization components and a data-flow-based execution model. This combination provides common visualizations and the possibility to connect them to a data source or with each other, hence, providing all tools to create an adaptable and full-fledged visualization interface.

Our distribution layer is implemented as a Vistrates meta-package and makes use of the offered functionality of the before-mentioned layers, e.g., when accessing

[github.com/tomhorak21/vistribute](https://github.com/tomhorak21/vistribute)
view properties (including states and data flow configurations). The distribution algorithms are run on one client; the resulting distribution is synchronized with all clients as a JSON object. Then, the clients move their assigned views to the given position on an interface layer. The creation of the visualizations and their connections is, however, left to the user.

13.5.2 Deriving Properties

The first step for the distribution is to derive all required information, i.e., visualization and device properties.

View Properties and Relationships

To extract these properties, we directly access the standardized state of the VisTrates components, e.g., template, size, data source(s), and accessed data properties. Based on the rendered view, we can distinguish between visualization and UI components. We also identify the incoming data as a basis for following steps.

The visual similarity is calculated by comparing selected properties and assigning points for matches; specifically, we consider the component template (3 pts; comprises type and encoding), dimensions (i.e., consumed data properties; 2 pts), number of data points (1 pt), and size (1 pt). By traversing the components’ data source, we extract the connectivity (exclusive or supplementary) and the data similarity (none or same). For performance reasons, data points were not compared directly; instead, we determine the closest common source and check if the data structure changes on the way (by, e.g., aggregation). While this does not allow detecting data overlap, it provides an indication if the data structure is the same.

Device Properties

Current browsers provide access to a set of device specific properties, allowing us to characterize as well as (re)identify them. Besides common properties such as resolution, language, platform, and user agent, in many cases also hardware-specific properties (e.g., parallel threads, memory size, CPU, GPU) are available. However, some device information is missing, e.g., advanced display properties, physical size, or attached input devices (e.g., keyboard, mouse). As a result, we cannot distinguish larger displays (e.g., digital whiteboard, projector) from desktop displays, as their resolution is identical. Similarly, contextual information (e.g., user proximity, ownership) would require external sensors.

Notably, one physical device can host multiple clients (e.g., laptop with projector), where each client should be considered independently. At the same time, in some setups multiple clients must be perceived as one unit (e.g., display wall consisting of multiple displays), even if they are not hosted on the same device. Therefore, we introduce an abstracted representation of a device called surface. Each surface represents one or more clients and maps its resolution to them. For
13.5. THE VISTRIBUTE SYSTEM

Figure 13.5: Example distribution illustrating $H_1$ and $H_3$. On the laptop, views form a block based on their connectivity to the filter ($H_3$, exclusive); the line charts form two pairs based on their visual similarity ($H_1$, all-same, 7 of 7 pts).

the distribution, only these surfaces are considered; except for resolution, the device’s properties are inherited.

13.5.3 Distribution: Grouping, Assignment, and Adjustment

As described before, we consider the distribution to be a multi-step process. The first step is to identify the view groups and their types (strong and weak). To qualify as a strong group, views must have an exact visual similarity ($= 7$ pts; $H_1$), while weak groups are formed based on data similarity ($H_2$) and connectivity ($H_3$).

An example distribution is given in Figure 13.5. In addition to these groups, we also calculate a relative space requirement $V_{SR}$ for each view based on the number of visualized data points (damped via $\log_2$) and normalized so that $\sum V_{SR} = 1$ ($H_4$). Similarly, based on the available area, we calculate a relative screen estate $S_{SE}$ for each surface, again with $\sum S_{SE} = 1$.

Next, we identify special view-device pairs, e.g., offloading input components to smaller mobile devices, and assign the views directly to the surface ($H_5$). Then, we proceed with the default assignment of views to surfaces based on the space requirement ($H_4$). We consider strong view groups first, then weak view groups, and finally all other views. If no surface is big enough to exclusively host a group, we either accept to scale down the views (strong groups), or to split them up across multiple devices (weak groups).

The last step is arranging the views on each surface. Here, we applied an approach similar to bin packing: basically, we create columns and fill them up until the available surface height is no longer sufficient. The initial size of views is based on their space requirement in relation to the surface’s screen estate (i.e., $V_{Area} = V_{SR} \times S_{Area}$; $H_4$). Because of different aspect ratios and sizes, some rows may not fill up the whole column width; in these situations we try to fill up the spots with smaller views. While adding views to columns/rows, we allow for a flexibility in view size and aspect ratio (up to 25%). As constellations can exist, where views cannot be fit into the available screen space (e.g., because of contrary aspect ratios), we scale the whole layout down to fit into the surface.
Finally, we again adjust view height and width up to 50% to eliminate any free space. Although our implementation does not explicitly align views yet, this approach typically maintains the alignment/grouping implicitly as the views are processed in order of their group membership.

13.5.4 Control Panel for User Adaptations

Our implementation provides a control panel allowing users to fine-tune the distribution (H6). The panel shows the surfaces and distributed views in both a preview and lists. The lists provide indicators for group membership and space requirement and allows ignoring surfaces and views, making them ineligible for automatic layout. Views can also be manually assigned to surfaces by drag and drop. The system reacts differently to these changes: while ignoring views or surfaces triggers a recalculation of the complete distribution, the manual assignment only re-runs the local layout. Here, we expect users to have the mental model of reassigning one specific view, regardless of its relations to other views. Therefore, we skip the view assignment to avoid side effects. The user can also switch to a completely manual process, where they can place and scale views freely.

Currently, distribution updates are only triggered on major changes, such as a changed device configuration or when new views are added to the interface. In these situations, we fade in a miniature overview map of the surface configuration highlighting moved views and/or new surfaces. However, smaller view-specific changes, e.g., caused by filter conditions, are ignored to avoid interrupting the user.

13.6 Study: User-Created Distributions

In order to back up our heuristics, we compare distribution and layout generated by our system to multiple user-created ones as well as report on user ratings of the distributions.

Participants

We recruited six paid participants (age M=36.8, SD=12.59 yrs; 1 female, 5 male) at the University of Maryland. We required that all of them have both theoretical and practical background in data analysis and/or visualization theory, i.e., actively conducting research in this area or work with these interfaces on a regular basis. All participants have been active in the field for at least 3 years (M=9.8, SD= 10.26 yrs).
Apparatus and Dataset

We used the Vistribute system as described before on a crime dataset from the City of Baltimore. The example interface consisted of 10 views. Two bar charts showed the overall crime distribution for districts and crime types (BC-Dist-All, BC-Types-All). Selections in these were used as a filter for two connected line charts each, showing the distribution over time for 2016 and 2017 (LC-Dist-16/17, LC-Types-16/17). A filter component allowed for filtering the data to explore subsets (FILTER). The filtered output was consumed by two bar charts (weapons, BC-Weap-Filt; inside/outside location, BC-InOut-Filt) and a map (Map-Filt). We extended the prototype with a manual layout mode, allowing a free view assignment and arrangement using the control panel. Once placed on a surface, views could also be moved and resized directly in the interface.

Physical Setup

We included three device ensembles:

S1 A traditional dual-display desktop (each 24″, full-HD, 1 landscape, 1 portrait);

S2 A novel desktop setup with a laptop (13″, 1600 × 900 px) on a standing desk and a large display (55″, full-HD) within arm’s reach; and

S3 A mobile device ensemble consisting of a tablet (HTC Nexus 9, 9″, 2048 × 1536 px, landscape), a smartphone (Samsung Galaxy S8, 5.8″, 2690 × 1440 px, landscape), and the laptop from before.

We chose these setups as they represent realistic combinations that are already in use or are likely to be commonly used in the near future. Figure 11.1a-c shows similar setups.

Procedure

Participants first received a short introduction on view distribution as well as the experimental dataset. We provided them with an initial understanding for the requirements of a distribution by explaining typical scenarios and tasks in the context of the crime dataset. We also explained the abilities and connections of the existing views as well as provided a printout showing these connections.

In Phase I, participants were asked to distribute all views across the available surfaces for all three setups (within-participants, counter-balanced order). None of Vistribute’s automatic layout functionality was active during this phase. We asked participants to think-aloud while distributing views and logged the created distributions. As the interface offered no support for alignment, we carefully

https://data.baltimorecity.gov
adjusted them afterwards to remove smaller and unintended overlaps or offsets. These adjusted distributions were used for Phase II.

In Phase II, participants were shown three existing distributions for each setup. For all distributions they were asked to rate its quality on a 5-point Likert-scale and provide free-form comments. Since we included three physical setups, each participant rated nine distributions. The setup order was the same as in Phase I. From the three distributions, two were created by prior participants (randomly selected), while one was generated by Vistribute. Their order was also randomized per participant. We did not indicate to participants how these distribution were created. For the first two participants, we used distributions created during earlier pilot runs. In total, sessions lasted approximately one hour.

13.6.1 User Feedback and Findings

We found three main results: when considering a distribution, (1) participants make decisions based on very similar aspects as embodied in our heuristics, but (2) personal preferences have a strong influence leading to diverse distributions across participants (Figure 13.6a), and (3) the manual distributions were rated slightly better than the automatic ones (Figure 13.6b).

When stating their thoughts during the distribution, participants touched on similar principles as covered in our heuristics. For instance, they explicitly stated that views with more data points should be placed bigger (P1–6), connectivity must be valued (P1–4, P6), or that similar views should be aligned for comparison (P3–P4, P6). Figure 13.6a also shows some of these patterns, e.g., the line charts (LC-Types-16/17 and LC-Dist-16/17) form clear pairs, and, especially for S2, are...
often assigned to the same device (e.g., Figure 13.5). We also observed participants considering the influence of device size (P1, P3–4, P6) or input capabilities (P2–4, P6).

However, multiple aspects were considered differently across participants. While most participants valued smaller devices as appropriate for input purposes, P2 used the mobile devices explicitly for visualizations, as these “can be easily passed around.” For connectivity, we observed that some participants strongly favored placing connected views adjacent to each other (P1, P6), while others found it useful to split them between devices. We also found that some aspect are not covered in our framework yet: multiple participants had a higher-level definition of data similarity by considering their semantics. As an example, the views encoding districts (LC-Dist-16/17, BC-Dist-All), the map, and the Inside-Outside bar chart were classified as geographical data, and therefore combined by three participants (P2, P4, P6). Participants (P1–2, P6) also mentioned the importance of surface adjacency and its influence on the perceived proximity between views.

As a result, we could observe a high diversity across the created distributions. In Figure 13.6a, this can especially be observed for the bar charts in S1 and S2, as well as for most of the views in S3. Further, no two distributions were similar. Three distributions for S1 and two for S2 used the same view-to-surfaces assignment; however, they had different local layouts. This diversity in user preferences can also be observed in the ratings in the form of high standard derivations. On average, participants rated the manual distribution (M=3.9, SD=0.99) slightly better than the automatic ones (M=3.6, SD=1.21; see Figure 13.6b). However, the ratings must be considered carefully: our study included only a small number of participants and they all worked only for a limited time on the distributions without performing specific analysis tasks.

Interestingly, multiple participants found the distribution “exhausting”, and one participant explicitly stated that “the computer should suggest where to put things; there should be some optimization for this” (P5), also stressing that a manual placement is considered a burden (P1, P5). On average, participants spent 8 minutes on the second and third distribution (M=19.6 minutes for the first one). Although a certain part of this time is caused by the think-aloud design and lacking interface support for aligning, even in a real-world system users would eventually have to spend a couple of minutes for the distribution. Any shortcut offered by an automatic distribution would therefore be an improvement. Finally, P1 also noted that “semantically beautiful is much more important than aesthetically beautiful.” Hence, even if an automatic approach is not able to reach the visual quality of a manual one, it may still be able to provide a valuable layout. All created distributions are listed in the supplementary material.
13.7 Discussion & Future Work

We believe that our framework can serve as a foundation for future research on distributed visualization. Although limitations remain, we hope to stimulate follow-up work on distribution approaches as well as aspects even beyond that. Our long-term goal is to simplify the usage of multi-device environments so that their full potential can be realized.

13.7.1 Limitations, Framework Extensions, and Evaluations

Participant feedback indicates that some of our heuristics or the implementation could be refined; for instance, a semantic data similarity (e.g., all location-related views) or contextual device aspects (e.g., physical device arrangements) are currently not represented, as they are hard to capture. For example, device proxemics [21, 207] can currently only be sensed with external tracking systems, which are hardly applicable outside of research prototypes [182, 251, 322]. However, this might change as internal device sensors improve [156], allowing to better facilitate cross-device dependencies.

Our current Vistribute implementation dynamically responds to changes in the device and view ensemble by automatically recomputing the distribution. Unfortunately, such events may trigger a radical rearrangement of the distribution, particularly if the surface in question is large. Beyond the overview minimap, we currently provide no mechanism to help a user reorient themselves when this happens. In the future, we may want to incorporate specific technologies to visualize changes [11, 129], e.g., by using animated transitions showing how views are rearranged from one distribution to another, or by using transient color highlights [20]. Also, distribution layout changes may require explicit user confirmation. Finally, a history of the latest applied distributions could allow switching between different variations.

In extension to our current study, more thorough evaluations should be conducted. An extensive observation study on how users manage visualizations in MDEs during an analysis session could provide further insights, e.g., how often they want to adapt the interface and for which tasks. In a quantitative manner, it would be interesting to measure performance indicators (e.g., task completion time, error rate) in comparison to non-optimized or random distributions.

13.7.2 From Heuristics Towards Formalism

While the Vistribute framework does not stipulate a specific distribution algorithm, our example implementation is a rather simple algorithm realizing our heuristics, rather than a formal user interface specification such as AdaM [234]. For this paper, we explicitly eschewed such a formal approach, since we felt that current practice in arranging visualization views is mostly qualitative in nature. Instead we relied on heuristics that could be balanced for each specific imple-
The results from our evaluation bore this decision out; our automatic layouts were similar to layouts hand-crafted by experts. Nevertheless, extending our current algorithm towards an optimizer can help to improve the distribution quality, especially when cases exist that cause sub-optimal layouts. This could be done by running multiple variations with different parameters and identifying the best one.

Beyond that, nothing is preventing us from implementing a formal version, akin to AdaM, based on our heuristics in the future. This could also be further extended by applying machine learning approaches for deriving weights for the heuristics. However, as machine classifiers require a large training dataset of successful distributions, this can only be a second step after introducing distributed visualization interfaces to a broader audience. Finally, even when following this vision towards a distribution purely based on formalism, we believe that allowing users to modify the result is central. Notably, it should be possible to apply these adaptations in a natural way, e.g., by drag-and-drop, and not through abstract parameters, as it is often the case for current optimizers [234].

13.7.3 From Distribution Towards Visualization Generation

In the process of developing our framework, we noted several times that being able to automatically generate and modify the views (instead of working with existing views) would make our approach more powerful. For instance, when scaling a view, this would make it possible to optimize the aspect ratio for improved perception [127]. Instead of just aligning two views in order to promote visual comparison, an even more sophisticated approach would be to rebuild the views to use the same chart type and normalize both of their scales to further increase consistency [246]. This step, to either generate views to complement existing ones, or even to generate a complete dashboard from scratch [221], is not far.

In other words, to truly realize the potential of multi-device environments for visual analytics, it may be necessary to entirely relinquish the task of visualization specification to the distribution middleware, merely specifying the datasets and tasks involved. Unlike the human designer, who can only enumerate so many variant visualizations for a finite set of possible device ensembles, a fully automated visualization generation engine would be able to construct precisely the visual representations that are best suited to the available hardware, physical context, and overarching analysis task.

13.8 Conclusion

We have presented Vistribute, a combined design space, set of heuristics, and prototype implementation for cross-device distribution of visualizations across a dynamically changing ensemble of displays and devices. Informed by current visualization practice, we have validated our heuristics and their implementation
in a qualitative evaluation where visualization experts manually constructed distributed layouts. Our findings suggest that there is little qualitative difference between manual and automatic layouts, and that the automatic layout can save significant time and effort.

Acknowledgments

We thank Karthik Badam for his valuable feedback as well as Sigfried Gold for providing his voice for our video. This work was supported by the Deutsche Forschungsgemeinschaft (DA 1319/3-3, 1319/11-1), the Danish Center for Big Data Analytics Driven Innovation (IFD-5153-00004B), the Aarhus University Research Foundation, and the U.S. National Science Foundation (IIS-1539534). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.
Integrating Guided Clustering in Visual Analytics to Support Domain Expert Reasoning Processes

Andreas Mathisen\textsuperscript{1}, Matthias Nielsen\textsuperscript{2}, & Kaj Grønbæk\textsuperscript{1}

\textsuperscript{1}Department of Computer Science, Aarhus University, Denmark
\textsuperscript{2}The Alexandra Institute, Denmark

Abstract

Recent research shows promise in combining Information Visualization (IV) and Machine Learning (ML) to assist data analysis performed by domain experts. However, this approach presents non-trivial challenges, in particular when the goal is to incorporate knowledge provided by the domain expert in underlying ML algorithms. To address these challenges, we present an analytical process and a visual analytics tool that uses visual queries to capture examples from the domain experts’ existing reasoning process which will guide the subsequent clustering. Our work is motivated by a collaboration with personnel at the Danish Business Authority, who are interested in two types of insights: (1) On which data dimensions is a selected subset of companies different from the remaining companies? (2) Which other companies lie within the same multi-dimensional subspace? The poster will illustrate a real analysis scenario, where the presented analytic process allows auditors to use their knowledge of identified "suspicious" companies to kick-start the analysis for others.

14.1 Introduction

Combining IV and ML was recently suggested as being a core research objective at a Dagstuhl Seminar [163], to extend the existing work on using ML methods within visual environments. Numerous approaches have been introduced to visually convey high-dimensional data, for instance using lower dimensional projections or clustering algorithms. However, applying ML algorithms in practice is usually an iterative process, where the designer extracts new features and validates intermediate results. Since this process can be challenging, it typically requires domain expert knowledge. We present an analytical approach that exploits the scenario in which domain experts can provide a partial labeling, i.e. instances of interest to their analysis. The core idea is to find relevant clustering results using a two-round clustering approach guided by examples which domain experts can provide via visual queries.
14.2 RELATED WORK

Our work is based on a collaboration with personnel at the Danish Business Authority, who lack automated tools to systematically exploit their data to, e.g., uncover fraudulent behavior. We found that their analytical reasoning processes are often started from examples or risk factors derived from previous cases (e.g. bankrupt companies). Given the nature of available examples the resulting labeling of the companies is only partial which can be challenging to cope with in ML. Concretely, we found that the knowledge provided by the auditors suffers from two distinct characteristics, which we denote abstract and incomplete. A labeling is abstract w.r.t. label \( A \) if the items labeled as \( A \) are not similar in the feature space and therefore should have sub-labels, as illustrated with different decision boundaries in fig. 14.1a. A labeling is incomplete w.r.t. label \( A \) if further instances should have label additional to those currently labeled as \( A \), as illustrated in fig. 14.1b. Intuitively, these additional instances are of utmost interest, since they are similar to the provided examples in the feature space. Note, that a labeling can both be abstract and incomplete w.r.t. a label \( A \), and if this is the case it can be difficult to find satisfactory results with conventional supervised or semi-supervised learning methods.

Figure 14.2: A visual analytics tool where a parallel coordinates visualization (c) is enhanced with clustering functionality. Users can (a) select features of interest and (b) provide visual queries using brushes to the clustering process. Afterwards users can inspect the best results shown with the V-measures (d) using two coordinated views (c) and (e).

14.2 Related Work

Analysing high-dimensional data is an active field of research within both the IV community and the ML community. Liu et al. [197] recently provided a thorough review of the recent advances in high-dimensional data visualization. Several
techniques exist for visual mapping of multiple dimensions [91, 93, 144, 160, 227] as well as visualizing uncertainty [56]. However, visualizing all dimensions severely limits our ability to spot meaningful patterns. A common approach is therefore to project high-dimensional data to lower dimensional spaces to enable simpler visual mappings [63, 155, 196]. Visual tools have also been used to inspect ML results [100] in order to understand the output or to manipulate the model [49, 116]. The visual analytics concept is excellent to support exploratory analysis that incorporates domain knowledge [278] and various approaches have been proposed to achieve this goal [105, 123, 139].

14.3 Exploiting Domain Knowledge

To exploit domain knowledge that is abstract and incomplete we propose an analytical process consisting of three steps: (1) define examples, (2) generate clusters and (3) inspect results. In our prototype, we use conventional methods to visualize high-dimensional data; parallel coordinates [100, 144, 227] for the multidimensional features space and scattersplots for the reduced feature space. Figure 14.2 depicts the web-based prototype with two coordinated views that displays one of the potential clustering results.

(1) Define examples: The user can provide examples using visual queries (brushing in our case [31, 132]) in the parallel coordinates visualization, which then generates a binary distinction. The instances satisfying the current selections are one group and the remaining instances constitute the other group. This allows to effectively compare the selected examples with the rest. The user can furthermore choose to limit the feature space by selecting only those features of interest to the current analysis.

(2) Generate clusters: A two-round clustering is utilized based on the visual query of a user. In the first round, clustering is performed on each initial group of instances defined by the user’s query. In this round we use the silhouette coefficient [260] to reason about the structural properties of the clusters to find the optimal number. The result of the first round is a sub-labeling of the examples, i.e. it is a way to deal with an abstract labeling. In the second round, clustering is performed on the entire data set to deal with an incomplete labeling. In this round we use combinations of the sub-labels found in the first round together with the V-measure [259] to find the optimal parameters. While our method is not specific to a single clustering algorithm, we use the K-means clustering algorithm [13] due to its speed. We search for results both in the number of clusters and in the feature space, and continuously report the best results found so far. To verify the usefulness of our process, we applied it also to the popular Iris data set [194]. The Iris data set contains 3 classes, but using clustering on this data set will traditionally yield only 2 clusters. However, if an expert can provide a partial labeling which separates the majority of the two similar classes, our approach will suggest 3 clusters.
(3) **Inspect results:** The clustering results will be presented as a new axis in the parallel coordinates visualization and color-coded in the scatterplot, where the PCA algorithm is used to reduce the feature space. The views are coordinated, so users can update both views by either hovering the scatterplot or by creating filters in the parallel coordinates visualization.

### 14.4 Applied to the Business Auditing Case

The motivating use case for this analytical approach is to support business audit personnel in identifying fraudulent or otherwise troublesome companies. Currently, the selection of which companies to investigate is based on whether individual companies satisfy some of the known risk factors, using either historical registration data (e.g. board members), employment data or financial data. As an example, we converted the registration data to features by counting the number of occurrences for each type of registration. We then normalized the resulting data with the time span between the first and last occurrence. The data presented in Figure 14.2 shows the companies in Denmark with the most registration updates. In the example in Figure 14.2, all companies with a status different from normal are queried as one class. The resulting labeling is abstract, since the status does not describe why a company has gone bankrupt or been forced to dissolve. From this example we for instance learned that if a company changes name more frequently than business type and legal district, they are within a cluster where 100/202 of the companies have stopped. Since we believe the labeling to also be incomplete, we interpret the 102 remaining companies to be more suspicious than a random one out of all the 3836 normal companies.

### 14.5 Conclusion

In real world scenarios it is infeasible to expect perfect domain information, hence we have presented an approach that can still utilize partial information in the underlying clustering process. We present a prototype tool that incorporates our analytical approach and we provide a proof of concept of our approach in a relevant use case. Immediate future work include enhancing the usability of our prototype by doing additional user studies with the Business Audit personnel. We will also investigate how to mitigate potential expectation or confirmation biases, which can be prominent when inexperienced users are evaluating ML results.

### Acknowledgements

This work was conducted in the DABAI project (IFD-5153-00004B) supported by the Innovation Fund Denmark.


[121] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. Frequent pattern mining: current status and future directions. Data Mining and Knowledge Discovery, 15(1):55–86, 2007.


[312] Wesley Willett, Jeffrey Heer, Joseph M. Hellerstein, and Maneesh Agrawala. CommentSpace: structured support for collaborative visual analysis. In


