Data expression : understanding and supporting alternatives in data analysis processes

Jiali Liu

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2021IPPAT022

Thèse de doctorat

Data Expression: Understanding and Supporting Alternatives in Data Analysis Processes

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom Paris

École doctorale n°626 l'institut polytechnique de Paris (ED IP Paris)
Spécialité de doctorat : informatique

Thèse présentée et soutenue à Palaiseau, le 23/09/2021, par

JIALI LIU

Composition du Jury :

Harald Reiterer Professor, Human-Computer Interaction Group, University of Konstanz	Président
Susanne Bødker Professor, Department of Computer Science, Aarhus University	Rapporteur
Alex Endert Associate Professor, School of Interactive Computing, Georgia Institute of Technology	Rapporteur
Anastasia Bezerianos Assistant Professor, Université Paris-Saclay, CNRS, Inria	Examinateur
Michel Beaudouin-Lafon Professor, Université Paris-Saclay, CNRS, Inria	Directeur de thèse
James Eagan Associate Professor, Télécom Paris, Institut Polytechnique de Paris, LTCI	Co-directeur de thèse

Contents

Abstract

To make sense of data, data workers consider different kinds of alternatives: they examine a variety of data sources, explore diverse sets of hypotheses, try out different types of methods, and experiment with a broad space of possible solutions. These alternatives influence each other within a dynamic and complex sensemaking process. Current analytic tools, however, make it cumbersome and cognitively demanding to handle such alternatives during sensemaking.

We focus on sensemaking as a human cognitive process enabled through appropriate tools. We address the following questions: *How do alternatives fit within the sensemaking process? How can tools better support the exploration and management of alternatives?*

We first conduct semi-structured interviews with twelve data workers to better understand the role of alternatives in sensemaking. We look at the different kinds and meanings of alternatives for our data workers, why they explore them, and how these alternatives fit into their overall sensemaking processes. Drawing upon our analyses and findings, we characterize alternatives using a theoretical framework based on participants' 1) degree of attention, 2) abstraction level, and 3) analytic processes. We show how this framework can help describe and reason the kinds of alternatives considered in sensemaking, the related processes, and how tool designers might create more flexible tools to better support them.

With this general understanding, we then study alternatives in the qualitative analysis context. We find that practitioners often mix alter-

native methods, compare multiple analytic lenses, and adopt different representations. In current tools, these alternatives and relevant artifacts are often disconnected, leaving analysts to cobble together workflows and apply ad-hoc strategies. We implement a proof-of-concept prototype, ADQDA, to explore how analysts can appropriate available digital devices as they fluidly migrate between analytic phases or adopt different methods and representations, all while preserving consistent analysis artifacts. We validate this approach through a set of application scenarios that explore how it enables new ways of analyzing qualitative data that better align with the identified alternatives in sensemaking practices.

Finally, we study alternatives in the context of reuse. Analysts often reuse contents generated in computational notebooks to other places like presentation slides. We consider the original content and its derivative as "alternatives" that diverge from the moment of reuse. These alternatives might be applied with the same updates or be modified differently during the sensemaking process. We introduce "computational transclusion" as a novel reuse approach that maintains the links between pairs of alternatives to facilitate tracking and coordinating changes. We explore this concept in the context of transcluding data visualizations. By examining various reuse scenarios, we propose a set of transclusion properties to clarify the different kinds of links between the alternatives. We also explore how to reify them in the user interface to enable flexible (re-)configurations of transclusion to suit users' contextual needs.

We conclude that tools can better support sensemaking by adapting to data workers' contextual needs and linking associated alternatives as they arise in the sensemaking process. Chapter 3 explores the kinds of alternatives that arise in sensemaking; Chapter 4 shows how tools can link alternatives and be appropriated within the specific context of qualitative data analysis. Chapter 5 probes different types of links between alternatives within the context of reuse.

Résumé

Pour bien comprendre les données, les “data workers” considèrent différents types d' alternatives: ils examinent une variété de sources de données, explorent diverses hypothèses, essayent différents types de méthodes et expérimentent un large éventail de solutions possibles. Ces alternatives s'influencent mutuellement dans un processus dynamique et complexe de “sensemaking”. Pourtant, les outils d'analyse actuels sont lourds et cognitif exigeants pour traiter ces alternatives.

Nous nous concentrons sur le sensemaking comme un processus cognitif humain facilité par des outils appropriés. Nous répondons aux questions: *Comment les alternatives s'intègrent-elles dans le processus de sensemaking ? Comment les outils peuvent-ils mieux soutenir l'exploration et la gestion des alternatives ?*

Nous menons des entretiens semi-structurés avec douze “data workers” afin de mieux comprendre le rôle des alternatives dans le sensemaking. Nous examinons les différents types et sens des alternatives pour nos participants, les raisons pour lesquelles ils les explorent, et la manière dont ces alternatives s'intègrent dans leurs processus généraux du sensemaking. En s'appuyant sur nos analyses et résultats, nous caractérisons les alternatives en utilisant un cadre théorique basé sur (1) le degré d'attention, (2) le niveau d'abstraction, et (3) les processus d'analyse des participants. Ce cadre est capable d'aider à décrire et à raisonner les types d'alternatives dans le sensemaking, les pratiques qui y sont liées, et comment les concepteurs d'outil pourraient créer des outils plus flexibles afin de mieux les soutenir.

Avec cette compréhension générale, nous étudions ensuite les alternatives dans le contexte de l'analyse qualitative. Nous constatons que les praticiens mélangent souvent des méthodes alternatives, comparent plusieurs angles d'analyse, et adoptent différentes représentations. Dans les outils actuels, ces alternatives et les artefacts liées sont souvent déconnectés, ce qui laisse l'analyste griffonner des flux de travail et appliquer des stratégies ad hoc. Nous mettons en œuvre un prototype, ADQDA, pour explorer comment les analystes peuvent s'approprier les dispositifs numériques disponibles lorsqu'ils migrent de manière fluide entre les phases analytiques ou adoptent différentes méthodes et représentations, tout en préservant des artefacts d'analyse cohérents. Nous validons cette approche à travers un ensemble de scénarios d'application qui explorent comment ADQDA permet de nouvelles façons d'analyser des données qualitatives qui s'alignent mieux avec les alternatives identifiées dans les pratiques de sensemaking.

Enfin, nous étudions les alternatives dans le contexte de la réutilisation. Les analystes réutilisent souvent les contenus dans les "computational notebooks" à d'autres endroits, comme les diapositives de présentation. Nous considérons le contenu original et son dérivé comme des "alternatives" qui divergent à partir du moment de la réutilisation. Ces alternatives peuvent être appliquées avec les mêmes mises à jour ou être modifiées différemment au cours du processus de sensemaking. Nous présentons la "computational transclusion" comme une nouvelle approche de réutilisation qui maintient les liens entre les paires d'alternatives pour faciliter le suivi et la coordination des changements. Nous explorons ce concept dans le contexte de la transclusion des visualisations. En examinant divers scénarios de réutilisation, nous proposons un ensemble de propriétés de transclusion pour clarifier les différents types de liens entre les alternatives. Nous explorons également comment les réifier dans l'interface utilisateur afin de permettre des (re)configurations flexibles de la transclusion pour répondre aux besoins contextuels des utilisateurs.

Nous concluons que les outils peuvent mieux soutenir le sensemaking en s'adaptant aux besoins contextuels des utilisateurs et en reliant les alternatives associées dans le processus de sensemaking. Le chapitre 3 explore les types d'alternatives qui apparaissent dans le sensemaking ; le chapitre 4 montre comment les outils peuvent relier des alternatives et être appropriés dans le contexte de l'analyse des données qualitatives. Enfin, le chapitre 5 examine les différents types de liens dans le contexte de la réutilisation.

Acknowledgements

Thanks to the support of many people, I am able to present my research contributions in this dissertation.

First and foremost, thank you to my supervisors, James Eagan and Michel Beaudouin-lafon, they are the ones who introduced me to HCI research and inspired me to pursue my Ph.D. degree. James has been always supporting me throughout my research journey. He offered me a lot of freedom and patience to find my own research interests. He always believed in me even during the moments when I feel lost. Without his share of knowledge, inspiring conversations, and encouraging words, I would not be able to produce the solid work presented in this dissertation.

I would like to thank the members of my jury: Anastasia Bezerianos, Susanne Bødker, Alex Endert, Harald Reiterer. Thanks to the insightful reports that Susanne and Alex gave to me as my thesis reviewers, I am able to better understand the strengths and limitations of my work. Thanks to Anastasia, also my mid-term reviewer, for providing me with valuable feedback. Thanks to Harald, my thesis president, for the efforts to organise my defense. Thank you all for taking your valuable time to read and examine my dissertation, and for asking inspiring questions during my presentation. You all offered me such a pleasant defense memory.

Thank you to Nadia Boukhelifa for being a great collaborator in one of my papers. She is an amazing researcher and always passionate

to pursue her research ideas. She provided me with valuable suggestions on how to conduct qualitative studies. I feel so happy to have the chance to work with her and to learn from her.

I also want to thank all Diva members. Professors Eric Lecolinet, Jan Gugenheimer, Samuel Huron, Rémi Sharrock, and Yves Guiard. All of them are exceptional researchers from whom I learned a lot. They offered me constructive comments on my research projects. They also opened up my horizons on the variety of research approaches and domains in HCI. Senior PhDs, Bruno Fruchard, Wanyu Liu, Marc Teyssier, Emmanouil Giannisakis, and Jessalyn Alvina provided me good advice and helped me out when I felt lost in my journey. By listening to their experience and stories, I was able to regain my motivation and confidence. Fellow PhD students Zhuoming Zhang, Mingming Qiu, and Wen-Jie Tseng, I am so glad to have you in the lab. I cherish the happy moments that we passed together.

I am grateful to have the chance to work with Clemens Klokmose at Aarhus University. Though my exchange in Aarhus got aborted after two weeks due to the COVID pandemic, Clemens and his team members still impressed me with their warmness and endless exciting research ideas.

Finally, thank you to my parents, who always unconditionally support me, believe in me, and give me all the love they have. I also owe a great debt to my lover, Quang Nhat Nguyen, the one who accompanied me through all my fragile moments, who has been enduring all my bad tempers and negative emotions, but still, always stay by my side.

CHAPTER 1

Introduction

With increasing access to diverse kinds of data sets, analytic tools, and interactive devices, people from various domains rely on data analysis to gain insight, inform actions, and make decisions. For example, an HCI researcher may analyze participants' data collected in experiments to verify the impact of a novel interaction technique. A police officer analyzes incident and crime reports to connect the dots and discover crime patterns. We refer them as "data workers." The process that people go through to collect, organize, and structure information for understanding is known as "sensemaking" [1, 2].

1.1 Motivation

Data workers often need to solve ill-defined and open-ended problems. Instead of following a straightforward pipeline, they need to consider and explore different kinds of alternatives: they examine a variety of data sources, explore diverse sets of hypotheses, try out different types of methods, and experiment with a broad space of possible solutions until they feel confident to draw conclusions [3–5]. These alternatives further influence each other within a dynamic and complex sensemaking process. For example, the application of an alternative visualization could reveal hidden insights and lead to an alternative hypothesis. This hypothesis, in turn, could require building different

models or collecting additional data sets.

Back in the 1960s, John Tukey used the term "exploratory data analysis" to describe the messy and iterative analysis processes [6]. Data analysis is like experimentation, where "the actual steps are selected segments of a stubbily branching, tree-like pattern of possible actions". Later in the 1990s, Pirolli and Card proposed a sensemaking model to characterize the analytic process. The model contains two major loops: the information foraging loop where analysts search for data and transfer it to meaningful information units; and the sensemaking loop where they organize data to schemas and form hypotheses [1]. During the overall sensemaking process, analysts arbitrarily mix these two loops and iteratively go back and forth between various analytic tasks.

Building upon their work, we consider data workers' sensemaking practices as a series of explorations and iterations on a combination of different types of "alternatives" — of hypotheses, data, algorithms, tools, etc. Despite the breadth of work around sensemaking practices, the particular role of alternatives remains under-explored. While past work has explored various concepts of alternatives, such as multiples and multiforms in data visualizations [7] or variations and versions in coding practices [8, 9], what constitutes an alternative remains ambiguous. Different meanings of alternative exist in different contexts, and the term itself encapsulates many concepts. To fill in this gap, the first goal of this dissertation is to provide a deeper understanding on the particular role of alternatives as they fit within data workers' sensemaking processes.

Tool designs can impact the way analysts perform, record, and share their work due to human-tool co-adaptation [10]. Without appropriate external tools, data workers could not conduct efficient sensemaking practices. Technology today might provide good support for transforming, visualizing, and even identifying potential patterns of data by using advanced machine learning techniques. Yet sensemak-

ing is still fundamentally a human activity. Data workers rely on their working experience and domain knowledge to select pertinent analytic questions, apply appropriate methods, reason on multiple schemas, and combine a variety of tools to fulfill these analytic tasks. Multiple types of data workers need to collaborate to perform more complex analytic tasks [4, 11].

Existing tools tend to serve a specific purpose with some predefined functionalities and workflows. It is hard to extend existing functionalities, adapt them for alternative purposes, or combine them based on analysts' contextual needs. For example, computational notebooks facilitate data workers to explore ideas by experimenting with code, but users need to apply ad-hoc strategies to version the code, compare different analytic paths, and record the underlining reasoning process [8, 12].

To explore alternatives, analysts often cobble together workflows, switch between different files, applications, and apply ad-hoc strategies. As a result, the alternatives which are meant to be related and connected end up in scattered and isolated files and applications. Analysts waste a significant amount of time and mental resources in trying to reconnect them and coordinate them as the analysis continues. Worse still, they might lose track of the explored alternatives, which can hamper their analytic reasoning and might lead to poor problem solving and decision-making [13]. Therefore, the second goal of my dissertation is to explore how to design tools that can be easily appropriated to explore alternatives and can re-establish the missing links among relevant alternatives to facilitate managing them.

Many inspiring concepts and theories have been brought up to redefine the relationship between tools and human activities. For example, the "shareable dynamic media" concept [14] imagine systems as information substrates that can be easily shared across people and devices, and can be dynamically modified from within. Instead of designing tools for certain tasks, Activity Theory studies heterogeneous devices

as part of an artifact ecology where "one can dynamically interplay with others and with users' web of activities" [15,16]. This dissertation demonstrates how these concepts and theories can be applied in building more flexible tools to support exploring and managing alternatives throughout the sensemaking process.

Thesis Statement Data workers consider different kinds of alternatives within a dynamic and complex sensemaking process. Digital tools can facilitate sensemaking by (1) applying various linking mechanisms to help track and manage associated alternatives; and (2) being re-configurable in terms of devices and features to let the user adapt these tools to their contextual needs during the overall sensemaking process.

1.2 Research Questions & Approaches

This dissertation has two high-level goals: first, to provide a deeper understanding of alternatives and their role in data workers' sensemaking processes; second, to explore how to build flexible tools that can better support the exploration and management of alternatives during the sensemaking process. We formulate two main research questions with a set of subsequent questions:

RQ1: How do alternatives fit within the sensemaking process? Do data workers consider alternatives in their sensemaking process? To what extent do they explicitly explore and manage alternatives? What are the different kinds of alternatives that arise? How do they fit within the overall sensemaking process?

RQ2: How can tools better support the exploration and management of alternatives? Do current analytic tools sufficiently support the practices around alternatives? When do they break down? How do encouraging tool design concepts and technologies can be applied to solve these problems?

Research Approach To answer these questions, we study sensemaking as the intersection of human cognitive processes and the tools that enable them. It is essential to combine both deeper understandings of real-world sensemaking practices around alternatives with encouraging concepts or theories on designing more flexible tools to support them. We thus apply a combination of various qualitative methods, such as interviews and observations, together with designing and prototyping new forms of tools to identify solutions to existing problems.

More concretely, we first conduct an empirical study to characterize alternatives in data workers' sensemaking processes. We present our findings and a framework of alternatives as a systematic way to think and reason alternatives in general. With the aid of this framework, we look into two specific analytic contexts: 1) alternatives in qualitative analysis using the affinity diagramming technique, and 2) alternatives in the context of reuse, where data workers reuse artifacts from computational notebooks to other places during sensemaking. We present our vision of more flexible tools which can better support alternatives in these two sensemaking contexts. We demonstrate show how new-ish technologies can be combined to realize our vision.

1.3 Contributions

Based on Wobbrock and Kientz's seven research contributions types in HCI [17], the primary contributions of this dissertation fall under three areas: theoretical, artifact, and empirical.

- **Theoretical contributions:** a theoretical framework to help describe, understand, and reason about the role of alternatives in general data analysis practices (Chapter 3); a design space on qualitative data analysis tools under the lens of alternatives (Chapter 4); a concept of "computational transclusion" that helps track and coordinate alternatives in the context of reuse (Chapter 5).

- **Artifact contributions:** a proof-of-concept prototype, ADQDA — a web-based, collaborative, cross-device affinity diagramming tool for qualitative data analysis (Chapter 4); a web-based sandbox system for exploring different kinds of reuse cases; and the implementation of several demonstrative scenarios on transcluding data visualizations (Chapter 5).

- **Empirical contributions:** semi-structured interviews with 12 data workers that report the reasons; triggers and barriers; different notions and processes around alternatives; as well as strategies applied by our participants (Chapter 3). Informal interviews with 3 affinity diagramming practitioners combined with our own experience that reveal real-world qualitative sensemaking practices with affinity diagrams and the pain-points with current tools (Chapter 4).

1.4 Outline

We outline the chapters of this dissertation as follows:

Chapter 1 introduces the motivation of this dissertation, our research questions and approaches, as well as our contributions. We demonstrate why we study alternatives in sensemaking processes and how current tools fail to support the exploration of alternatives (section 1.1). We describe what questions we aim to solve and how we plan to solve them (section 1.2). Finally we conclude with a summary of our contributions in this dissertation (section 1.3).

Chapter 2 presents the background of this work, including various notions around alternatives that have been studied and how existing sensemaking models capture them (section 2.1); past research on the different kinds of techniques and systems to support alternatives and sensemaking practices (section 2.2); the emerging concepts and theories that lead to redesign digital tools in a more flexible manner (section 2.3).

Chapter 3 investigates the role of alternatives as they fit within the sensemaking processes. We present our semi-structured interviews and our findings (section 3.1, 3.2 & 3.4). We characterize alternatives using a framework to help better describe, understand, and reason the kinds and notions of alternatives, as well as the related processes in sensemaking (section 3.3). We showcase its application on tool designs by looking at four specific analysis systems (section 3.5). We further discuss our work by comparing with other existing notions around alternatives and reflect on its limitations (section 3.6). Finally, we review the potential future research directions of this work (section 3.7).

Chapter 4 focuses particularly on alternatives in qualitative data analysis using the affinity diagramming method. We review related work around affinity diagramming practices and tools that support them (section 4.2). Based on informal studies and our own experience, we describe a design space to characterize data workers' sensemaking processes and the kinds of alternatives involved (section 4.3). We present our vision of flexible and linkable qualitative sensemaking tools and a proof-of-concept prototype ADQDA (section 4.4). We validate this approach through a set of application scenarios (section 4.5). Furthermore, we discuss the kinds of alternatives supported in ADQDA using the framework of alternatives as identified in section 3.3 (section 4.6). We end this chapter by discussing the limitations of this work and what can be done in the future (section 4.7 & 4.8).

Chapter 5 introduces the concept of "computational transclusion" as a novel reuse approach that maintains the links between alternatives in the context of reuse. We first introduces our motivation for envisioning "computational transclusion" (section 5.1). We review related work on existing reuse techniques (section 5.2). We explore transcluding data visualizations by examining various reuse scenarios using a sandbox system (section 5.3). We propose six pertinent properties of transclusion and explore how to reify them in the user interface to enable flexible (re-)configurations of transclusion (section 5.4). Finally, We discuss the limitations of our work as well as what can be done in the future (section 5.5 & 5.6).

Chapter 6 concludes how our research goals have been achieved and reviews our contributions in this dissertation (section 6.1). We further discuss what we have learned and the implications for future work (section 6.2).

CHAPTER 2

Background

This dissertation investigates how people make sense of data through the lens of alternatives, and explores how to design tools to better support exploring alternatives during sensemaking. There are three key concepts involved: sensemaking, alternatives, and tool design. This chapter expands on these concepts and their interplays.

In section 2.1, we review various notions and concepts around alternatives and present typical sensemaking models which formalize sensemaking processes. We discuss the role of alternatives either explicitly or implicitly mentioned in these models. In section 2.2, we look at how interactive systems are designed to support the exploration of alternatives and the overall sensemaking process. In section 2.3, we present a brief history of software design as well as emerging approaches and theories. We discuss how these theories inspire us to design more flexible tools to support alternatives in sensemaking.

2.1 Alternatives and Sensemaking

In this section, to define the scope of our work, we first explain the concept of sensemaking and the type of analysis tasks we consider. We then review different notions and concepts of alternatives and their role in existing sensemaking models.

2.1.1 Sensemaking and data analysis

The term and concept of "sensemaking" was introduced to Human–Computer Interaction (HCI) by PARC researchers Russell, Stefik, Pirolli, and Card in 1993 [1, 18]. It is also widely applied in domains such as psychology (Klein's sensemaking) [19, 20], library and information science (Dervin's sense-making) [21] and organizational science (Weick's sense-making) [22]. In this dissertation, we take the perspective of sensemaking in HCI — the process that people go through to build understanding out of a collection of data, to structure the unknown into insights, schemas, and hypotheses that can be acted on.

In sensemaking tasks, people often handle open-ended and ill-defined problems. The analytic process is like experimentation. Instead of following a pipeline, data workers often need to explore a broad space of alternatives and iteratively go back and forth to update, compare and manage them. The data to be analyzed could be either quantitative or qualitative or a mix of them. It could be obtained from diverse sources. Sensemaking has been viewed as a prerequisite to inform conclusions, solve problems, and support decision-making [23]. We present a more detailed review on sensemaking in the following section (section 2.1.3).

There exist another well-known term, "data analysis". We can use both "data analysis" or "sensemaking" to describe the process of making sense of data. These two terms could be distinguished based

on their focus: sensemaking and the studies around it tends to put a cognitive focus to analyze and describe the data analysis process; data analysis tends to be used with a computational focus to emphasize the operations on cleaning, transforming, and modeling data. Data analysis can be further divided into descriptive statistics, exploratory data analysis (EDA), and confirmatory data analysis [24]. Among them, exploratory data analysis, defined as "looking at data to see what it seems to say" by John Tukey [6], most closely resembles the kind of analysis tasks we consider.

In this dissertation, we leave aside this difference between these two terms. We use data analysis or sensemaking interchangeably to describe the process that people go through to understand data. Data workers, sense-makers, and (data) analysts are also used interchangeably to describe the groups of people who need to gain insights from data in their daily work. They might not necessarily consider themselves as data scientists due to the diversity of their domain knowledge, levels of programming skills, and degrees of formalism when they handle their analytic tasks.

2.1.2 Various concepts around alternatives

Merriam-Webster's online dictionary [25] defines an alternative as something that is "available as another possibility or choice." The notions and concepts around alternatives have been widely explored in the domain of data analysis and design. These two domains share common characteristics: they both portray highly iterative processes and handle ill-defined, open-ended problems.

Unlike our work, which focuses on the role of alternatives in the overall sensemaking process, past research on alternatives tends to focus on specific artifacts or particular analytic contexts. For example, the notion of "multi-forms and multiples" considers alternative forms

for data visualization [7]; the notions of "code versions" and "variants" examine alternative code pieces [8, 12, 26, 27], etc. These studies illustrate the diversity of the kinds and notions of alternatives, and why it is important to data analysis and design processes.

In the domain of design, working with multiple alternatives has being considered a central activity [28–30]. Tohidi et al. [30] find that designers, either working individually or in a team, explore multiple alternatives throughout all stages of the design process. They further find that exposing the design alternatives to end users can obtain more critical comments which may lead to better design solutions. Hartmann et al. [3] find that interaction designers build alternative prototypes to better understand the design space, reason about various design trade-offs, and enable effective decision making within organisations. They identify two distinct types of alternatives: source code alternatives for testing different application logic, and parameter variations for tuning and observing interactions at runtime. Terry et al. [31] further distinguish between the notions of variations and iterations for a design: variations are "alternative solutions to a given problem at a point in time," and iterations are "the same solution to a problem at a given point in revision."

Parametric and generative design methods enable users to explore a large set of alternative design solutions by tuning the expression of parameters and rules [32–34]. Woodbury et al. [29] define alternatives as "structurally different solutions to a design," while variations are "design solutions with identical model structure, but having different values assigned to parameters." However, the large number of alternatives also causes problems and challenges, including choice overload problem, high cognitive cost, and display constraints to view multiple alternatives simultaneously [28, 35, 36].

Focusing on the sensemaking domain, the importance of generating, elaborating, and testing alternative hypotheses has been emphasized in the process of intelligence analysis, a prototypical and

intensive sensemaking task [37–39]. Analysts often employ competing hypotheses to evaluate all reasonable alternatives [40]. Similarly, in the domain of business intelligence, analysts view alternatives as different scenarios explored to inspect the behavior of a complex system (i.e., the enterprise business). The process is referred to as "what-if" analysis, measuring variations of a given simulation model by changing a set of independent variables [41]. In digital marketing, Guo et al. [42] find that instead of showing the top-one result in event sequence prediction, users are more confident in making decisions when strong alternative predictions are displayed. Boukhelifa et al. [11] further study how different kinds of domain experts collaboratively make sense of simulation models, and find that alternative analysis scenarios branch out from previous research questions and hypotheses.

For data analysis tasks in general, recent studies show that data scientists write alternative versions of exploratory code [8, 12, 26, 27], apply multiple alternative models [43], or probe into variations on a single model [44]. For example, in sequence models, alternatives are defined as "reasonable and interpretable modifications of the model input and internals during inference mode to help with understanding and debugging." [45] Kery et al. identify challenges in terms of code alternatives for data scientists using computational notebooks, including duplicated and inconsistent alternatives, difficulty in generating, cleaning, and tracking alternatives, etc [8, 12, 46].

Data workers often apply visual analytics to make sense of data. By reviewing visual analytics systems, including Jigsaw [47], visTrails [48], and Tableau[1], Chen and Guenther identify three typical types of alternatives considered by them: data, visualization, and hypotheses alternatives [33,34]. Van et al. [7] propose the concept of "small multiples, large singles" which enables users to explore multiple alternative states of a visualization. Wood et al. [49] emphasize the importance

[1] `https://www.tableau.com/`, accessed on 21/03/2021.

of capturing and sharing design rationales behind each alternative visualization exploration path, and propose the "branching narratives" model to prompt such actions.

Apart from data analysis and design processes, conducting research is also a sensemaking process. Kale et al. [50] focus on the importance of alternative analyses to managing uncertainty in research synthesis. They refer to alternatives as a "garden of forking paths" — a series of analytical decision-points, each of which has the potential to influence findings. Guo and Laidlaw [51] propose using topic models as an aid for research idea generation, and find this approach leads to more divergent thinking and encourages users explore more alternative angles when elaborating the core aims.

The above research provides a variety of notions and concepts around alternatives. It tends to study specific types of alternatives under particular contexts. Our studies, instead, focus on the role of alternatives as they fit within the overall sensemaking process. We find that the different types of alternatives considered by our data workers align well with those discussed in previous findings. This dissertation contributes a theoretical framework to help describe and reason about the various kinds of notions and meanings of alternatives in sensemaking (chapter 3). It further provides notions of alternatives under two specific sensemaking contexts: qualitative data analysis (chapter 4) and sensemaking with computational notebooks (chapter 5).

2.1.3 Alternatives in sensemaking models

To understand how alternatives fit within data workers' sensemaking processes, it is important to understand sensemaking in general. Past research has examined sensemaking processes under different contexts and from different viewpoints. In this section, we review representative models developed to describe the sensemaking or data analysis process.

We discuss how these models consider, either explicitly or implicitly, the exploration of alternatives, as well as the kinds of alternatives they consider.

By observing how people make sense of large amounts of information about laser printers, Russell et al. [18] develop a sensemaking model that focuses on the external knowledge representations in sensemaking activities. As shown in figure 2.1, analysts first search for a good representation and then iteratively encode information in this representation. The central activity is identified as the "representation shift loop" — when analysts find data items that do not fit into the current representation, they need to keep adjusting the representation for better data coverage. This adjustment could lead to a variant on the current representation or a new alternative one. We consider that Russell et al.'s model emphasizes the "representation alternatives".

Klein et al. [20] take macrocognition approach and propose the

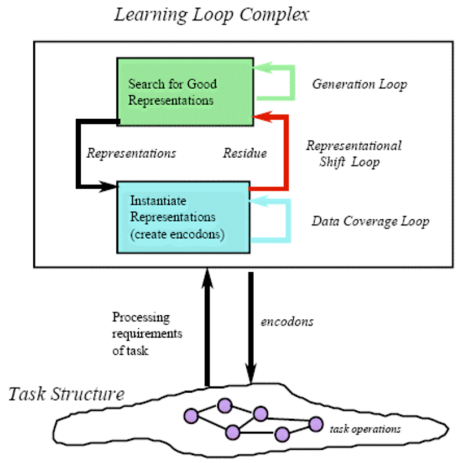


Figure 2.1. Russell et al.'s sensemaking model, demonstrating how analysts iteratively change the representation to fit data [18].

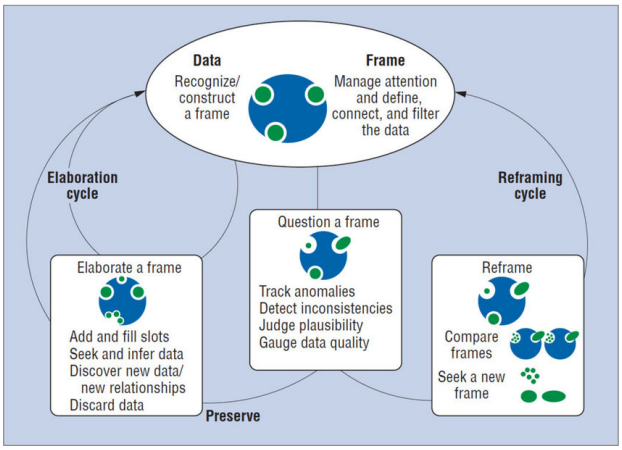


Figure 2.2. Klein et al.'s Data Frame Model, demonstrating an active two-way process of "fitting data into a frame" and "fitting a frame around the data." [20]

Data-Frame theory. They use the metaphor "frame" to represent the mental models held by analysts on the connections among data. A frame can be expressed in various forms, such as stories, maps, organizational diagrams, scripts, etc. They describe sensemaking as an active two-way process of "fitting data into a frame" and "fitting a frame around the data." Frame and data are thus interdependent: frames are built and iteratively updated based on data and in turn shape and define the relevant data, as shown in figure 2.2. For example, an analyst may generate new alternative frames when detecting anomalies or inconsistencies. Thus their model describes "frames" and data alternatives, as well as the interconnections between them.

Pirolli and Card [1] further extend Russell et al.'s sensemaking model for intelligence analysis in general. As shown in figure 2.3, the processes are structured by degree of effort and degree of information structure. The overall process contains two major loops: a foraging

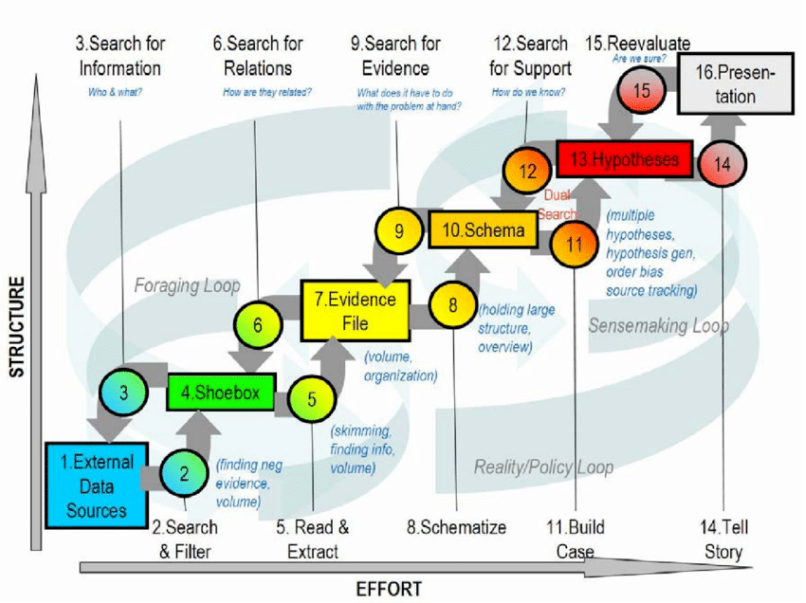


Figure 2.3. Pirolli and Card's sensemaking model, demonstrating the iterations between different analytic stages and tasks during the sensemaking process [1].

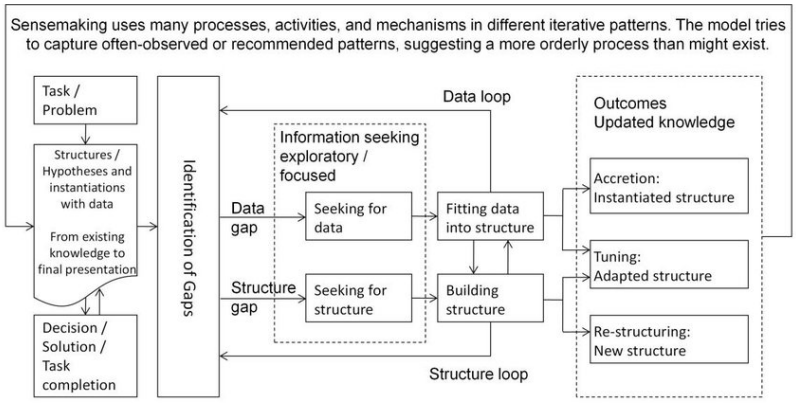
loop — where analysts search for information, often from various kinds of sources, and filter it based on the task-specific questions and goals; a sensemaking loop — where analysts organize the information into certain kinds of schemas to build mental models and update their understanding on the analyzed problem. An analysis could begin either top-down (theory or structure driven) or bottom-up (data-driven). No matter the strategy, it involves an opportunistic mix of them: analysts iteratively weave between forgaing and sensemaking processes, as well as their sub-activities. Each back and forth between analytic stages could potentially introduce certain types of alternatives. For example, the generation of alternative hypotheses could lead to gathering

alternative data items for processing.

Our research is largely inspired by Pirolli and Card's sensemaking model. We consider sensemaking as a messy and iterative process where analysts go back and forth between different analytic tasks with different kinds of internal or external representations. Instead of focusing on intelligence analysis, we study the practices of data workers in general and examine the role of alternatives as they fit within their analysis process.

Zhang and Dagobert [23] further extend existing sensemaking models by taking ideas and concepts from the fields of learning and cognition. As shown in figure 2.4, this model provides a more detailed look on the activities and cognitive mechanisms engaged in sensemaking. Similarly, Zhang and Dagobert emphasize the iterations between different activities. In terms of alternatives, they distinguish between the concepts of "tuning" and "restructuring": tuning represents the "evolutionary conceptual change in the schemas" or a "weak revision;" and restructuring, represents "conceptual changes that involve the radical change of existing structures or creation of new structures."

All these sensemaking models focus on cognitive process and describe the high level practices around individual sensemaking. Other data analysis models, which not necessarily emphasize on the cognitive process or mechanisms, also consider the importance of alternatives. Huber [53] identifies nine stages in the overall data analysis process and stresses the importance of conducting alternative what-if analyses. For example, he suggests pooling some subsets of the data or using different types of models. Guo [52] characterizes the process of research programming, where researchers from different domains write exploratory code to test ideas and obtain insights from data. As shown in figure 2.5, he identifies "explore alternatives" as one component of the model, which ties together the reflection and analysis process. Although these models consider alternatives as a single stage of analysis occurring within an iterative sensemaking loop, in practice



Information seeking is often divided into *searching for data and structure* and *searching for / extracting data and pieces of structure.*

Most sensemaking processes involve both external and internal representations, and the interplay between them. In each process, the cognitive mechanisms listed below can be used as applicable.

Inductive (data-driven, bottom-up)	**Structure-driven (logic-driven, top-down)**	**Both or Neither**
Key item extraction	Definition	Comparison
Restatement	Specification	Analogy
Judgment or evaluation	Explanation-based mechanisms	Classification
Summarization	Elimination	*Stereotyping*
Schema induction	Inference	Semantic fit
Generalization		Questioning
		Socratic dialogues

Figure 2.4. Zhang and Dagobert's Sensemaking Model, focusing on the processes, activities and cognitive mechanisms in sensemaking [23].

it may be difficult to separate the exploration of alternatives from other sensemaking actions. Huber explains the reason that "alternatives is represented in its own stage because it is difficult to order the sensemaking phases since they naturally and repeatedly appear in cycles between different actions."

In conclusion, despite that these models differ in emphasis and nuance, the basic pattern of switching and looping between different kinds of analytic stages and representations is in common to all models. They all explicitly or implicitly illustrate the exploration of

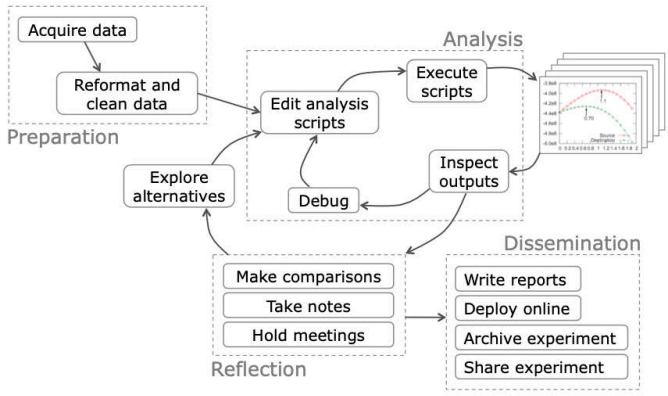


Figure 2.5. Overview of a typical research programming workflow by Guo [52].

alternatives, some consider it as a component of the model, and some as a driven factor for, or the consequence of the execution of other components in the model. Yet, they do not specifically study sensemaking under the lens of alternatives as our work. This dissertation contributes findings and a theoretical framework to describe and reason about the role of alternatives as they fit within the sensemaking process in general (chapter. 3). It also contributes the kinds of alternatives as they fit within the specific context of qualitative sensemaking (chapter. 4).

2.2 Interactive Systems & techniques for Alternatives and Sensemaking

Although data workers explore a variety of alternatives in their sensemaking processes, past research reveals that existing tools cannot efficiently support them. In this section, we review the systems and techniques proposed to facilitate the exploration of alternatives in the domain of data analysis and design (section 2.2.1). We also review several representative interactive systems that support sensemaking processes in general (section 2.2.2).

2.2.1 Systems & techniques for alternatives

Many analysis tools, including those used by data workers, rely on the Single State Document Model [31] — where only one state is supported in the system (or document) at any point in time. As a result, users need to develop workarounds and apply ad hoc approaches to compensate for the lack of sufficient tool support for alternatives. For example, data scientists duplicate code snippets and functions; interleave comment and un-comment; or rely on redo and undo commands to explore alternative analysis. They also apply informal or formal versioning methods to manage the explored alternatives [8,54,55]. Designers also improvise various ways to create and manipulate alternatives, such as creating and toggling layers; using a large canvas for viewing multiple designs; creating file naming conventions; and using undo and redo to cycle between different states [31].

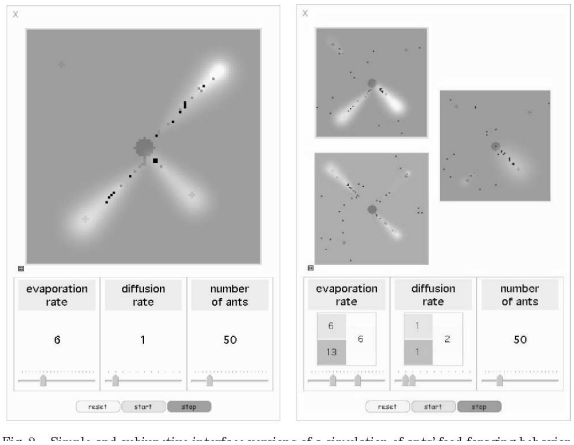


Figure 2.6. Subjunctive Interfaces, a technique to extend interface for exploring multiple alternative scenarios in information processing tasks [56].

A variety of methods and tools for have been introduced to better support the exploration of alternatives. Toomim et al. [57] invent "Linked Editing," a lightweight editor based technique to enable simultaneous authoring of alternative pieces of code. Terry et al. [31, 58] introduce set-based interaction techniques which allow simultaneous manipulation on multiple graphic elements. Side Views [31] supports explicit preview, comparison and generation of alternative visualization. In Parallel Pies [58], users can explore multiple parameter configurations of an image by sub-dividing the canvas for different transformations. Lunzner & Hornbæk's Subjunctive Interfaces [56] pioneered techniques for parallel exploration of multiple scenarios by extending the application's user interface, as shown in figure 2.6. Hartmann et al. [3]'s Juxtapose, a system designed for for interaction designers, supports both code alternatives while authoring application logic, and interface alternatives when tuning application variables at runtime.

The above systems and techniques are pioneers in supporting alternatives. They emphasize the requirement of simultaneous viewing, editing, and comparing multiple alternative instances, including source code, parameters, graphic or simulation results. In our research, we consider them as artifact alternatives.

Besides support for parallel editing of multiple alternatives, Zaman et al. implement post-hoc synchronization of alternatives in generative design in the GEM-NI and MACE system [36]. Kolarić et al.'s CAMBRIA [59] implements two kinds of operations to explore design alternatives: "pass variables" to enable the exchange of elements between alternative designs, and "pass values" to enable the propagation of element's property values to other alternatives with similar elements. CAMBRIA provides various kinds of layout for comparing design alternatives, such as juxtaposition and superposition, as shown in figure 2.7.

The development of interactive machine learning (iML) tools, such

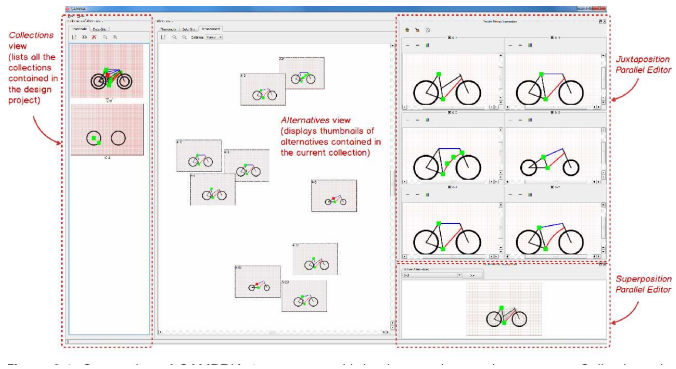


Figure 2.7. CAMBRIA system supports different kinds of operations on design alternatives and provide various kinds of layouts to facilitate the comparison of alternatives [59].

as RapidMiner [60] or orange [2], allow data workers to quickly explore alternative ML algorithms using visual workflows [61, 62]. However, users can only access to predefined components and methods. It is difficult to add additional algorithms, alter them for specific needs, or inspect into each component and vary its behaviors.

In visual data exploration, Elzen et al. [7] introduce the "small multiples and large singles" method and filmstrip metaphor to help non-expert users view, compare and interact with alternative visualizations to explore multivariate data sets. In the context of mixed initiative systems, EvoGraphDice [63] combines the automatic detection of visual features with human evaluation to propose alternative views of the data in the form of two-dimensional projections organized in a scatterplot matrix.

Build upon the literate programming concept [64], computational

[2] `https://orangedatamining.com/`, accessed on 27/03/2021.

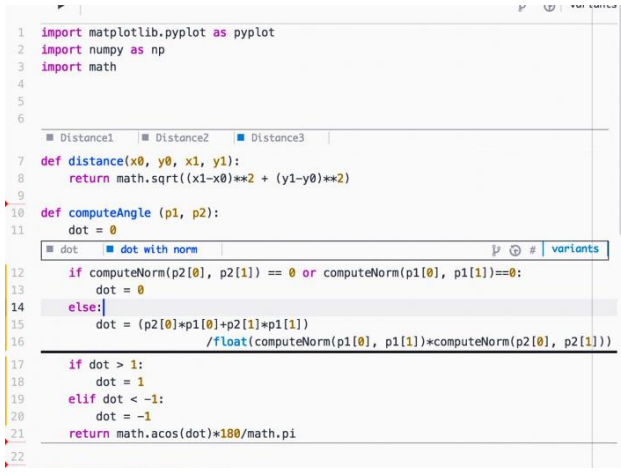


Figure 2.8. Variolite extends the traditional code editor to facilitate the exploration of alternative code pieces in arbitrary sizes [8].

notebooks enable diverse groups of data workers to explore alternative ideas and analysis paths using the combination of code, text narratives, and data visualizations. Yet, the explored alternatives are often messy and intertwined inside one or more notebooks, making it hard to manage and retrieve [8, 9, 12, 26, 27]. Kery et al. 's Variolite [8], as shown in figure 2.8, attempts to generalize interactions for dealing with alternatives in the notebook interfaces. Users can create alternative code blocks, test different combinations of alternatives, and compare them by interchanging the running versions. Head et al.'s Code Gathering [27] deploys a "post-hoc mess management" strategy to help users get a clean analysis among all explored alternatives. By selecting the desired analytic results, Code Gathering automatically generates ordered, minimal subsets of code that produce them. Kery et al.'s Verdant system enables data workers to trace, replay, and compare many alternative versions of code and non-code artifacts, in both the notebook and cell level [9, 65].

Computational notebooks are initially designed to prompt the capture of analytic reasoning by treating analytic narratives as first class citizen as source code. The Variolite, Code Gathering, and Verdant systems facilitate the management of code and version alternatives, but they do not explicitly consider alternatives in the cognitive level.

Based upon and extending the concept of iterate programming [64], Wood et al. propose LitVis [49] (literate visualization) to capture alternative visualization designs as well as the associated reasoning process. LitVis composes a model of design exposition and a set of narrative schemas. Analyts need to integrate code for implementing data visualizations with descriptions of their design choices. Mathisen et al.'s InsideInsights [66] aims to capture the hierarchical and branching nature of data analysis. Users structure their findings into hierarchical stories and link the analysis states which support them. The final product is a dynamic and interactive report that can be shared. Both LitVis and InsideInsights support the management of alternatives in

the cognition level and tie them together with the related artifact alternatives.

The reviewed systems and techniques illustrate a variety of ways to better support alternatives. They tend to tackle a subset of alternatives and practices among the overall sensemaking process. Most of them provide solutions for managing artifact alternatives, such as code, parameters, and algorithms. Some also provide support for recording and managing cognitive alternatives like findings and hypotheses.

2.2.2 Systems for sensemaking

In this section, we look at systems that support the overall sensemaking processes. Instead of focusing on specific types of alternatives, these systems concentrate on supporting the various kinds of analytic tasks and activities across the sensemaking process.

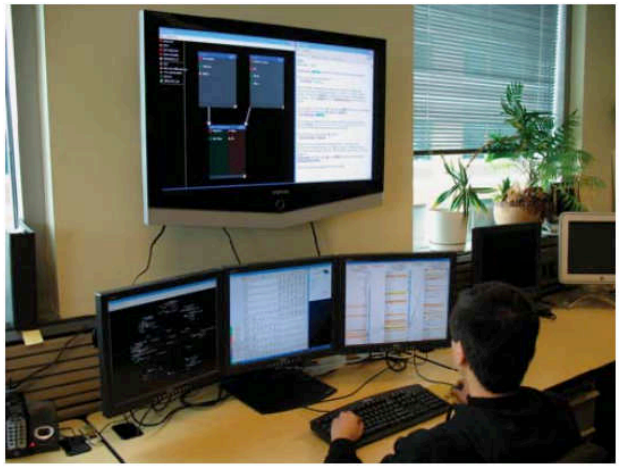


Figure 2.9. Multiple views in Jigsaw system to facilitate the sensemaking processes of investigative analysis [47].

Investigative analysts need to make sense of large collections of data, often in text documents, to connect the dots and discover hidden truths [38, 67–69]. Stasko et al. [47] invent a visual analytic system Jigsaw for investigative analysis, as shown in figure 2.9. Drawing upon Pirroli and Card's sensemaking model [1], Jigsaw tackles on both loops of information foraging and sensemaking. Jigsaw provides multiple distinct visualizations. These different views are alternatives for a same document collection. They enable different analytic methods, perspectives, and data representations, which are complementary to each other in the analysis. Besides alternative visualizations, Jigsaw also enables analysts to organize evidence and build hypotheses. The system's primary focus is on "displaying the connections between entities across the documents" [47]. It allows analysts to establish different types of links inside a particular view, and also integrate links between multiple views to facilitate the navigation.

Another system Aruvi, designed by Shrinivasan et al., takes emphasis on the analytical reasoning process during interactive data exploration [70]. Besides the common data view where analysts can interact with various kinds of data representations, Aruvi further introduces a navigation view and a knowledge view. The knowledge view enables users to record their interpretations, findings, and hypotheses. Aruvi automatically captures the exploration states. Users can revisit a state and branch out alternative analysis by interacting with the navigation view. They can also establish links between the analysis artifacts in the knowledge view and a visualization state in the navigation view to facilitate the validation of a finding or a hypothesis.

In the user study of Aruvi, Shrinivasan et al. find that analysts highly value the knowledge view and the links between analysis artifacts and specific visualization states. These features help analysts to clearly see the analysis process and also improve their quality of results. Yet, they find that to compare alternative analysis, users need to keep switching between the alternative states and views. Some users sug-

gest that a side-by-side comparison would be a more effective way [70].

Past researches reveal the benefits of combining heterogeneous displays to enable "space to think" and to support different types of tasks in sensemaking activities [71–74]. The size, mobility, and inherent affordances of diverse displays all play unique and crucial roles in supporting different analytic tasks within the sensemaking process. They together form a *display ecology* - "a system of displays that engage the entire workflow of a task to better assist analysts in achieving their desired outcomes" as defined by chung et al. [73].

To facilitate searching, organizing, and synthesizing relevant information across multiple displays, the system SAViL deploys direct visual connections among different types of objects on multiple heterogeneous displays [74]. Chung et al. find that these visual links help analysts better perform information foraging tasks, and maintain awareness of connections between documents. SAViL also facilitate leveraging multiple displays and creating semantic layers over

Figure 2.10. SAViL provides direct visual links among different types of objects on multiple heterogeneous displays [74].

them [74].

These sensemaking systems share some commonalities: they all provide multiple views to enable users explore different data representations and examine multiple analytic perspectives. They also provide various types of links between these views and analysis artifacts to facilitate sensemaking. Our study on alternatives in sensemaking also reveal data workers' needs of multiple analytic views and the links between different kinds of alternative artifacts. Our work in chapter 4 explores the alternative views in qualitative sensemaking with affinity diagramming technique. Our proof-of-concept prototype ADQDA explores different types of implicit links to facilitate tracking and synchronizing related artifacts across these views. Our work in chapter 5 explores different types of explicit links to help manage alternatives in the context of reuse.

2.3 Designing Software

In this section, we provide a brief review on the history of designing software which largely inspire and shape the way we handle information today. We then present several emerging concepts and theories on building more flexible software tools which greatly influence our research on designing tools for alternatives in sensemaking.

2.3.1 A brief history

In the article "As We May Think" published in 1945, Vannevar Bush envisioned a device called the Memex (as "memory extender") to store massive amounts of information and knowledge in various kinds of forms, as illustrated in figure 2.11. The Memex would allow people to efficiently and flexibly access, retrieve, and interact with information

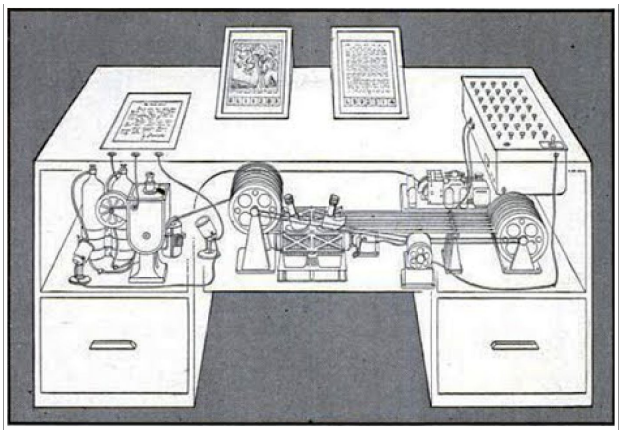


Figure 2.11. Original illustration of the Memex, a theoretical analogue computer described by Vannevar Bush [75].

in different kinds of forms. It is an "enlarged intimate supplement to one's memory" [75]. Memex is the pioneer in recognizing and envisioning the computer as an empowering tool to externalize, organize, access, and share knowledge. Even if never built, this vision has deeply affected how we access and interact with information today.

In the 1960s, inspired by Vannevar Bush's vision, Ted Nelson coined the notions of hypertext — "written or pictorial material interconnected in an associative fashion, consisting of units of information retrieved by automated links, best read at a screen" [76]. Instead of emulating physical paper to handle information in a digital world, Ted Nelson redefined the structure and model of texts in a digital context. Information in a variety types of forms and across different kinds of places is supposed to be flexibly organized to align with human thought and creativity. These interconnected documents are powerful external aids to human memory and amplify our ability to manage complex information resources. The central concept is in this

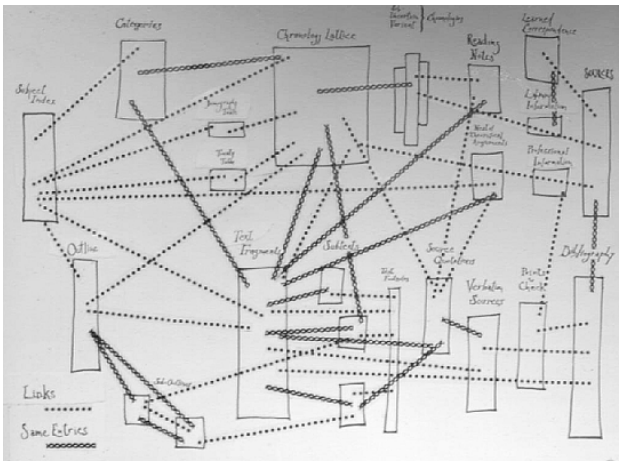


Figure 2.12. Hypertext as schematically illustrated in 1965 [76].

vision is the hyperlink. As illustrated in figure 2.12, hyperlinks are the references to other information material which facilitate the immediate access and navigation among large troves of interconnected documents.

The concept of hyperlink is widely applied in today's information system. For example, the omnipresent World Wide Web, which was invented by Tim Berners-Lee in 1989, is built upon hyperlinks among web pages. Whereas another essential component of Ted Nelson's vision has been omitted in many of existing tools: "transclusion". Transclusion is defined as "the same content knowably in more than one place" [76], or "reuse with original context available, through embedded shared instancing" [77]. It can be viewed as a mechanism to compose documents. Through dynamic embedding of a same content within multiple documents, changes to the embedded content are reflected in all the embedding documents in real time. They are not separate copies, but rather references to the same underlying content. Our work on "computational transclusion" is largely inspired by Ted Nelson's transclusion concept and its extension of "software transclusion" [14, 78]. A more detailed discussion on transclusion is presented on transclusion in chapter 5.

Another great invention in the history is the HyperCard system, created by Bill Atkinson in 1985. It was a popular form to interact with information in the pre-Web era. Bill Atkinson describes HyperCard as "a software erector set that lets non-programmers put together interactive information" [79]. Inspired by Ted Nelson's Hypermedia concept, HyperCard are a stack of cards interconnected by hyperlinks. A card contains both information and interaction. It can hold various objects, such as texts, graphics, icons, and interactive buttons. Users can easily build their own interactive systems by using a combination of various kinds of cards and linking them based on how things are associated. They can also use the built-in programming language called HyperTalk to customize new cards, or to redefine and extend existing

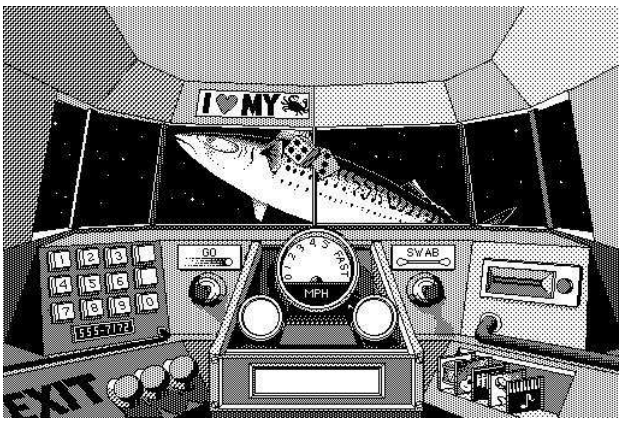


Figure 2.13. A game called Cosmic Osmo implemented in hyperCard.

ones.

There exist many interactive systems built with HyperCard [79,80], from simple word processing applications to complex ones like business management applications. Figure 2.13 shows a game created with HyperCard. HyperCard has been proved to be easy to learn and use for people from various domains at diverse age range [79]. For example, primary school kids could use HyperCard for class assignments, or even build their own systems. Myer et al. evaluate tools using "threshold and ceiling": threshold is "how difficult it is to learn how to use the system", and ceiling is "how much can be done using the system" [81]. HyperCard can thus be viewed as a successful tool due to its high ceiling and low threshold. It is easy for non-programmers to start. While the users learn more about HyperCard, they can customize it to build complex systems for various kinds of needs.

These pioneering systems and concepts all recognize the natural complexity of human activities. They envision digital tools can help flexibly organize information and knowledge as the way we think and use them. Despite this vision, our ability to handle information is

still largely constrained in today's digital world. Current tools largely adopt the application model, which tends to encapsulate information in silos, impose how interactions should carry out, and limit what users can do with pre-defined system features. It is hard to extend or combine these tools to suit users' situated needs. In this dissertation, we explore how to build digital tools that can be flexibly appropriated to handle alternatives in the dynamic and iterative process of sensemaking.

2.3.2 Concepts & theories on designing flexible tools

More recently, many researches re-question the largely adopted application model and redefine the relationship between tools and human activities. In this section, we demonstrate the concepts and theories that offer new opportunities on designing digital tools.

Based on how people manipulate objects in the physical world, Beaudouin-Lafon proposes a new interaction model of "instrumental interaction" [82]. To interact with a digital object, "the user acts on the instrument, which transforms the user's actions into commands affecting relevant target domain objects." Unlike direct manipulation [83], instrumental interaction decouples the object of interest and the instrument used to manipulate them. For example, scrolling a digital document (the domain object) is mediated by dragging a scrollbar (the instrument).

In the physical world, an instrument can be used in a flexible way beyond its original context. The instrument can be appropriated to various kinds of needs according to its affordances. For example, chopsticks are not only limited to getting food but also can be used to roll up hair in lack of the hair clip. However, in the application model,

tools or instruments are tightly tied to objects. It is nearly impossible for end-users to apply them differently or to move them between applications. To leverage the instrumental interaction concept into applications, Beaudouin-Lafon further summarises three principles for designing visual interfaces: reification, polymorphism, and reuse [84]. His work has prompted the redesign of software interface in various domains including graphic design and data analysis, e.g. [84–86].

Another attempt to challenge the application model is inspired by the field of hypermedia. Instead of treating software as static systems isolated from each other. Hypermedia, since its creation, values the flexibility and complexity of human activities. It emphasizes the collaboration among users, the distribution across places, and the constantly changing procedures in real work [87]. Reflecting on what software can learn from hypermedia, Klokmose et al. introduce the "shareable dynamic media" as an alternative model [14]. Shareable dynamic media are collections of information substrates, which embody content, computation, and interaction. Substrates can be easily shared among users and be distributed across diverse devices and platforms. Substrates can evolve over time and be dynamically modified and extended from within. Like HyperCard, there is no clear separation between authoring and consuming the software artifact. It can act as "documents or applications or a mix of the two according to the context".

Klokmose et al. further provide an experimental implementation of this model - the Webstrates. Changes to each webstrate (web document) are synchronized through all its instances so that a webstrate can be shared across people and devices. Webstrates can be composed using the transclusion mechanism, embedding one webstrate within another. As discussed earlier, transclusion enables dynamic sharing rather than just copy the content. Substrates can be composed in various ways. For example, one substrate can define the behaviors of another or give structure to another. The generative power

of Webstrates has been proven by the following creation of research prototypes in a variety of domains, such as Codestrates for computational notebooks [88], Videostrates for programmable video manipulation [89], and Vistrates for visual analytics [90]. Our prove-of-concept prototype ADQDA (chapter 4) and our sandbox for exploring "computational transclusion" (chapter 5) are all built upon the Webstrates platform.

As technology evolves, the way to think about the human-tool relationship has been evolving as well. Interactive devices, such as mobile phones, tablets, laptops, tabletops, and wall-sized displays, are becoming increasingly accessible and pervasive in our daily life. It is common for both individuals and organizations to own a combination of devise displays. These different type of devices have their affordances and technical capabilities, in terms of their forms, sizes, input modality, and input accuracy. They all play unique and crucial roles in supporting different analytic tasks within the sensemaking process.

The activity theory acts as one of the most fundamental theoretical approaches in the design and evaluation of many interactive systems [91, 92]. The introduction of activity theory shifts the attention of designing tools for certain tasks to understand humans, activities, and the larger contexts as mutual factors where one can influence and determine another. Bødker and Klokmose [15, 16] further developed the Human-Artifact Model. They emphasize studying heterogeneous devices as part of an artifact ecology where "one can dynamically interplay with others and with users' web of activities". Many studies explore the combination of various displays to support sensemaking activities, both for individuals and for groups [72, 90, 93, 94].

This dissertation demonstrates how these concepts and theories can be applied in building more flexible tools to support exploring and managing alternatives throughout the sensemaking process.

Part I

Understanding Alternatives in Sensemaking Practices

CHAPTER 3

A characterization on Alternatives

In this chapter, we aim to answer the first research question - **how do alternatives fit within the sensemaking process?** We present our study on semi-structured interviews with twelve data workers from a variety of disciplines. We focus on the different kinds and meanings of alternatives for our data workers, and how these alternatives fit into their overall sensemaking practices [1].

Section 3.1 presents how these interviews were conducted and the types of analysis performed on the collected data. Section 3.2 explains why data workers explore alternatives and the triggers and barriers to doing so. Section 3.3 demonstrates a framework of alternatives to help describe and reason how data workers consider alternatives in their analyses, and how tool designers might create tools to better support them. Section 3.4 discusses our data workers' strategies to cope with alternatives. Section 3.5 shows how this framework can be used to describe and analyze analytic tools through the lens of alternatives. Section 3.6 compares the proposed framework with existing notions of alternatives and discusses the study limitations. Section 3.7 ends this chapter by discussing remained questions for future work.

[1] This chapter draws heavily upon our TVCG publication [5].

3.1 Study Design

We conducted semi-structured interviews with twelve data workers with different types of expertise. Through these interviews, we aim to understand the following questions in particular:

Q1 To what extent do data workers explicitly consider alternatives in their workflow?

Q2 When do they consider alternatives? Are there specific triggers & barriers for exploring alternatives?

Q3 What types of alternatives do they consider?

Q4 What strategies do they deploy to cope with alternatives?

3.1.1 Participants

We recruited twelve participants who perform data work daily. Ages ranged from 22 to 63; three identified as female, nine as male. Participants were recruited by email through our social and professional networks. They come from a variety of disciplines: four participants work in industry, in sectors such as marketing, medical data modeling, and cost management. Eight participants work in research domains including Human-Computer Interaction (HCI), humanitarian research, and biology (see Table 3.1). They held a range of job titles from "researcher" to "assistant project manager," "cost manager," "consultant," and "data analyst." Most (nine) are junior in terms of experience in dealing with their specific domain of activity (1–5 years). Three have nine or more years of experience. While all participants have experience in data work, they have diverse levels of domain and computational expertise. In Kandel's categorization [4], which focuses

P#	Org	Domain	Experience with data	Analysis Tools
1(H)	E	Cost management	1.5 years	Excel, VBA
2(H)	E	Medical 3D modeling	3 years	Visual Studio, OpenGL
3(S)	R	Visual tracking	5 years	Matlab, Python
4(A)	E	Project management	2 years	Excel
5(S)	R	InfoVis research	1.5 years	D3.js, Tableau
6(S)	R	Optimization	9 years	Python
7(S)	R	HCI research	2.5 years	JupyterNotebook, PyCharm
8(H)	E	Food delivery	2 years	Jira, Python, Metabase
9(H)	R	Topological research	10 years	TTK (customized)
10(A)	R	Humanitarian research	3 years	Excel, Tableau
11(S)	R	Ecosystem services	5 years	QuantumGIS, Matlab
12(A)	R	Education	30 years	Paper, digital folders

Table 3.1. Interview participants by domain of expertise. Brackets after IDs refer to S: Scripter, H: Hacker, A: Application User. Org is short for Organization, where R: research, E: enterprise.

on computational tool capabilities, four can be classified as hackers, five as scripters, and three as application users.

3.1.2 Setting and procedure

The study was conducted through semi-structured interviews. Interviews were conducted in English, French, or Chinese, depending on the participant's preference. Seven interviews were conducted in the participant's primary workplace. Five were conducted by videocon-

ference, including the participant's computer desktop, for practical or workplace security restrictions.

Questions were open-ended to encourage participants to describe their experiences in their own words. Each interview consisted of three phases, as demonstrates in figure 3.1. The first phase aimed at understanding the participants' general work context (goals, data, methods, tools, role in the team, etc.) and workflows. In the second phase, participants were asked to describe their workflow in more detail and to walk us through a recent analysis, step-by-step. (Participants were asked to prepare this analysis walkthrough during the recruitment phase.)

During recruitment and these first two phases, we did not reveal to participants that this work focussed on alternatives. Participants were told that we were interested in their reasoning and sensemaking process and in understanding their workflows. Our goal was to understand whether and how participants think about alternatives throughout their sensemaking process [Q1]. We did ask questions of the form "Did you think of other ways to do this?" or "Did you generate any other [artifacts, e. g., models]?" to encourage participants to describe any tacit alternatives they may have explored without explicitly orienting their thinking toward alternatives.

In the third phase, we revealed to participants that the main goal of this study was to understand the role of alternatives in their work. However, we attempted not to give any concrete definition of alternatives so as to reveal participants' own interpretations. If pressed, we gave vague definitions such as "different possibilities you considered or tried out" or grounded them in the descriptions participants had already shared: For example, if a participant mentioned having tried several machine learning models, we might ask, "Do you think the different models that you explored are alternatives?" and if so, "Are there any other kinds of alternatives?" The goal of this third phase was to understand what alternatives meant to participants and to fo-

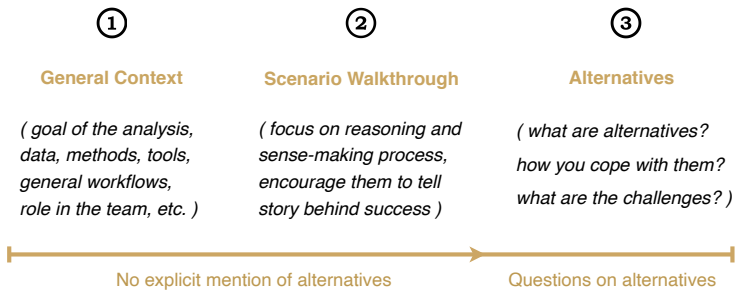


Figure 3.1. Our interviews contain three main phases: phase 1 to understand the participants' general work context; phase 2 to their sensemaking workflow, analytic reasoning process; and phase 3 to understand the particular role of alternatives in their sensemaking practices.

cus more deeply on how they specifically managed them throughout their workflows [Q2–4].

3.1.3 Data collection and analysis

All interviews were recorded, generating 827 minutes of recordings in total. Recordings were transcribed into 73 895 words. Two authors individually open-coded the transcripts, highlighting interesting portions of text or key ideas. We then collectively cross-checked these extracts to reach agreement, yielding 585 unique extracts (which we call notes or snippets). We then used an iterative coding approach based on grounded theory [95] and digital affinity diagramming. We iteratively grouped notes based on common themes using a custom appli-

cation on a wall-sized multi-touch display. Multiple analyses grouped snippets according to different facets drawn from the initial study goals (e. g., types of alternatives, tasks around alternatives, strategies for dealing with alternatives, triggers barriers) as well as those that emerged during the analysis phase (e. g., the role of expertise, collaboration). We further created higher-level diagrams to identify themes amongst the different categories (or clusters) of direct observations. This analysis yielded a collection of the various topics and themes extracted directly from the participants' interviews.

We also conducted a second analysis that aimed at describing the overall workflows the participants followed. This analysis consisted of four steps: 1) we analyzed the transcript data and drew a high-level process diagram of the participant's workflow, figure 3.2 demonstrates some of them,; 2) each diagram was submitted to the corresponding participant for validation; 3) we analyzed the validated workflow diagrams to better understand when and how participants explore alternatives and the link between the different stages of analysis.

Figure 3.2. Diagrams on the overall workflows for different participants.

3.2 Why Data Workers Explore Alternatives

In this section, we present the reasons why data workers consider and explore alternatives in their sensemaking processes and the triggers and barriers to dealing with alternatives.

3.2.1 General reasons

We found four main reasons why participants valued exploring alternatives during sensemaking: to clarify goals and processes, to delay decision making, to build confidence in a solution, and to partition the sensemaking workload.

The very nature of data work and sensemaking is messy, with partial hypotheses, loosely defined goals, and incomplete methods to address them, particularly at the start of an analysis. Analysts must instantiate multiple ideas and iteratively update them to better define the problem and a reasonable path to solve it. P9 describes how he needs to consider multiple visualisation and analysis methods: *"One of the things that is specific to our research is that [data providers] are not sure what they want. So if there are several options there, possible or interesting for them, they need to be able to switch between them."*

When different exploration paths are equally viable, data workers may delay decision making by considering as many of them as possible. In such cases, deciding which analysis path is the most pertinent does not have to be taken at the start of the exploration. As P6 observes, *"For machine learning, …it's correct that there is no single technique that is the best for every problem. …so you try them all."*

Other participants considered alternatives when they felt their current alternative was just "not good enough." By considering alternatives, they would either find a better solution or build their confidence in the current solution. This lack of confidence can also originate from the analyst's lack of experience, as expressed by P8: *"If you have more experience, maybe you will know, for this type of data that method will always be bad, so you don't need to try it, but this needs more experience. So for now, we just try all the possibilities that we can."*

Often collaboration with other data workers led our participants to consider new alternatives. For example, a colleague might suggest a new direction to consider: *"I tried with some evolutionary algorithms I coded myself with Matlab. Then I worked with [a colleague], he proposed to use some packages for evolutionary algorithms"* (P11). In this way, participants would partition the sensemaking workload.

3.2.2 Triggers & barriers

We coded interview transcripts to look for specific triggers or barriers to considering alternatives. We found four main triggers evoked by our participants that led them to consider alternatives and six main types of barriers that reduced the likelihood of doing that. Table 3.2 summarizes these findings.

Triggers It is common for a data worker to run into a dead-end during analysis or to find evidence counter to the current hypothesis. These events act as triggers to finding alternate solutions. For example P7 initially wanted to model his data with machine learning algorithms, but he failed to find an appropriate solution and decided to manually annotate the data instead: *"All of these things [ML algorithms] we explored, they don't work as we would like them to. Because*

Triggers of Exploring alternatives	Barriers of Exploring alternatives
- confront a dead-end	- limited availability
- realize limitation	- too much learning effort
- cognitive leap	- time limitation
- collaboration	- cognition bias
	- lack of expertise
	- collaboration

Table 3.2. Triggers and barriers to alternative exploration.

in the end, they are very prone to errors, outliers…. In the end, we decided to annotate our data manually instead" (P7).

When data workers recognize a limitation or deficiency in the current solution, they may consider new alternatives. For example, when P10 realized that online data scraping was not providing the rich information she needs, she recognized the *"…need to go and collect data with refugees themselves."*

When inspiration strikes, a data worker's cognitive leap might trigger a new collection of alternatives to explore. For example, P9 made the cognitive leap that a less-realistic orientation of the data would lead to easier interpretation of the data: *"everybody showed it the real way, …but I figured that to see the geometry of the thing, it was better to flip it and we would have a better view."*

Finally, collaboration events can trigger the pursuit of alternatives. Different people in a team may have diverse background and expertise, which can bring new points of view on how to solve a problem. For example, P11 used a manually-created model until his collaborator suggested trying out machine-learning approaches. Similarly, P5 changed his coding environment to Jupyter Notebooks to facilitate communication with his mentor.

Barriers To identify barriers to exploring alternatives, we included questions of the form, "What stopped you from trying alternatives?" and "You just mentioned you considered other options, did you pursue them? (If not) why not?" Participants also revealed such barriers through their own descriptions about the difficulties or problems encountered. We coded transcripts for these kinds of barriers, yielding six common types of barriers amongst our participants.

Limited data and tool availability often prevented participants from instantiating their ideas. For example, when P11 wanted to enrich his model, he could not because *"…at that moment, I didn't have the data"* (P11). Similarly, when he wanted to perform a different analysis, he found that his open-source tool did not have that feature implemented. Just as financial constraints limited his tool availability, time constraints prevented participants from exploring potential alternatives. As P5 succinctly described: *"we didn't have that much time."* Similarly, too much learning effort hindered efforts to attempt new solutions, as when P10 was hesitant to try a new tool *"because there are not that many tools that you can learn how to use without training."*

A lack of expertise prevented some participants from judging or evaluating alternative approaches. For example, P4 abandoned automating her analysis pipeline using Excel macros because she realized she did not sufficiently understand the VBA language used to write them. More generally, such lack of expertise often acted as a counterforce to recognizing, evaluating, or pursuing alternatives.

While a lack of expertise can act both as a reason why participants considered alternatives and as a barrier to considering alternatives, we do not include it as a trigger since we did not see it act as a specific catalyst to considering alternatives.

Cognitive biases tended to guide participants to reuse alternatives that they had considered in previous projects, thus preventing them from considering new alternatives. Conversely, participants sometimes

justified excluding alternatives because they had found them not to work in a past project. This tendency has been recognized as a bias that can lead intelligence analysts and policy makers to make poor decisions [13].

Finally, collaboration effects can act not only as a trigger but also as a barrier to exploring alternatives. In order to maintain consistency within an organization, improve communication efficiency, and facilitate management, data workers often follow certain predefined pipelines and use a set of predetermined tools. Introducing a new tool, analysis, or model not shared by collaborators introduces a collaboration cost, even if individually it may be worthwhile to pursue. For example, P1 continues to perform analyses using Excel: *"Personally, I would like to try Python, but if we use Python, it's kind of difficult for others to manipulate the interface or the code inside…. The people who need to see are not developers, but likely managers or directors."*

3.3 A Framework of Alternatives

Rather than treating the exploration of alternatives as a separate stage of the data analysis pipeline [53] or as a separate activity that ties reflection with analysis during research programming workflows [52], our participants revealed that alternatives and their impact on data analysis activities are ubiquitous.

Just what constitutes an alternative, however, can be ambiguous. Alternatives could encompass multiple iterative versions of the same artifact or refined versions of a given hypothesis or altogether distinct methods to analyze data. Intuitively, however, these different kinds of alternatives seem fundamentally different.

To clarify these multiple notions and roles of alternatives, we structure our observations in terms of participants' degree of attention

(section 3.3.1) and around the abstraction level of alternatives (section 3.3.2). We further characterize alternatives based on the high-level processes performed around it (section 3.3.3). These three primary dimensions: 1) degree of attention, 2) level of abstraction, and 3) processes on alternatives, together, form our proposed framework for understanding alternatives within the sensemaking process. We explain them in details in the following sections.

3.3.1 Degree of attention

According to the amount of attention paid by the analyst, we classify alternatives as: *Multiples* are analytic artifacts generated or explored during the sensemaking process, such as code versions, annotations in a notebook, or visualisation trails. They exist as direct entities evoked during ideation and exploration, drawn from the larger universe of possibilities. Data workers are aware of them, but no special attention is put on any single one. *Options* are drawn from these multiples as they are brought into the analyst's consciousness for closer examination or deeper consideration, such as when the analyst narrows down focus to a smaller set of potential analyses drawn from the literature. *Choices* are options that are actively pursued in analysis.

The distinction between multiples, options, and choices thus depends on the amount of attention paid by the analyst. In the remainder of this dissertation, we refer to this as the M-O-C model. We use the term alternative to refer to the union of multiples, options, and choices, regardless of the analyst's degree of attention.

Fluidity of Attention At different phases throughout the analytic process, a given alternative might be a multiple, then an option, a choice, and end up a mere multiple. At any given time, there may be any number of each of these types of alternative. Multiples may

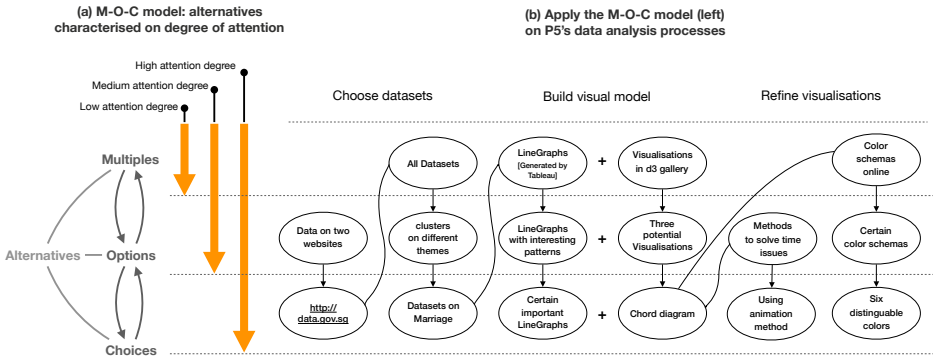


Figure 3.3. (a), left: Alternatives characterized based on the data worker's degree of attention. (b) A workflow showing alternatives for P5. Alternatives are grouped along the y-axis according to their degree of attention. Along the x-axis are different stages along the analysis. We can see different alternatives migrate between different attention degrees at different stages of the analysis as their frames of reference evolve.

be generated as an initial step in exploration to enlarge the possible solution space. For example, P2 collected around 50 papers describing potential methods to use in modeling organs. We can think of these articles as multiples drawn from the larger universe of potential methods. P2 then skimmed through these multiple methods to identify two options to pursue. At first, she implemented one technique, making a choice. Later, she implemented the other one. After comparing these two options and analyzing the trade-offs, she chose to continue her project with the first technique.

The distinction between multiples, options, and choice can be fuzzy as alternatives fluidly move both up and down among these three levels. A multiple can turn into an option or even a choice along the sensemaking process. On the other hand, a choice can become again

an option or multiple when the analyst gives consideration to other alternate options.

These definitions thus depend on the analyst's "frame of reference." For example, P5 collected multiple potential datasets to visualize. During this process, the different datasets are active alternatives that move up and down along the three-levels. Once he had selected a given dataset and moved on to considering what type of visualisation to use, the consideration of alternative datasets faded into the background. As such, under his new frame of reference, the dataset becomes a fixed choice and different visualisation types become the current active alternatives, as shown in Figure 3.3.

Moreover, exploration can start at any of these three levels, depending on the strategy involved (section 3.4). In a breadth-first, or "shotgun," strategy, multiple multiples are selected first to identify and focus on as options. In a depth-first, or "sniper," strategy, a data worker may make an immediate choice to focus on a given path until some trigger motivates the exploration of other alternatives (section 3.2.2).

3.3.2 Level of abstraction

We performed a bottom-up analysis of interview transcripts and identify ten types of alternative considered by our participants: hypothesis, mental model, interpretation, data, model, representation, tool, code, parameter, and method. Most of these alternative types, such as methods, were explicitly identified by participants. Others, such as mental models, were not explicitly mentioned by participants but were revealed in the course of their interviews. We group these types of alternative into three levels of abstraction based on their role in the analytic process:

Cognitive alternatives pertain to human reasoning, including alternative *hypotheses, mental models*, and *interpretations.*

Artifact alternatives relate to different types of concrete artifacts, such as alternative *data, models, representations,* or *tools.*

Execution alternatives refer to how data workers carry out an activity, including *method, code* and *parameter* alternatives.

While these ten types of alternative are not intended to comprise an exhaustive taxonomy of the different kinds of alternative that may exist, we do expect any other alternative types to fit within these three layers of abstraction.

Cognitive Alternatives pertain to data workers' evolving mental processes throughout the different iterations of their analysis. Data workers may develop alternate interpretations, adjust their mental models, and formulate new hypotheses. For example, when analyzing apps built for migrants, P10 considered alternative hypotheses about their production rate: *"When these apps were built, is there kind of a pattern there [e. g. following refugee crises], or it's like 20 every year?"* P7 expressed alternate interpretations of a finding on people's behaviors in log data: *"This is also an interesting result, because this may indicate that people actually started considering to switch… before they actually did, or maybe this finding is just noise… we need to analyze more data to see."*

Artifact Alternatives involve the different *things* involved in analysis, such as data, models, representations, or tools: which data sets are to be analysed, what kinds of models to use on the data, what visual designs are best-suited for presenting results, or which analysis tools to use to perform these tasks. Each of these types of artifact

alternative describes a collection of different specific kinds of alternative. For example, data alternatives include all alternatives pertaining to data, such as using different data sets, data providers, dimensions, or even different values. Model alternatives involve, for example, different machine learning algorithms, statistical models, mathematical models, or other ways of structuring the interpretation of data. They may be machine-centric (e. g., neural networks or random forests), human-centric (e. g., manual annotation or hand-crafted models), or even hybrids of the two. Representation alternatives pertain to different ways of depicting the various artifacts in the sensemaking process. Finally, tool alternatives include the different analytic tools that can be used, including software tools (e. g., Excel, Tableau, home-grown libraries) or analyses.

Execution Alternatives involve the means by which the data worker carries out an activity. This can include method, code, and parameter alternatives. For example, to obtain data from a given data source, P5 would choose between directly downloading data or using a data scraper. Six participants described tuning parameters of their models. Another six participants described using or creating different versions of code. In all of these cases, execution alternatives describe the means by which to accomplish a particular task.

The abstraction level of an alternative can further depend on its degree of attention. For example, when tuning a model, different parameter alternatives fall in the execution level: they pertain to the means by which the data worker carries out the activity. At some point, however, different parameter configurations of the same model might themselves become distinct artifacts, as when P3 compares different computer vision solutions. As such, the type of an alternative does not appear to be a concept intrinsic to the alternative; rather, it seems to at least partially depend on the data worker's frame of reference.

Interdependency among alternatives Alternatives arose throughout the data analysis processes and at different abstraction levels. They are often interdependent: changing between alternatives at one level can influence the others. For example, changing hypotheses may lead to building alternate models, running scripts with different parameters as input, or even changing the underlying methods. Similarly, choosing new parameter values can result in new findings that lead to alternate mental models and hypotheses.

This interwoven nature among different types of alternatives corresponds to the complex, intertwining processes of data analysis activities. The analytic results or insights generated are also sensitive to these alternatives at different layers—how data are collected, what features are extracted, what methods are used to produce them. Any alternative in the analysis pipeline can lead to different results. As such, different types of alternatives cannot be treated as wholly independent.

3.3.3 High-level processes on alternatives

In this section, we describe five high-level processes, or abstract tasks, that our participants engaged in when handling alternatives: *Generate, Update, Reduce, Reason,* and *Manage* (see Figure 3.4). Each process operates on alternatives. The first three differ in the shape of how they affect the cardinality of the alternative space. The other two do not directly produce alternatives, but operate on them. As we will see below, Generate, Reduce, and Manage apply to alternatives at all three levels of attention, whereas Update and Reason seem to apply only to options and choices. Each process happens at all three layers of abstraction.

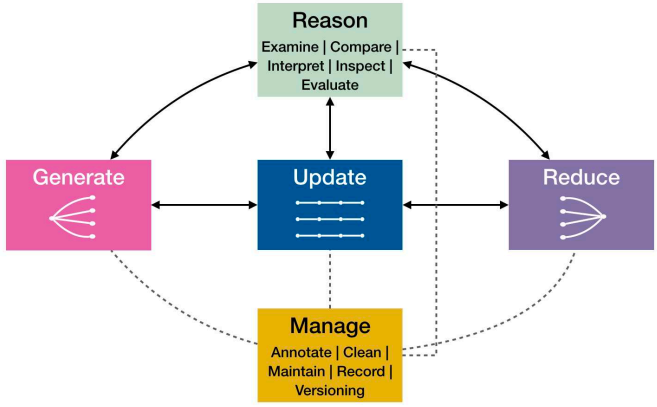


Figure 3.4. The five high-level processes around alternatives: Generate, Update, and Reduce are distinguished by their different "shapes" within the alternative space (divergent, parallel, convergent). Reason and Manage do not directly produce alternatives, but operate on them.

Generate expands the alternatives space. It encompasses tasks such as formulating new hypotheses, finding a new visual design (e. g., from the D3 example gallery), identifying a new data collection method, combining parts of several methods to make a new one, or creating a new code branch. All participants generated alternatives during their analysis, either in a preparatory phase prior to exploration or during the analysis as situated actions [96] in response to new task requirements.

For example, prior to analysis, P3 performed a literature review and consulted with colleagues to select several promising approaches to improve a visual tracking application. During analysis, however, he found that the results were still not satisfactory, so he investigated a different family of methods.

Update refines an existing set of options and choices. At first glance, this process might seem to be contradictory: updating an alternative typically produces a new alternative. For example, when tuning (updating) an image saturation parameter, this refinement yields a new (tuned) alternative in addition to the original. In most cases, however, either the original alternative or its update becomes a mere *multiple*: it no longer receives any attention. As such, while the number of alternatives may increase, the number of options or choices remains constant. In other words, the difference between updating an alternative and generating a new alternative largely has to do with whether the user *intends* to come back to both alternatives.

For example, P10 who studies the use of technology by migrants, regularly receives new, updated, data. Although each new update provides more up-to-date information, P10 continues to store all the different versions: *"I stored all of them…. For now, I don't need to look at the previous versions, but I feel safe that they are there. I can reach them when needed."*

Reduce decreases the number of alternatives and represents the convergent phase of analysis. It includes tasks such as filtering, excluding data sets (e. g., those identified as poor in quality), and merging multiple code versions. These operations can be manual, automatic, or semi-automatic. For example, P6 works on a multi-objective problem where he needs to present the set of alternative optimal solutions to a decision maker. He deploys a two-level filtering approach, where he first uses a multi-objective optimization algorithm to automatically select a subset of optimal solutions (the Pareto front), then the decision maker manually selects the best solution(s) according to his preferences.

Reason encapsulates all tasks that result in the generation of thoughts, insights, or decisions on alternatives. We identified reasoning tasks such as compare, inspect, examine, interpret, understand, and evaluate. The process of reasoning is often a combination of a series of tasks. For example, P6 created a table to compare the performance of different classifiers and regressors, revealing that decision trees tend to produce better results. Reflecting on his past experience and domain knowlege, he could confirm this finding and share it with colleagues.

Manage refers to the operations that structure, organize and maintain alternatives for later reference or reuse, such as annotating (e. g., comments on pros & cons of alternate methods), versioning pieces of code, and writing scripts to remove multiples that are no longer needed.

3.4 Strategies to Cope with Alternatives

In this section, we share various strategies deployed by our participants, structured by process. We do not try to give an exhaustive categorization by enumerating all possible strategies. Instead, we provide descriptions based on recurring strategies observed among participants.

The two most common strategies for exploring alternatives could be called *depth-first* (or "sniper") and *breadth-first* ("shotgun"). In the depth-first strategy, the participant would concentrate on a given, promising alternative (choice), considering other alternatives only when needed (such as when recognizing a dead-end). In the breadth-first strategy, the analyst would generate many multiples or options to pur-

sue before eventually focusing down on one or two promising choices. For example, P6 would throw "all" of the machine learning algorithms in his tool chest at a particular problem (*"So you try them all"*) before winnowing them down to those that perform best.

3.4.1 Generating, updating and reducing alternatives

In each of these three processes, participants would often *consult external resources*; *draw on past experience or domain knowledge*; or use *trial and error*. Participants frequently consulted external sources to generate new methods, update current models, or eliminate impertinent alternatives. These external sources could be artifacts (e. g., research papers, code repositories) or human (e. g., colleagues, domain experts). For example, P2 often consults external resources before using a sniper strategy: *"…I read a lot of papers, and then I will see which algorithm is highly referenced among those papers and can apply to a wide range of cases."* Participants also rely on their past analytic experience or domain knowledge to generate, update and reduce alternatives, as when P3, P10–12 generate new hypotheses or when P2 eliminated algorithms based on past experience: *"We had a previous project about liver which used the first method. …It's about 20 000 points. The first method was already time-consuming. So this time we have more than 50 000 points and a more complicated structure, so sure the first one cannot meet our requirements, so we directly used the second."*

When a problem is poorly defined or the goal is unclear, participants might employ "trail & error." For example, P9 describes generating visualisations for clients: *"We need to talk with them to understand what they mean by block, or by thing. Once we think we understood,*

we write down a mathematical definition of the object, we compute it, and show it to the people and say this is what you were talking about, this geometry here. They are like yes or no."

When working in a team, participants might adopt a *divide & conquer* strategy, with each team member exploring a subset of the alternatives. For example, P5 and his teammate individually collected potential datasets. After separately filtering and clustering, they presented the clusters to their professor to decide which one to pursue. Similarly, P7 works with two other colleagues to find a model for their data collected in experiments, where each of them implemented one or more algorithms to see whether it works.

3.4.2 Reasoning about alternatives

Reasoning About Alternatives often involves sub-tasks. One common task is to compare and contrast alternatives to inform evaluation or decision-making. Some participants would use *metric guidance* for this comparison, as when P3, P6–8 calculated percentage accuracy, error rate, or deviation from a baseline. Other participants used qualitative approaches, such as making visual comparisons between alternatives (P1–2, P5, P9). Our participants expressed the need to view and compare multiple alternatives simultaneously, as was also highlighted in other studies on alternatives [3, 7, 56]. They often compared alternatives side-by-side (P1–3, P5, P7, P11), such as when comparing two program files. Others used layering (P3, P5–6, P9) and animation (P2, P6). Some alternatives exist for the sole purpose of comparison. For example, P1 created an intermediate alternative of a spreadsheet calculation that served as a link to a past version that he could use when communicating with his boss.

Reasoning also involves drawing on data workers' *past experience or domain knowledge*, as when P3 and P10–12 rely on their domain

knowledge to generate new hypotheses. P3, for example, would interpret the results of different visual trackers by inspecting their output layers in a convolutional neural network. When beyond their expertise, they might *consult external resources*, such as by asking doctors to make sense of different modeling results or consulting data providers to better understand data features.

Reasoning is harder when facing a large scale of alternatives. Four participants needed to tackle a large number of alternatives during their analyses (P1, P3, P6, P9), ranging from hundreds of alternative graphs to thousands of data alternatives. For example, P1 analyzes graphs of hundreds of model simulations. His approach resembles the small multiples technique, where he arranges the resulting graphs in a matrix, providing an overview and allowing comparisons. He describes, *"We have pictures of more than 80 cartographies, with different parameters. We save them as pictures to Powerpoint, like a list of images or matrix. At that moment we can start looking at them, to analyze the change of parameters and the impacts, what changes, why it changes, when it changes, changes how."* P6, instead, uses an optimisation algorithm to narrow the solution space before focusing on specific simulations.

3.4.3 Managing alternatives

Participants managed alternatives either informally via annotations, using folder systems, or via dedicated versioning tools such as Git or Microsoft TFS. Often, however, they used a combination of methods and tools. Some distinguish alternatives management at different abstraction layers. For example, P2 records different interpretations in her notebooks, saves parameter trails in spreadsheets to facilitate future analyses, and uses TFS to collaborate with colleagues. P6 saved all generated multiples in one place and moved promising options to a

separate folder. Four users felt tension between exploration and management, as P11 struggled, *"if I'm in the mode of exploring, I really don't want to lose time in making things clean…."* Two participants performed post-hoc cleanup, while another tried to balance generating alternatives with maintaining a cleaner record of those explored. Another admitted to rarely cleaning up: *"I want to go and clean it up afterwards, so that I can share it with other people. It's kind of my responsibility to do it afterwards, but I don't do it, in theory you can easily delete a cell, but I don't do it"* (P7).

3.5 Application of Alternatives Framework on analysis tools

In this section, we apply our alternatives framework to four data analysis systems drawn from the literature. We chose the first two systems—*Variolite* [8] and *LitVis* [49]—because they seem conceptually similar: both explicitly address non-linear, branching processes involved in data science coding activities, and both build in a code editor environment. Whereas the first two systems focus on code-oriented tasks, the second two systems—*EvoGraphDice* [63] and *Aruvi* [70]—both focus on interactive data exploration and modeling. These two systems address different kinds of problems, both conceptually and in terms of their fundamental relationship to alternatives. We demonstrate how the alternatives framework can help reveal the different kinds of alternatives they address.

Variolite integrates version tracking/history logging similar in concept to Git directly into the code editing environment [8]. It introduces the conceptual concept of code variants, where the user creates alter-

nate implementations directly in the code editing environment—for example to use a "strict matching" vs. a "fuzzy matching" implementation of a search function, as shown in figure 2.8. These variants can be revisited through a branch map directly embedded within each variant.

In terms of degree of attention, a variant under active development is a choice, with other variants as options. As the user no longer considers certain variants, they may become multiples rather than options until the user eventually (perhaps) revisits them. As such, Variolite provides explicit support for managing these three types of alternative.

The key insight of Variolite is in the execution abstraction level, focusing on providing support for how the user can manage such alternatives directly within the coding environment and reducing the interactive friction involved. Rather than managing chronological versions of a whole project, the user explores these alternatives and their combinations directly in the interface through explicit support for the execution-oriented processes: generating, updating, reducing, and managing alternatives. Variolite reduces the friction involved in the management of these execution and code artifact alternatives so that more cognitive resources can be devoted to the cognitive aspects of the task, beyond the scope of the system.

LitVis also provides explicit support for alternatives in a code editing environment, focusing on the concept of design expositions rather than code variants in the context of a literate visualisation computational notebook [49]. Instead of creating alternatives in a single document as in Variolite, in LitVis a user generates alternatives by adding a new parallel "branch." Each branch is a single file which can be referenced in the root file and can further diverge into subbranches by referencing to other files, which together form a document

tree structure, as shown in figure 3.5. As the user's attention shifts in the analysis processes, the role of these branches can move along multiples, options, and choices. Alternatives can arise at a more fine-grained level—for example, a single branch can contain many code blocks, each of which can render a different visualisation with various encodings or take different data as input.

LitVis also introduces "narrative schemas," which drive visualisation designers to reflect on and share each design's rationale, reifying these cognitive alternatives in the system design. For example, by applying the Socratic questioning schema, users answer questions such as "What would you do differently if you were to start the project again?" As put by the authors, "Each branch in a tree can represent alternative potentially competing designs each with their own rationale...intend to capture the rationale and knowledge associated with visualization design decisions at a more granular level."

Comparing these two systems in terms of the alternatives frame-

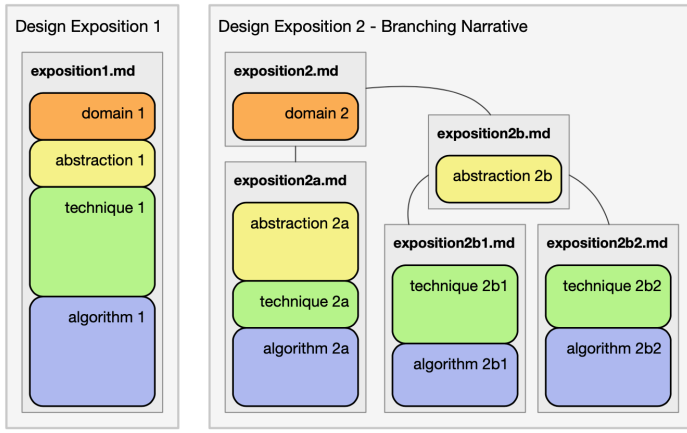


Figure 3.5. A more traditional single litvis file (left), vs. A branching narrative structure in multiple files (right) in LitVis [49].

work, both systems thus provide explicit support for multiples, options, and choices. Variolite focuses primarily on the execution abstraction level, leaving cognitive alternatives beyond the direct scope of the system. In contrast, LitVis provides more explicit support for cognitive alternatives, while providing more limited, coarser-grained support for execution-level alternatives.

EvoGraphDice helps users explore multidimensional datasets on a large number of alternative projections [63], as shown in figure 3.6. The primary alternatives here are different visual projections. EvoGraphDice combines the automatic detection of visual features with human interpretation and focuses on the generation of new multiples as well as the transition between multiples, options, and choices. The tool first presents the user with a scatterplot matrix of potentially interesting views based on principal component analysis (PCA). In terms of degree of attention, the individual views in the scatterplot matrix are multiples, receiving only limited attention from the user. As the user explores and considers different multiples, those with meaningful or interesting visual patterns often become options. The user can select a specific view to inspect and can provide a satisfaction score to help guide the system as it uses an evolutionary algorithm to breed new candidate projections. Users iteratively repeat this process until reaching a satisfactory finding. Using the notion of "frame of reference," the user's choice itself becomes a multiple amongst the new generation of multiples bred by the evolutionary algorithm. As such, EvoGraphDice is the only one of the four systems to provide explicit support for the Generation of multiples. Moreover, a "selection history" panel helps users manage insightful views by saving them as favorites (options) and revisiting previously saved configurations (making choices).

In terms of abstraction level, we find that EvoGraphDice provides limited support for alternatives at the cognitive or execution layers.

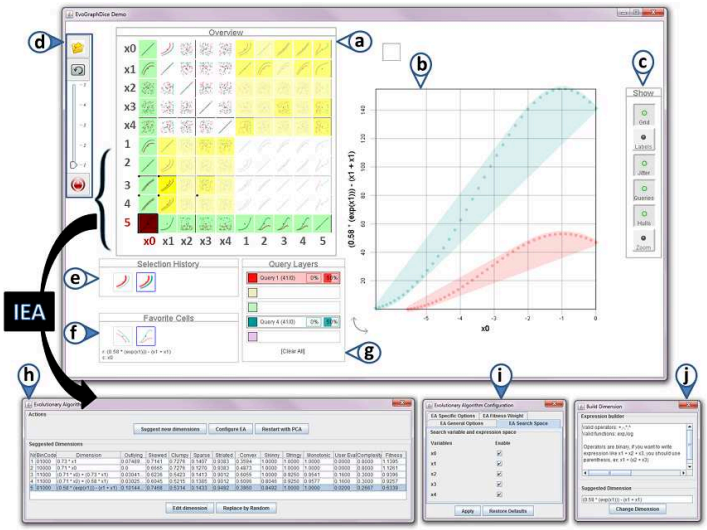


Figure 3.6. EvoGraphDice enables users to explore multidimensional data sets on a large number of alternative projections [63].

The latter would seem beyond the scope of this system. For the former, the user can generate, update, test, and discard hypotheses, but the application provides little explicit support to manage them beyond saving interesting views.

Aruvi aims to better support the sensemaking process during interactive data exploration [70]. Of the four tools, it provides the most explicit support for all three abstraction levels. The system interface combines three views—a data view, a knowledge view, and a navigation view, as shown in figure 3.7. The user can explore artifact and execution alternatives in the data view, such as for testing different types of visualizations and various encodings. The knowledge view supports generating and managing cognitive alternatives such as different hypotheses and mental models. It provides a flexible graphical

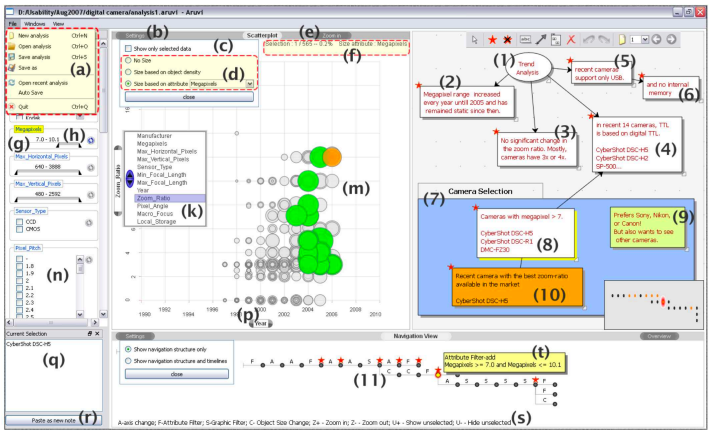


Figure 3.7. Aruvi system combines tree different kinds of views (data, knowledge and navigation) to support analytical reasoning process in visual data exploration [70].

environment where users can construct diagrams to externalize their cognitive alternatives. These cognitive artifacts can be linked with related visualizations in the data view to provide context and arguments.

During the exploration process, all changes made in the data view are captured automatically and represented as a history tree in the navigation view. The interdependency among artifact/execution alternatives and cognitive alternatives is partially shown on this tree, as the node which represents a specific state in the data view is marked when linked to a cognitive artifact in the knowledge view. With these three views, it is relatively easy for the user to navigate alternatives, recover contexts, and generate new ones based on existing state.

3.6 Discussion

In this section, we compare our findings on alternatives with past work and discuss the limitations of our study.

3.6.1 Comparison with other notions around alternatives

Many of the alternatives we have identified can be found in past research. For example, code and data alternatives correspond to the types of alternative mentioned in prior work [3, 8, 34, 43, 56]. In contrast to our work which study the role of alternatives in the overall sensemaking process, past studies focus on a subset of alternatives in specific sensemaking contexts or stages.

As discussed in section 2.1, past work has distinguished between alternatives, variations and versions (e. g., [8, 26, 27, 52]), with these definitions depending on the iteration of the alternative. While this distinction is useful, we find it incomplete: alternatives can evolve throughout the analytic process. What might be ephemeral (a multiple in our framework) at one stage might become the focus of attention later (an option or a choice).

Taking the example of tuning a model, according to past definition, the different parameters tested are often viewed as variations to a given model. In our framework, at one point, parameter alternatives can fall in the execution level and act as multiples, such as when P1 writing a script to help him test out different combinations of parameters. It's just a means to carry out his analysis activity, and no particular attention was put on any single parameter setting; at some point, however, the combinations of parameters themselves become distinct artifacts, as when P1 trying to inspect into the intermediate results to

explain the difference caused by the different sets of parameters, thus they become options.

As we discussed earlier, the type of an alternative does not appear to be a concept intrinsic to the alternative; rather, it seems to at least partially depend on the data worker's frame of reference. In our framework, the orthogonal combination of degree of attention with level of abstraction can help clarify the different, often conflated, notions of alternatives in sensemaking activities, and can enrich our vocabulary to describe them. It enables us to better capture the transitions of roles of alternatives, and their fluidity in sensemaking activities.

Moreover, past studies tend to focus most on artifact or execution alternatives and invent techniques or systems to support them. Our framework, instead, reveal the equal importance of the cognitive, artifact and execution alternatives. Our framework further emphasizes the interdependency between artifact alternatives and alternatives at the cognitive and execution abstraction layers. When designing tools to support alternatives in sensemaking, it is important to consider the links between them.

The third dimension in our alterntives framework describes the processes around these different kinds of alternative. In general, it contains a divergent process to generate many alternatives, followed by maintaining and updating all or some of them (the parallel process), and eventually a convergent process to narrow down the scope of possible solutions. Our high-level processes around alternatives share similarities to Peng's double-diamond metaphor on the data analysis process [97]. While we further demonstrate the various kinds of reasoning and management activities around alternatives (figure 3.4). They are two essential parts accompany the throughout of divergent, parallel, and convergent processes.

3.6.2 Study Limitations

The proposed framework for alternatives is only an initial attempt to unpack the notion of alternatives in data work. In this section, we reflect on the limitations of our work from three aspects, conceptual, method and sample limitations.

Conceptual limitation Data analysis is often not performed alone, same as the exploration of alternatives. Although collaboration takes an essential part in sensemaking activities, in this study, we understand and structure alternatives by observing and studying individual sensemaking process. This focus on individuals might constrain our findings. We could potentially neglect some types of alternatives that are more likely to happen in a collaborative context. The processes around alternatives and the strategies to cope with them could also be not the same.

Yet, in our semi-structured interviews, some collaborative factors that influence alternatives still get captured. When analysing the reasons why people explore alternatives, participants declared collaboration as both triggers and barriers of exploring alternatives. Our participants also revealed that they consult collaborators or other types of domain experts to better reason on alternatives and to inform evaluation or decision-making. Various kinds of strategies also have been applied to communicate alternatives to their collaborators, such as P1 always created an intermediate alternative of a spreadsheet calculation that served as a link to a past version that he could use when communicating with his boss.

Method limitation Before adopting the interview method for this study, we initially collected online data stories such as blogs and computational notebooks (e.g. from Kaggle and other open sources [98]),

as well as from the IEEE VAST challenge reports. Although these resources are valuable to track the provenance of data analytics, they vary in terms of how much they reveal on the high-level goals of the analysis; the analysis processes data workers go through; the amount of detail regarding how the analysis is performed; and most important, the false starts or the dead ends encountered during analysis. Table 3.3 compares the four approaches we investigated: data stories, computational notebooks, reports from the IEEE VAST challenge, and interviews.

We found that data stories and blogs contain more detail about difficulties data workers meet during analysis, how they explore different alternatives, and how they reason about them. However, the number of high quality data stories we could find was limited. Computational notebooks combine code, visualizations, and text in a single document. Though computational notebooks are designed to support the construction and sharing of analytical reasoning, it has been shown that data workers tend to use them to explore ideas rather than to tell an explanatory story of a real world analysis scenario [26]. Similarly,

	Goals	Analytic Process	Details	False Starts
Online Data Stories	x	x	—	x
Kaggle		x	x	
Vast Challenges	x	—		
Interviews	x	x	—	x

Table 3.3. Four study methods based on how they reveal: high-level analytic goals; data workers' analytic processes; detailed descriptions on how they worked; any dead-ends encountered. Mark x represents a good information coverage. Partial information is represented by a line.

the computational notebooks we collected are either personal, messy artifacts that are hard for others to understand, or cleaned versions used to share distilled findings or to teach others. VAST challenge reports explain how new techniques and visualization systems function and share findings reached using those systems. However, few or no rich details can be found on the sensemaking process (especially failed analyses). As a result, we opted for a semi-structured interview method to unpack the processes behind successful and failed analysis scenarios.

While, interviews, as a retrospective method, has its own limitations. It could be common for participants to not be able to recall all the details happened during the analysis, as well as all the rationale behind. As a result, we could potentially miss certain "branches" of the real exploration process. Especially in our study, we intentionally design our interviews in a way that no explicit questions on alternatives were asked when inquiring their sensemaking processes (in order to not bias or orient our participants towards the specific role of alternatives).

Other types of studies could be further conducted to overcome this limitation, such as we can follow and observe the practices as data workers are conducting the analysis, and further combine with "think aloud" protocols to know what they are thinking. However, often a complex analysis can takes for weeks or even months, it could be difficult to conduct in practice. Other methods such as dairy study (a self-reporting of specific aspects of users' natural behaviors and thoughts) [99], or interaction logs could also be useful. For example, to study a certain type of alternatives, the frequency of different kinds of behaviors, or to study alternatives at different granularities.

Sample limitation With a sample size of 12 participants, we do not claim generalizability of our results. Our focus is on getting a deeper

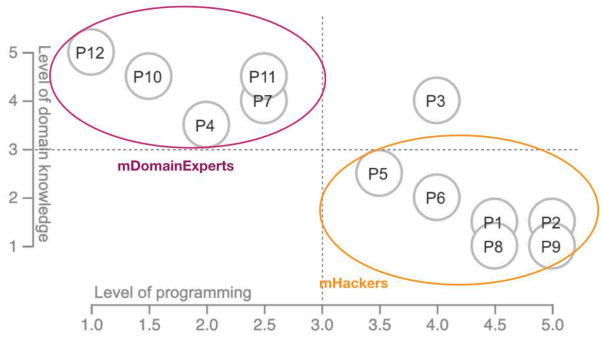


Figure 3.8. Participants' diverse levels of expertise in terms of computational skill and domain knowledge. Most participants belong to one of the following two groups: group "mDomainExperts" (mostly domain experts), who have more domain knowledge than programming skills; and participants with mostly strong computational skills but little knowledge of the application domain, "mHackers."

and rich analysis, rather than generalization from a larger sample size.

Our participants have diverse levels of computational expertise and domain knowledge; they play different roles within their broader organisational contexts. These different types of expertise may influence both the kinds of alternatives data workers tend to consider during analysis and the various coping strategies they generally deploy.

Although we did not control for expertise in this study, our study participants fell into two main categories (figure 3.8): those with high programming skills and little domain expertise (mHackers), and those with high domain expertise but little programming skills (mDomainExperts). We believe that many data workers in practice fall into those two groups. Our framework however does not assume any specific level of data work expertise.

3.7 Future Work

In this section, we identify design challenges for sensemaking tools in terms of supporting alternatives; and discuss other research directions for future work.

3.7.1 Design challenges for sensemaking tools

By looking at analytic systems under the proposed framework of alternatives, we find that current analysis tools often lack or have limited support for alternatives at the cognitive layer. Cognitive alternatives are intrinsically difficult to manage, as it requires getting tacit reasoning out of the head of the individual and into a form that can more easily be shared.

LitVis [49] provides one solution to reduce this friction by providing various kinds of schemas for analysts. These schemas allow users to structuring computational notebooks around their visualization design narratives. But this approach may not easily transfer to other contexts. In collaborative sensemaking context, Wang et al. [100] value team discussions in terms of externalizing design rationales and explaining alternative analytic paths. They design Callisto that integrates conversational messages as part of computational notebooks. Callisto further provides various kinds of links to enable two-way interaction between messages and related notebook contents.

We observed that our participants often use a combination of: note-taking tools (both digital or physical) to record cognitive alternatives; analysis tools to perform execution alternatives; and save artifacts alternatives in files, or other type of applications, such as Microsoft

PowerPoint. Linking and navigating alternatives across abstraction levels and tools remains an open challenge.

In general, current analysis environments break the chain of alternatives across different abstraction levels, as data workers often use a combination of tools to record alternatives at the cognitive, artifact, and execution levels. As such, these alternatives are treated as tangential and independent, separate from the actual analytic process. Users must mentally manage and navigate these different alternatives across distinct tools and environments.

Moreover, tool designers could consider providing more explicit support for managing data workers' degree of attention and the non-linear, fluid nature of alternatives. Many systems leave the management of alternatives beyond the scope of the system, leaving data workers to create ad-hoc solutions. How to design tools that make the generation, management, and exploration of various kinds of alternatives a fundamental element within the sensemaking process is an open challenge.

As discussed in section 2.3, the software application model makes it difficult to extend a tool, exchange information across multiple tools, and to combine different tools based on users' situated needs. We believe that to support alternatives in sensemaking is more than just add on a new feature in one existing system, or integrate an extra component in the interface. It requires more radical changes on the fundamental way tools are designed.

3.7.2 Other research directions

Though our study provides a deeper understanding on the nuance of alternatives and reveal its role in data sensemaking process, lots of challenges and questions still remain unexplored.

As discussed previously, further study using complementary methods is necessary to help reveal data workers' true underlying practices. For example, more detailed case studies can be conducted to ensure richer capture of alternatives.

The role of expertise in alternatives exploration also need to be analysed in more depth. Our study find some relationships between expertise and types of alternatives, such as participants with strong domain knowledge tend to mention cognitive alternatives more often; or participants who work in research organisations tend to explore more tool alternatives than participants who work in an enterprise, etc. (More details in paper [2]). Further research is necessary to help understand the impact of expertise in terms of alternatives exploration.

Furthermore, future research could explore the role of alternatives in collaborative sensemaking contexts. We would imagine new insights on various strategies deployed to coordinate exploration of alternatives within groups, to communicate the explored alternatives with others, and to evaluate or validate an alternative. New challenges could arise and new walk-around methods could be deployed to compensate for the lack of adequate features in the current tools.

In our interviews, we find that participants frequently struggle with other challenges around alternatives, such as how to evaluate them properly, how to decide when to stop exploring more alternatives, or how to compare, reason about, and manage a large space of alternatives. One direction could be researching how to leverage the power of machine, such as different kinds of algorithms, with human-in-the-loop in terms of generating, reasoning, and reducing alternatives. Another topic could be studying how to support the comparison of a large amount of alternatives given the limitation on cognitive and computational resources, as well as the limitation on screen size.

[2]See our publication here: `https://ieeexplore.ieee.org/abstract/document/8805460`

Part II

Supporting Alternatives in Sensemaking with More Flexible Tools

CHAPTER 4

ADQDA - Supporting Alternatives in Qualitative Data Analysis

With the general understanding on alternatives as fit within the sensemaking process, we then move on to the second research question - **How can tools better support the exploration and management of alternatives?** This chapter identifies the kinds of alternatives within the specific context of qualitative data analysis with affinity diagramming technique. We provide a vision and proof-of-concept prototype ADQDA to support analysts appropriating available devices to fluidly migrate between multiple analytic phases or adopt alternative analytic methods and representations. By embedding various kinds of linking mechanisms, ADQDA preserves a consistency between alternative artifacts as they arise in the process.

Section 4.1 introduces the affinity diagramming technique and how it is applied to qualitative data analysis. Section 4.2 reviews literature on affinity diagramming, qualitative sensemaking, as well as systems

designed for supporting them. Section 4.3 presents a design space for qualitative sensemaking tools and identifies painpoints of existing systems. Section 4.4 demonstrates our vision for more flexible sensemaking tools and the design and implementation of a proof-of-concept prototype, ADQDA. Section 4.5 validates ADQDA through a set of application scenarios. Section 4.6 discusses ADQDA using the proposed alternatives framework in chapter 3. Section 4.7 discusses the limitations of our work and section 4.8 presents the directions for future work.

4.1 Introduction

Affinity diagramming (also known as the KJ method) is a spatial clustering technique where analysts manually move around and group individual data items based on their similarity or relevance to a shared topic [101]. It has been applied for a wide variety of tasks in different domains, such as Human–Computer Interaction (HCI), anthropology, and management [101–103]. It is generally applied for three kinds of purpose [104]: to elicit diverse input, such as in brainstorming or project planning; to organize data into a known categories; and to analyze data, where people exploit spatial organizations to help them make sense of unstructured and seemingly fuzzy qualitative materials, such as interview transcripts. This work focuses on this last category.

Many activities are involved in building such understanding from collections of data: analysts need to closely examine original materials; make highlights or annotations while reading; extract related pieces of information to put them in space; experiment with different kinds of arrangements to explore conceptual alternatives; they might also build higher level diagrams to abstract the emerged concepts, etc. During the sensemaking process, analysts often mix alternative meth-

Figure 4.1. An example analysis using paper, where the user has customized the environment to support various analytic activities and representations in qualitative sensemaking. Source: Sarah Scarsbrook.[7]

ods, go back-and-forth between different analysis stages, and adopt various kinds of representations [1, 5]. Fig. 4.1 shows a real analysis environment customized to support various activities and representations with physical paper and walls [1]. Instead of discrete pieces, these analytic artifacts and representations are connected in various ways. For example, a participant's quote in a sticky note would make little sense without its context in a transcript.

During analysis, many additional factors may come into play: whether the analyst is working alone or collaborating with others; what kinds of analytic methods they prefer, such as open- or closed-coding or a mix of the two; whether they are working in the same location or from remote work sites; the tasks they are engaged in; or available devices. When working on a common analysis project, different collaborators might adopt different tools, artifacts, data representations and methods to gain insights from the data, and they might tackle on the analysis from different perspectives [105, 106]. Even a single user might need to adopt different displays or methods when works from office or from home.

Despite the pervasive of all kinds of displays, many analysts still prefer using paper tools due to its flexibility and easy to be extended

[1] `https://theartsjournal.net/2019/03/28/the-coding-cave/`, accessed on 31/03/2021.

to enable "space to think'' [71]. Yet practitioners report wasting large amounts of time to maintain relevant artifacts across multiple representations which distracts them from the real analysis tasks [107, 108]. The sensemaking process is often a long and mentally effortful process, where analysts can face hundreds or even thousands of notes and can keep working on the analysis for weeks or months [104, 107]. With the absence of connections between relevant artifacts, analysts might even lose track of them. Paper tools are not persistent and also cannot support remote collaborations.

Digital tools, on the other hand, tend to support one particular analytic method or stage, or impose certain analysis workflows. They do not address the holistic process of qualitative data analysis. It is hard to combine them to fluidly migrant between different analytic phases, or adopt various methods and representations. Moreover, digital tools are often designed for specific displays, such as desktop-based applications. It is difficult to appropriate different devices or combine them to extend space for multiple diagrams.

We present a vision and proof-of-concept system ADQDA (Affinity Diagramming for Qualitative Data Analysis), a cross device, collaborative affinity diagramming tool for dynamic and iterative sensemaking process. In our vision, users may work alone or in collaboration across the different activities and representations of qualitative data analysis. Users may seamlessly flow between these different analytic phases, methods, and representations in arbitrary order, all while preserving consistent analysis artifacts. They may use any appropriate or available device for the task at hand, for example by having multiple users code interview transcripts on their individual laptops, then appropriate a large wall-sized display for affinity diagramming, or using a whiteboard-sized display in an office. They may use their wall in conjunction with a laptop, phone, or tablet as they need space to think [71] or to display related information. As such, the analysis tool should be flexible and dynamic—in terms of analytic process, in

number of devices, in number of users.

ADQDA applies the concept of shareable dynamic media [14] in multi-surface environments to allow the dynamic distribution of different devices across multiple users. Changes are synchronized live across devices to all users, similarly in concept to online collaborative word processors. ADQDA embeds various linking mechanisms to preserve consistent analysis artifacts throughout the process. We further design various multi-surface interaction techniques to reduce the cognitive demand of users when iterating between different analytic views, and offer awareness mechanisms to solve potential cognitive breakdowns of collaborators.

The contributions of this work are thus:

- a vision for collaborative, cross-device affinity diagramming for qualitative data analysis, drawn from our own experience and from interviews with practitioners,
- a proof-of-concept prototype, ADQDA, implemented as a distributed web-based system using Webstrates [14], and
- a series of demonstrative scenarios that show how ADQDA realizes this vision and can potentially facilitate richer, deeper qualitative data analyses.

4.2 Related Work

In this section, we review related work on how affinity diagramming is applied to make sense of data and the tools that support it. We then present different approaches to analyze qualitative data in general and the representative analysis tools that support them. Finally, we consider the device ecology in the web of analysis activities.

4.2.1 Affinity diagramming practices & tools

Affinity diagramming is a frequently-used tool for organising ideas and making sense of them [101,102]. Harboe and Huang [104] investigated real-world affinity diagramming practices by interviewing 13 participants. They found that the term affinity diagrams captured a variety of practices, with different purposes and procedures. When used "To Elicit Diverse Input," such as brainstorming or project planing, the number of notes tends to be smaller and the duration shorter; when used "To Analyze Data," such as from surveys [109, 110], field interviews and observations [5, 111–113], it tends to be more cognitively intensive with relatively large numbers of notes and a longer duration [104].

Many current digital diagramming systems, such as Affinity+ [114] or AffinityTable [115], are designed to support brainstorming-like scenarios. In this work, we focus on making sense of collected qualitative materials. When dealing with increasing numbers of notes, space constraints are often a problem [104, 107, 108]. With paper tools, people often appropriate available space, such as by spreading out to other walls or tables [107, 108]. Digital tools, however, "... [are] thought to restrict users to the confines of a pre-determined space such as a desktop instead of using any and all space available to them" [108]. Some commercial diagramming tools, such as Miro [2] or MURAL [3], use an infinite canvas to facilitate an ever-expanding diagram, but still are limited to a single window's viewport. Different strategies are adopted under such space contraints, such as only working on one part of a diagram at a time or omitting less important ones, which can harm the analysis [104, 116].

Another common painpoint is the transition of artifacts between

[2] `https://miro.com/`, accessed on 05/04/2021.
[3] `https://www.mural.co/`, accessed on 05/04/2021.

different systems [104, 117]. For example, Harboe et al. found some participants spent as much time just copying highlights to sticky notes as on the analytic task itself [104]. Tools can help transforming paper notes to digital [117, 118]. Much of today's data are collected initially in digital forms; transforming directly between digital systems seems appealing. However, many current digital diagramming tools focus on scenarios where inputs are primary ideas—as in brainstorming—rather than data items extracted from documents. Additional tools, such as word processors, need to be integrated in the process. Since each system has its own design abstractions, transitioning data from one to another could be much more cumbersome than just using import, export functionality.

Sensemaking is an iterative process [119], and affinity diagramming as a specific sensemaking technique inherits this nature [101]. Analysts iteratively go back and forth between different analytic phases and representations. In this context, the disconnect between analytic artifacts in each phase and representation could lead to cognitive breakdowns or otherwise hinder the analysis. For example, Judge et al. [108] observed that "Even if interview transcripts are available to users, as the number of notes increases, it becomes impractical to look up the meaning and context of notes." Based on their observations of different affinity diagramming sessions, they suggest to attach snippets of original transcript to the notes, and to provide personal workspaces for individuals in conjunction with the wall sized display, and also facilitate users as they transition between them. Practitioners may also generate multiple diagrams, such as for the purpose of looking at data from a different perspective [5, 113], which may add on additional challenges.

Our research identified similar painpoints that align with the related work. We propose addressing space constraints by allowing users to add in additional available displays, such as using a digital whiteboard in conjunction with a wall size display. Additionally, we propose

linking the various artifacts in different contexts, and providing interaction techniques to help users iteratively transition between analysis phases and representations.

Moreover, affinity diagramming is often performed collaboratively [101, 108]. The number of users can range from a small group of two, to a group of more than twenty collaborators [104, 107]. Paper and some digital tools, constrain users to a shared location. Digital tools, such as Distributed Designers' Outpost [120] or Miro[2] support remote collaboration on the same diagram, and provide extra awareness mechanisms such as shared cursors or shadows of the remote collaborator for presence. Our approach aims to not impose any specific collaboration style. Users can join an analysis in any phase, at anytime, in the same location or from remote places.

4.2.2 Various sensemaking methods & approaches

Affinity diagramming, as a specific sensemaking technique might be used together with other approaches, such as coding, which is more structured and is embedded in documents [121, 122]. Even within the context of affinity diagramming, various strategies could be applied [104]. Two common strategies are open coding and closed coding. Open coding, also called substantive coding or grounded coding, as an inductive form of analysis, emphasizes gradually identifying themes and concepts that emerge from data to construct new theories [123]. Closed coding, also called a priori coding or template coding, takes a deductive approach where codes are created beforehand based on certain theoretical frameworks or pre-existing knowledge and are used to frame the data into a coherent construct [124].

In practice, rather than following one approach, practitioners often

blend the two into different variations to suit for their own needs. For example, they can begin with a set of a priori codes based on interview questions and iteratively add to them as new themes emerges. In other cases, researchers start with open coding to get some preliminary categories, and follow by closed coding to identify more specific themes [125, 126]. As Elliott observed, "The most pragmatic researchers will typically use both in a single research project" [127]. Blair found that a combination of these two can reduce confirmatory bias since they "speak to, and counter, one another" [128]. Chandrasegaran et al. [129] show how the processes of Grounded Theory [130] parallels Pirolli and Card's sensemaking model [119], where analysts iteratively switch between bottom-up (from data to theory) and top-down (from theory to data) activities. Both coding and diagramming or a mix of the two could be adopted to these models based on concrete projects.

Commercial computer-aided qualitative data analysis software, such as MaxQDA [4], ATLAS.ti [5], and NVivo [6], support both coding and diagramming. However, they impose a certain analytic sequence, such as coding before diagramming. They are also limited to a desktop-based, single-user model. We focus on the need to blend diagramming and coding in different analytic phases, across heterogeneous devices, with multiple users working in the same place or remotely.

[4] `https://www.maxqda.com/`, accessed on 05/04/2021.
[5] `https://atlasti.com/`, accessed on 05/04/2021.
[6] `https://www.qsrinternational.com/nvivo-qualitative-data-analysis-software/home`, accessed on 05/04/2021.

4.2.3 Artifact ecology in collaborative sensemaking activities

Real-world analytic processes are often dynamic, with users from same or different locations arbitrarily joining in any analytic task among a network of activities [104, 107]. Instead of designing tools for certain tasks, Bødker and Klokmose's Human-Artifact Model emphasizes studying heterogeneous devices as part of an artifact ecology where "one can dynamically interplay with others and with users' web of activities" [15, 16].

Interactive displays, such as mobile phones, tablets, laptops, tabletops, and wall-sized displays, are becoming increasingly accessible and pervasive in our daily life. Different type of devices has their own affordances and technical capabilities, in terms of forms, sizes, input modality and input accuracy. Brudy et al. classify the diverse devices as Ad-hoc/Mobile, Semi-fixed and Fixed, based on their support degree for dynamic changes and reconfiguration [109]. In Scharf et al.'s taxonomy, devices can be categorized based on ownership, access, and distance [131]. The unique characteristics of each of these devices make them more suited to certain collections of tasks and interactions. Different kinds of devices are often combined as an ecosystem for more complex tasks and knowledge work [15, 73].

Taking data analysis activities as an example, smaller devices, offer better mobility, portability and privacy [132]. They are often used at a personal scale or to mediate the interactions with large displays in multi-surface environments. Larger displays, by contrast, allow simultaneous access and can present much more information, often acting as a shared sensemaking space [93, 94, 133, 134].

To open up new opportunities to design tools for more flexible and dynamic practices, Klokmose et al. introduce Webstrates [14]. As discussed in section 2.3.2, Webstrates provide a malleable and col-

laborative environment. Users can easily access to it with any devices that provide modern web browsers. Different types of webstrates (web pages) can be assembled in a flexible manner to achieve asymmetric collaborations. Moreover, user can dynamically modify and extend the environment from within. Webstrates thus blur the boundaries of tools, devices and activities. Badam et al.'s Vistrates, built on top of Webstrates, demonstrates how heterogeneous devices can be flexibly combined to support dynamic activities around data visualization [90]. Similarly, our prototype ADQDA is built on top of Webstrates and we focus on the dynamic interplay of different types of devices and the various analytic activities within the context of qualitative sensemaking.

4.3 Design Space for Qualitative Sensemaking Tools

In this section, we describe our informal studies on gaining insights into how practitioners analyze data using affinity diagramming. We present a design space drawn from the literature, our own experience, and that of our participants to describe the various kinds of activities, methods, artifacts, and tools engaged. We then discuss the painpoints with current tools as revealed based on this design space.

4.3.1 Informal studies & analysis of collected data

As researchers in the HCI field, we have used affinity diagramming as a principal analysis method for taxonomic or interview-based studies

published at flagship HCI conferences. Besides reflecting on our own experience, we further conducted informal interviews with three practitioners, two HCI researchers and a sociologist, with rich experience using affinity diagrams. All had experience using paper, one had experience using a wall-size digital tool, and one had experience using desktop affinity diagramming software. Two had recently used affinity diagramming to analyze interview transcripts, while one had most recently used it to cluster social network diagrams. Interviews were took place in either their analysis environment or our multi-surface environment, with their own laptops or other related artifacts.

In our interviews, participants shared photos, videos, or websites of affinity diagrams generated in their past projects. They further walked through the process used to generate them. The goal of the interviews was to better understand how our participants used affinity diagramming in their research, their sensemaking processes, and to tease out potential painpoints they had experienced.

We then showed them an initial version of ADQDA with rudimentary support for creating basic affinity diagrams on a wall-sized display. It included basic support for importing notes from a .csv file, collaboratively organizing them on a multitouch wall (or with a mouse and scrollbars on a desktop computer), and adding textual annotations. (The primary affinity diagram view shown in Figure 4.3 is based on this initial version.) This part of the interview was aimed at helping participants to project themselves into the experience of conducting digital affinity diagramming on a wall-sized display.

Our analysis is based on the combination of literature reviews, reflections on our own experience, and that revealed by our participants. We applied grounded theory to the ensemble of all these sources, and focus on different themes, including the various kinds of activities in qualitative sensemaking process, the kinds of artifacts and representations involved, the different kinds of analysis methods applied, and the main pain-points that exposed.

4.3.2 Five dimensions in qualitative sensemaking

Drawn from the literature, our own experience, and that of our informal interview participants, we develop a design space to describe the different artifacts, representations, and analytic tools used in qualitative data analysis. In qualitative data analysis (with affinity diagramming technique), there are five main dimensions to be considered: (A) analysis phases, (B) conceptual methods, (C) analytic lens, (D) modes of collaboration, and (E) types of devices used. Various kinds of alternatives exist along these identified dimensions. In the following contents, we explain these dimensions as well as the related alternatives in more details.

4.3.2.1 (A) Analysis Phases

To better capture the sensemaking process, we first identify three core analytic phases in qualitative data sensemaking: 1) extract related data items from raw materials, 2) assign data items into categories (often called codes), and 3) draw connections between categories to form higher-level concepts, as shown in fig. 4.2 along the horizontal axis.

To analyze interview transcripts, analysts may adopt different approaches: some analysts may perform open coding, extracting coded concepts in a bottom-up fashion from the underlying data (starting from the first phase and moving right). Others may perform closed coding, where the specific concepts are defined in advance (starting from right and moving left). Frequently, the two are mixed. No matter the strategy applied, users may seamless flow between these different phases of analysis in arbitrary order. For example, codes or concepts

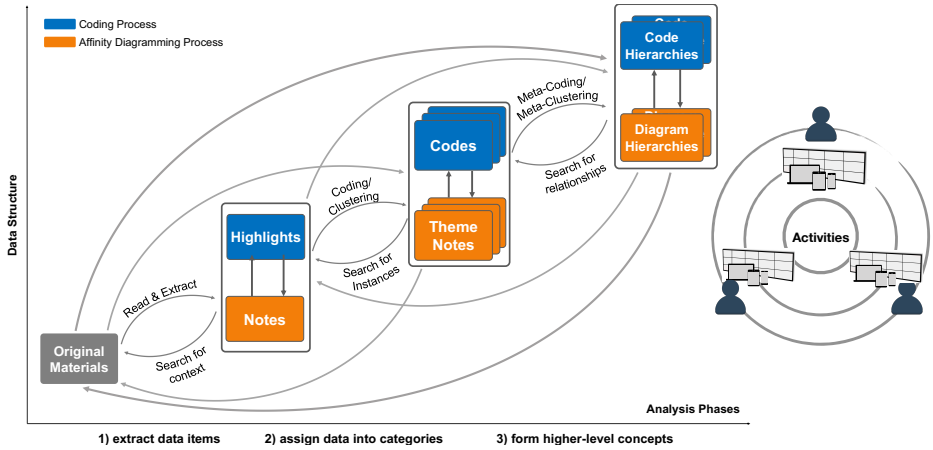


Figure 4.2. A design space for sensemaking tools: (A) analysis phases, along the x-axis; three primary analytic phases for qualitative data analysis; and (B) conceptual or analytic tools, such as coding (in blue) and affinity diagramming (in orange) as complementary methods for analysis. (C) Multiple Lens, multi-faceted analysis shown as stacks of coding systems or affinity diagrams. The links between phases and representations show the transitions and iterations between them during the sensemaking process. (D) modes of collaboration and (E) types of devices. Users arbitrarily join in any analytic task among a web of activities, and appropriate pertinent device for situated tasks.

could be updated as more data items are extracted (transition from phase 1 to 2 or 3); insights derived in one coding system could trigger the creation of another focused on a specific concept (loop inside phase 2) or could trigger a new pass over the raw transcripts (from phase 2 to 1); the emergence of new concepts or hypotheses could lead to collecting more data (from phase 3 to 1), such as conducting additional or follow-up interviews, etc.

Our interviews and experience align well with Pirolli and Card's

sensemaking model where analysts iterates on information foraging and sensemaking activities [119]. Instead of being discrete phases in a linear pipeline, these phases are indeed closely connected as one can influence and also rely on another. These phases describe iterative activities within the sensemaking process.

4.3.2.2 (B) Conceptual Methods

While we focus on using affinity diagrams as a tool to analyze data, it often is not used alone. We have seen practitioners mix coding and diagramming in a flexible way during different phases of their analysis. The blue and orange colors in fig. 4.2 demonstrate these two commonly used conceptual tools to conduct qualitative analysis.

Coding processes (in blue) represent analysis activities which are more structured. Such coding often entails highlighting text segments in the documents (phase 1); annotating highlights with keywords or tags - the codes (phase 2); and organizing or structuring those codes, such as into a hierarchy (phase 3). Affinity diagramming processes (in orange) are more spatially-oriented analysis activities, where the analyst uses "space to think" [71]. Extracting data is done by extracting them from their source and transforming them into notes, such as physical post-its or digital text snippets, that can be arranged spatially (phase 1). By clustering these notes into groups (often with theme notes to summarise the abstract ideas of a cluster), the analyst assigns data items into categories (phase 2). These theme notes can be further clustered to form higher level concepts, resulting in hierarchical diagrams (phase 3).

To be clear, the term coding does not share a universal definition, it can generally mean organise data to schemas no matter the concrete methods applied. Coding can be defined as "the process of analyzing qualitative text data by taking them apart to see what they yield

before putting the data back together in a meaningful way" [**?**]. In this dissertation, we separate the two terms, coding and diagramming, to better distinguish these two kinds of conceptual methods and put emphasis on their different characteristics.

We consider coding and diagramming as two alternative conceptual methods to analyze qualitative data. We find that instead of choosing either coding or diagramming, these two alternatives are often combined and mixed in a flexible way. They could be applied by different data workers in a same analysis project, some use coding and some diagramming; or be adopted by a single analyst, where he/she chooses appropriate methods depending on different contexts, such as using coding in the lack of empty wall space.

Coding and diagramming enable different kinds of complementary analyses, and both contribute to the cognitive interpretation of the data. In coding, ideas can be quickly recorded while reading. Highlights and margin codes are embedded in raw materials, which maintain their surrounding context. On the other hand, the act of moving notes around in space frees the analyst from the structure of the source document, helping to explore connections and categorizations of the data—at the cost of this context. In diagramming, moreover, the placement of notes, clusters or the distance between them can convey extra conceptual messages, such as a note put between two clusters could convey the uncertainty an analyst might have in mind of which cluster this note should belong to.

4.3.2.3 (C) Analytic Lens

During the sensemaking process, we found that analysts may need to view data through more than one lens, such as testing the fit of different ways of structuring the data [122]; structuring the data from the perspective of different analytic questions (e. g. strategies involved

vs. tools used vs. collaboration, etc.) [113]; partitioning the analysis across individual participants and then assemble them up to find common themes [5]; or exploring alternate hypotheses or interpretations by different collaborators. These analytic practices result in multiple alternative affinity diagrams, or multiple sets of coding systems.

We represent this multi-faceted analysis as stacks of coding systems or affinity diagrams in fig. 4.2. In this case, analysts often struggle at finding more space to arrange multiple diagrams. Though each diagram represents different interpretations and organizations of the underlining data, they often hold various kinds of relationships based on the users' mental model. For example, analysts may need to compare two alternative diagrams with overlapped notes to examine how a given note is clustered in each of them; or to form up higher-level analysis based on multiple diagrams. All of these representations as well as their relationships form up the conceptual map to understand what transpires in the data. Analysts hope for better ways to compare, link, and navigate among these alternatives.

In all, as shown in figure 4.2, the ensemble of these three dimensions - the different analytic phases, various conceptual methods, and multiple analytic lens - are integral parts of the whole analysis. Analysts iteratively flow between different analytic phases, choose or combine alternative methods across these phases, and generate, reason, manage multiple analytic representations and artifacts along the overall processes.

4.3.2.4 (D) Modes of Collaboration & (E) Device Types

An analysis can involve a dynamic number of collaborators at different analytic phases. They could conduct the analytic activities synchronously or asynchronously, co-located or from distributed work

sites. They may engage in the same or different phases or activities, using the same or different conceptual methods. Various coordination strategies could be adapted to their mode of collaboration. For example, based on our interviews and experience, some analysts might partition the data to work on distinct subsets at a given time; others may work on a first level diagram while a colleague simultaneously works on a higher level diagram; some use coding while others use diagramming. While although qualitative data sensemaking tend to be collaborative, sometimes the analysis could also been done all alone, such as the example shown in the introduction in this chapter[7].

In our study, we observed various kinds of heterogeneous devices that could participant in the overall analysis environment, including laptops, tablets, mobile phones, tabletops, and wall-sized displays. Modern technology has reduced the cost of interactive displays. It is common for both individuals and organizations to own multiple devices, from desktops or laptops to tablets, phones, and smart watches. Wall and table displays are also becoming more affordable. These alternative devices share some overlapping characteristics and still differ in terms of their physical affordances and computational properties, such as the size, form, input modality, input accuracy, etc.

The characteristics of alternate devices make a set of tools align well with some specific tasks. For example, coding interview transcripts might make most sense when using a keyboard and mouse or with a digital pen and a tablet, whereas affinity diagramming may be more natural when using a touch display and on a relatively larger screen. While it does not mean that a tool should be bonded with certain type of tasks, a user should be able to choose or combine the available technological devices best suited to the task at hand, just as a wood worker would use a chisel to sculpt wood and a screwdriver to

[7] `https://theartsjournal.net/2019/03/28/the-coding-cave/`, accessed on 16/09/2020.

join pieces.

With paper tools, it is common to expand to multiple walls and surfaces when requiring "space to think".We see examples where analysts take advantage of empty walls, tables, and windows in the room. Though digital displays have size limitations, the analysis view could be extended by combining multiple of them. Depending on the context, users should be able to adopt any appropriate and available device for the task at hand.

4.3.3 Painpoints: where current tools break down

In our study, practitioners also reveal various kinds of difficulties and challenges that they encounter. The design space stated above helps us to understand these painpoints and where current tools lack support:

Different analytic phases and methods are often treated as particular silos. Current tools tend to impose that users choose a certain conceptual method or follow certain analytical sequences. For example, digital affinity diagramming tools such as Affinity+, Miro, tend to focus on clustering techniques without considering where the notes come from or supporting coding in documents. Users must combine with other text editors such as Microsoft Documents. However, transitioning information between these tools requires a cumbersome export-import and potential data transformation. Qualitative data analysis tools such as MaxQDA seem to assume that users perform coding before diagramming. Re-coding ends up breaking existing affinity diagrams, such as changing the previously assigned locations of notes. A system brings barriers when users want to mix these conceptual methods, or could even prevent them from using another.

Though in real analysis processes, data workers frequently iterate between different analytic phases, mix coding and diagramming across these stages, we have find no tools that provide sufficient support for the overall sensemaking process as a whole.

Different analytic artifacts are frequently disconnected. As identified in the design space, qualitative data sensemaking involves a variety of analytic artifacts. These artifacts could be related to the two alternative conceptual methods: the pairs of highlights and notes, codes and theme notes. They can also be introduced by the multiple analytic lens, such as multiple diagrams focusing on different analytic perspectives, or alternative diagrams developed by different collaborators.

In current tools, these artifacts are often disconnected from each other, leaving users apply ad-hoc strategies to manage this mass of intertwined information, hidden in different locations and updated at different rates. For example, an excerpt from an interview transcript may no longer make sense or may take on a different meaning when extracted from its surrounding context. Judge et al. observed that, with paper tools, "Even if interview transcripts are available to users, as the number of notes increases, it becomes impractical to look up the meaning and context of notes." [108]. Some users manually write down participant number and page number in every notes to facilitate tracking back in the documents.

Another example considers mixing coding and diagramming. The asynchronization between the related artifacts could confuse data workers on their process or potentially cause mistakes. Users need to maintain a strong mental model to navigate between them and make sense of these artifacts distributed across multiple contexts and analytic stages. Even though the computational power of digital tools seems promising to maintain the various kinds of connections, extra

challenges could be introduced due to the messy iterations and the different characteristics of coding and diagramming.

Current systems are often designed to suit for some specific displays or collaboration modes. As identified in the design space, an analysis environment in real world is highly dynamic, in terms of the number of collaborators, their collaboration modes, and the number and types of digital devices involved. Users arbitrarily join in any analytic task among a web of activities, they may bring in new devices, remove devices no longer required and transition between different devices for situated tasks.

However, current tools are often designed for some specific displays or collaboration modes. For example, both MaxQDA and NVivo are designed for laptop-based environments, and are ill-suited for use on a wall-sized display. Paper tools make it hard to collaborate remotely. As one participant put it, "one of the biggest painpoints is that it needs people to appear in one place. But we have teams in other cities, it's just impossible to collect all to this room for example, standing in front of the wall to work together." All participants described challenges arising from the locality of the affinity diagram. For example, the sociologist shared with us affinity diagram photos taken to support working from the office, in a different room. Moreover, when collaborators were travelling or otherwise working remotely, it was difficult to collaborate around the same affinity diagram.

4.4 ADQDA - A Proof-of-Concept Sensemaking Tool

Drawing upon the design space and analysis of painpoints, in this section, we present our vision on qualitative sensemaking tools, and demonstrate a proof-of-concept prototype, ADQDA (Affinity Diagramming for Qualitative Data Analysis), where users can appropriate available devices to flow between different analytic phases, methods, representations, all while preserving consistent analysis artifacts.

4.4.1 Design vision & design goals

Rather than split the user's analytic workflow into discrete processes performed across independent tools, we envision qualitative data analysis as a holistic process, with consisting of fluid analytic phases, conceptual tools, interactive devices, and modes of collaboration.

Users should be able to seamlessly migrate between different artifacts, data representations, and analysis methods, and carry across those analyses as they do so. They should be able to combine whatever digital devices they have as they are suited to the analysis at hand. Users should be able to adopt the collaboration style that suits their context, whether they be colocated or across different time zones.

We summarize this vision with the following design goals:

[**G1**] Support different analytic phases and the iterative transitions between them.

[**G2**] Support the heterogeneity of analytic methods and the flexible mix between them.

[**G3**] Link related analytic artifacts in different phases, methods and representations.

[**G4**] Allow users to appropriate available digital resources based on situated tasks.

[**G5**] Allow users to apply flexible collaboration modes.

4.4.2 ADQDA Design

Based on these identified goals, we design and implement the ADQDA (Affinity Diagramming for Qualitative Data Analysis) system. We explain ADQDA in details and explain how the design goals are achieved in ADQDA.

4.4.2.1 System main components

ADQDA treats the three analytic phases as well as two conceptual tools as first-class citizens ([**G1,G3**]). The system consists of three

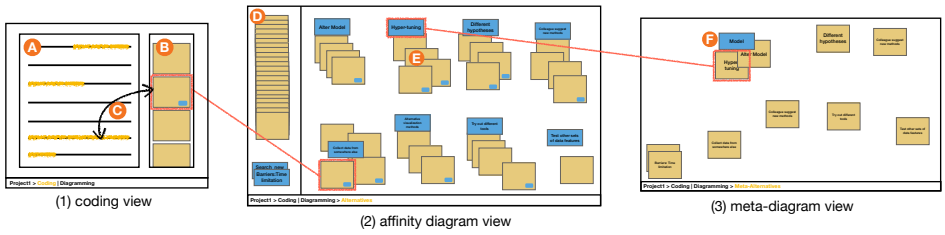


Figure 4.3. The main components of our system and the links between them: (1) coding view, (2) affinity diagram view, (3) meta affinity diagram view.

main components: (1) a coding view, (2) an affinity diagram view and (3) meta-diagram view, as shown in fig. 4.3.

The coding view contains the source document for analysis (fig. 4.3A) and a "notes panel" for identified data items (fig. 4.3B). New documents can be added and the one in current view can be switched. Users read through the documents and highlight content while reading. Once a text excerpt get highlighted, a corresponding note is automatically generated in the "notes panel". Users can directly tag these excerpts in the notes panel. Long clicking a highlight will navigate to the corresponding note in the notes panel and vice versa (fig. 4.3C). Documents can also be accessed on their own to facilitate read-only cases, such as when checking the context of a note on a mobile phone.

Users can switch the current view to an affinity diagram or create a new diagram using the navigation menu at the bottom. Affinity diagrams contain two types of notes: standard *notes* (colored in yellow) contain data extracted from raw materials; and *theme-notes* (colored in blue) contain user-assigned messages, often keywords that describe a given cluster in the diagram, as shown in fig. 4.3(2). The left *"messy pile"* region (fig. 4.3D) in an diagram contains notes that have not been analyzed.

Users create theme notes by double clicking/tapping on the diagram background. The theme-notes together with the notes grouped around it construct clusters(fig. 4.3E). All notes can be dragged around in space as in physical affinity diagrams. Every cluster can be moved as a unit as well. To group a note in a cluster or remove from it, users move the note near or far away from the cluster, a tag will appear or disappear on the note as a visual feedback. All tags within a cluster are synchronized: when the content of a theme note gets updated, all related tags are synchronized to the updates as well.

Meta-diagram is used to form up higher-level analysis from the basic diagram. In the meta-diagram view, standard notes (the yellow ones in 4.3B) are hidden by default, and theme-notes are turned

to standard notes and can be clustered under *"higher-level theme-notes"* (fig. 4.3F).

4.4.2.2 Iterating between phases

ADQDA aims to support non-sequential sensemaking by letting users fluidly iterate between the three analytic phases ([**G1**][**G3**]) - 1) extract related data items from raw materials, 2) assign data items into categories (often called codes), and 3) draw connections between categories to form higher-level concepts.

Iterating between Phase 1 and 2: When doing affinity diagramming, users often spend significant effort to coordinate documents and notes [108]. ADQDA facilitates this transition by automatically generating a note in existing affinity diagrams for each highlight in real time. Users can further navigate between these contexts: notes are linked to their source documents and vice-versa. In a diagram, the users can, for example, click the *"show me more"* menu on a note to open it

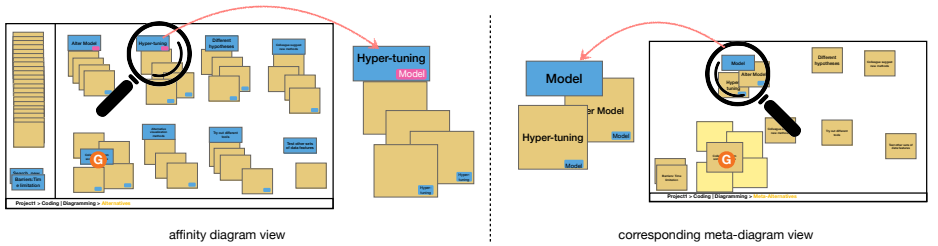


Figure 4.4. Cluster information generated in each hierarchy level is hidden in another by default. But it can be acquired on demand: Inquiring meta-cluster information in a basic diagram (left graph). Inquiring hidden notes from lower-level diagram in the meta-diagram (right graph).

in the corresponding transcript on a pre-registered device, such as a phone or tablet. Additional devices can be registered by navigating to the session's URL on the new device. Similarly, long-pressing on note in the coding view highlights it in its diagrams.

Iterate between Phase 2 and 3: In ADQDA, users can further generate meta-diagrams for higher level analyses. The notes to be analyzed in a meta-diagram come from the theme-notes in the basic diagram(s). It can come either from a single diagram or from the theme-notes in a combination of diagrams. To help users concentrate on current analysis, cluster information generated in one hierarchy level is hidden in another level by default. But it can be acquired on demand: In the basic diagram, users can turn on *"seeing meta info"* mode to examine corresponding meta cluster information, as shown in fig. 4.4; in the meta-diagram, users can expand notes to reveal the individual notes they collect at the lower level. To maintain spatial cognition [108] (fig.4.4G), this expansion preserves the spatial arrangement of the notes underneath as in the lower level diagram.

Iterate between phase 1 and 3: During the analysis, users could also go directly from meta analysis to original documents, such as to include additional participants data; or from phase 1 to 3, such as when new arguments find in documents reject a current hypothesis. In ADQDA, though users can view the basic notes (those extracted from documents) in a meta diagram, they are not allowed to directly moving these notes from the meta diagram. Users are not allowed to directly assigning a meta tag to a note in the document view as well. Thus the iterations between phase 1 and 3 happen more implicitly through the manipulation on the basic affinity diagrams.

Using the navigation bar at the bottom of each view, users can easily switch their current view to any other views. Moreover, the docu-

ment view, the multiple diagrams and meta-diagrams can be accessed simultaneously on different devices, allowing users to make comparisons and iterate their current analysis focus by simply switching their attention from one display to another.

4.4.2.3 Mixing coding and diagramming

In ADQDA, the three pairs of artifacts: notes and highlights, tags and theme notes, hierarchical codes and meta-diagrams, are treated as alternative forms of the same entity. ADQDA automatically maintains a bijection or a synchronization-mapping between coding concepts and their equivalent diagramming concepts ([**G2**][**G3**]).

Mapping Diagramming Activities to Coding: Theme notes are treated as tags (codes). Adding a note to a given cluster thus tags that note with its theme, and removing it from the cluster untags it. In the coding view, a highlight thus shows its associated themes as tags. If it has been clustered in different diagrams, it thus has multiple tags. When viewed in a diagram, the note only appears in its own cluster(s); tags from other diagrams are thus not shown to avoid distraction. Deleting a note from a diagram removes it from that view, but does not remove it from other views.

Mapping Coding Activities to Diagramming: When coding, users tag highlights in the document. To create a tag, users either double click the highlight or its corresponding note in the "notes panel" by side (fig. 4.3B). They can choose from a list containing all up-to-date cluster information or edit new ones. A highlight or a note can be assigned with multiple tags. The flowchart in figure 4.5 demonstrates how this coding action is mapped to corresponding diagramming actions: If the tag does not yet exist, the system prompts the user to

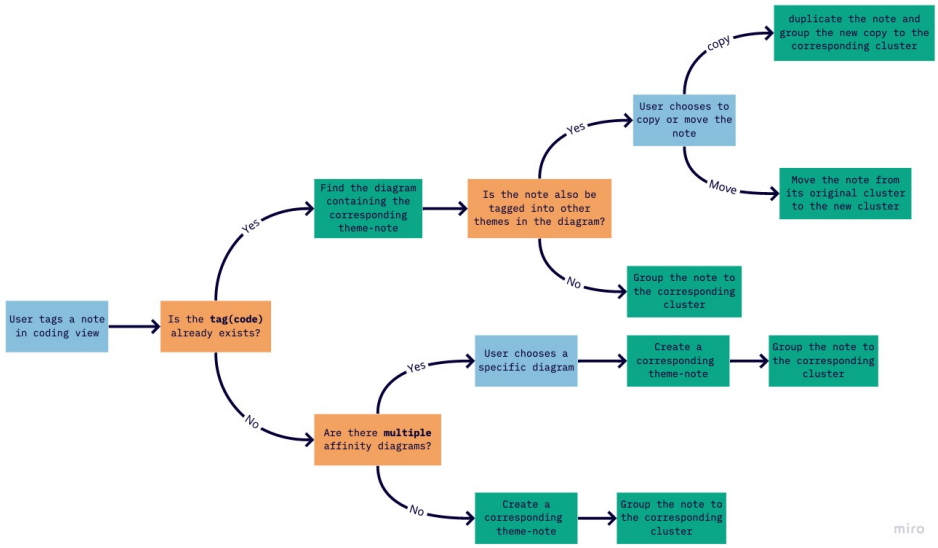


Figure 4.5. Flowchart demonstrates how a coding action is mapped to corresponding diagramming actions. Blue represents different kinds of input expected from user; green represents system reactions and outputs; orange represents the working logic.

indicate which, if any, diagram the tag should apply and creates a new theme note in that diagram. If the tag exists, the highlight's note is added to the corresponding diagram's cluster. If a note is tagged to multiple clusters in the same diagram, the user may choose between duplicating the note or moving it. Untagged notes as well as auto-generated theme notes in a diagram are added to a "messy pile" (fig. 4.3D) until the user arranges them in the diagram.

Users can also modify an existing tag directly from the coding view. This may lead to the following three situations: 1) change to a new theme which does not exist before. Similarly, ADQDA will confirm with user before creating a new theme note in the same affinity

diagram. 2) change to a theme which belongs to the same diagram, this will move the note from its previous cluster to the one just assigned to. 3) change to a theme which belongs to another diagram. Since our system assume that each affinity diagram represents a different lens to analyze data, or a different framework, thus we don't allow users to change a tag that initially belongs to one diagram to another.

4.4.2.4 Managing multiple analytic lens

ADQDA allows users to create multiple diagrams where each has a unique name, or assign multiple tags to one note or highlight. Yet multiple diagrams and multiple codes have differences: ADQDA hides cluster information generated in one affinity diagram from another to reduce distraction. With basic affinity diagram view, we have feedbacks and intuitions that users prefer being able to concentrate on a particular analytic lens, and to form up the analysis with other diagrams in a different stage. Whereas the coding view provides an overview on the cluster information, that means all tags are shown on the notes or highlights from the coding view.

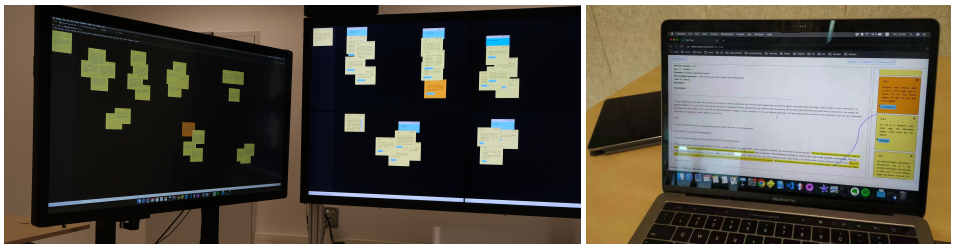


Figure 4.6. Two different affinity diagrams on two large displays, and a coding view opened in a laptop. Long press a note in any of them highlights notes with same content in the others.

Cross-linking among Multiple Views: Due to the multiple analytic lens and the mix between coding and diagramming, the same content can thus appears across different contexts and representations. ADQDA supports *"cross-linking"* ([**G3**]) to facilitate tracking and comparing notes in different views. As shown in fig.4.6, when long pressing a note, either in diagram view or coding view, it will highlight in all views to help users locate a specific note in multiple places. ADQDA also provides search functionality to help users find certain artifacts when they have some keywords in mind. They can perform a basic search in a single view or cross search in multiple views.

4.4.2.5 Distribute views across users & devices

So far, we have discussed how ADQDA can achieve the first three design goals ([**G1**][**G2**][**G3**]) to support flexible iterations among the various analytic phases, conceptual methods and multiple representations. As for design goals [**G4**][**G5**], ADQDA allows each view having multiple synchronized instances. These multiple instances can be opened in heterogeneous displays and be manipulated by multiple users both synchronously or asynchronously. The different views and

Figure 4.7. Analysts work in different contexts across the analysis processes, e.g., (a) coding in a laptop (b) collaborative diagramming in a wall-sized display (c) developing multiple diagrams from different analytic lens, with a mobile phone to check notes' contexts.

their instances constitute an integrated, holistic analysis system and can be extended based on users contextual needs.

Developed as a web application, users can access ADQDA through a single URL link, and choose a desired view or create new ones using the navigation menu. Multiple instances of the same view are synchronized. For example, moving a note in one view causes that note to move on the instances opened in other devices. Two users who access to the same diagram can thus follow movements in real time. A name tag appears below the note to indicate who is currently moving it.

ADQDA can be adopted for a variety of collaboration modes. We list a few examples to demonstrate how ADQDA could be appropriated under different situations: A single user can code the data with her laptop and arrange the notes on a larger screen (figure 4.7a); multiple users can collaborate on a same affinity diagram using a wall-size display (figure 4.7b); multiple users work on individual affinity diagrams by adopting multiple displays at hand and then put them side-by-side to compare (figure 4.7c). We develop the application scenarios in more details in section 4.5.

4.4.2.6 "Inboxes & Outboxes" for awareness

The synchronization-mapping and cross-linking mechanisms in ADQDA save users the efforts to maintain a consistent analytic progress among multiple representations, and facilitate the iterative transitions between them. However, they also bring challenges:

Awareness Issues across Multiple Linked Views: In an affinity diagram, changes can be triggered by other views, even in the absence of collaboration. A user might retag a snippet in the coding view but not realize that will alter its associated diagrams (the note changes from one cluster to another automatically). Or a user might make

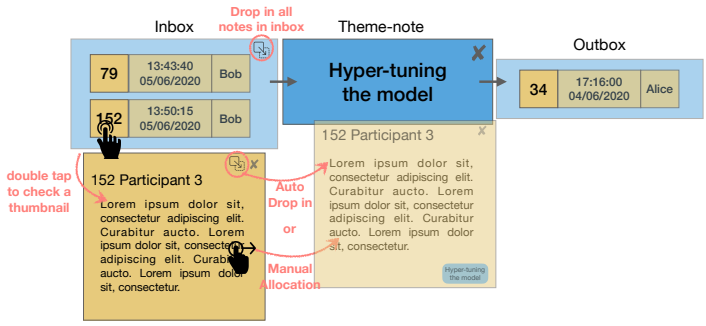


Figure 4.8. The "inbox, outbox" of a theme note. Users can track the updates caused by coding. They can expand a thumbnail to check the note, and manually drag it to the cluster or choose "drop in" to let the system automatically assign the position.

changes in another diagram hierarchy. When collaborating, it is more difficult for different collaborators to understand what happened. The propagated changes can introduce conflicts, such as when two users modify the cluster information of a same note in the same time, one using coding and one diagramming. In any case, users could potentially lose track of changes across views and collaboration.

Asymmetric Mapping between Coding and Diagramming: Though notes and highlights, and theme notes and codes are conceptually equivalent, the are not identical. Affinity diagrams are inherently spatial, whereas tags/codes are not. Coding an excerpt does not provide spatial information necessary to determine its notes' position in a diagram. It is possible to algorithmically position such notes, but this spatial arrangement also plays a part in the user's mental model. The "messy pile" on the left of a diagram works when new notes and theme notes come, yet it can not handle the tagging or re-tagging cases.

"Inboxes & Outboxes" as Solution: In ADQDA, each theme-note has an "inbox" and an "outbox", as shown in fig. 4.8. If a note is tagged to a theme from the coding view, instead of directly moving the note from one cluster to another, it will be put in the theme-note's "inbox" in a thumbnail format, with the time of modification and the user name to indicate who did it. Similarly, if a note is removed from that theme from coding view, it will be put in the theme-note's "outbox". Users can expand a thumbnail and further drag it into a specific position in the cluster, or just choose *"drop in"* option to let the system automatically put the note in the cluster. They can also perform *"drop in"* on a group of thumbnails in "inbox".

A stable spatial allocation is also helpful for users to maintain or recover their mental state. Furthermore, we don't want to impose a certain rule on the layout of notes, such as linearly spread out below a theme note. Different people might prefer different layout styles and give different conceptual meanings to them. With "Inboxes & Outboxes" mechanism, users can distinguish the updates caused by coding, and can flexibly choose between automation and manual allocation of notes' positions.

This "inbox, outbox" concept not only addresses the asymmetric mapping problem, but also helps the user keep track of changes in other views or by other collaborators.

4.4.3 ADQDA Implementation

We build ADQDA on top of Webstrates [14] using standard web technologies (HTML, JavaScript, CSS).

In an HTML document, elements on the interface are represented as node objects in a tree structure, called the Document Object Model (DOM). In webstrates, all changes made to the DOM get persisted on the server and synchronized to all connected clients by default. It

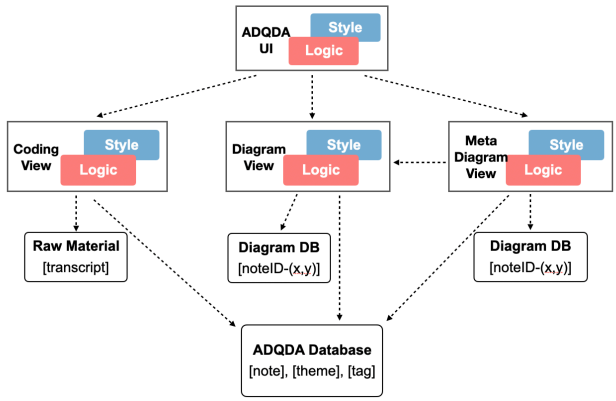


Figure 4.9. transclusion structure in ADQDA.

thus allows multiple users to collaboratively edit a document in real-time. We can locally control the look and behavior of a document by using scripts or style rules that do not affect elements in the DOM. For example, we can hide a note or a tag by inserting a "display: none" rule on that object to manipulate accordingly its appearance in multiple diagrams.

More importantly, we can compose documents using transclusion (a technique of dynamic embedding documents within another, where changes to the embedded document are reflected in the embedding document. The embedding document can access and manipulate objects in the embedded and vice versa). These concepts and mechanisms challenge the old application models and open up new opportunities to design tools to support more flexible and collaborative practices.

We adopt these techniques and generate ADQDA system based on a transclusion architecture, as shown in fig. 4.9. There are two kinds of webstrates(documents) in general, those implement the application logic, such as coding and clustering, and control the document's appearance, such as the color and position of a note (similar to the

view and controller concepts in MVC model); and those that store the analysis artifacts, such as notes, highlights, themes, in its DOM tree to ensure a real-time synchronization (similar as the model concept in MVC model).

To be more specific, there are six different types of basic webstrates: (1) "User Interface" webstrate that dynamically loads other webstrate instances in its interface based on user's choice of view through the navigation menu; (2) "Coding View" webstrate which contains the coding logic, controls the appearance for documents and notes panel. It further embeds the "Raw Material" webstrate and the "ADQDA Database" webstrate to store and update notes and tags information; (3) "Diagram View" webstrate implements diagramming logic, such as moving, clustering, and injects style rules to control the positions of notes in each diagram. It embeds the ADQDA database webstrate, and a diagram database webstrate which stores the positions of notes in a particular diagram; (4) "Meta Diagram View" webstrate embeds its lower-level diagram, and stores additional position information in the meta diagram's DOM. (5) "Raw Material" represents a list of documents to be analyzed; (6) "ADQDA Database" stores all analytic information, including notes, theme notes, tags, as well as the different kind of relationships among them in its DOM tree.

More webstrates can be generated by prototyping one of these basic webstrates. For example, multiple diagrams can be created by prototyping the basic diagram database webstrate, where each stores different positions and clusters information while sharing the same programming logic.

This transclusion architecture ensures the linking and mapping mechanisms of ADQDA. Analytic artifacts, such as highlights, notes, theme notes all have their unique id and attributes' values. By default, if a view embeds the Database, all information in the database will be shown. We dynamically inject different CSS style rules to selectively show information depending on the context. For example, in

the coding view, all notes are linearly presented, while in each affinity diagram, the notes' positions are changed based on the position information stored in each diagram's DOM. Each theme note has a creator attribute and a "themelevel" attribute to indicate which diagram it belongs to, thus it can be properly hidden or showed in multiple diagram views.

Furthermore, by using different kinds of mutation observers (which provides the ability to watch for changes being made to the DOM tree), every change in all views is recorded. Each view can make proper reactions to a change depending on different situations. For user identification, we use the embedded client management in webstrates to monitor users connected to any webstrate.

Considering the following use case: as shown in fig. 4.10, a user changes the current diagram named "trigger" to another diagram named "barrier" by clicking on the name list in the navigation menu. The ADQDA interface originally transcludes the diagram view which embeds the ADQDA database and the "trigger" diagram database.

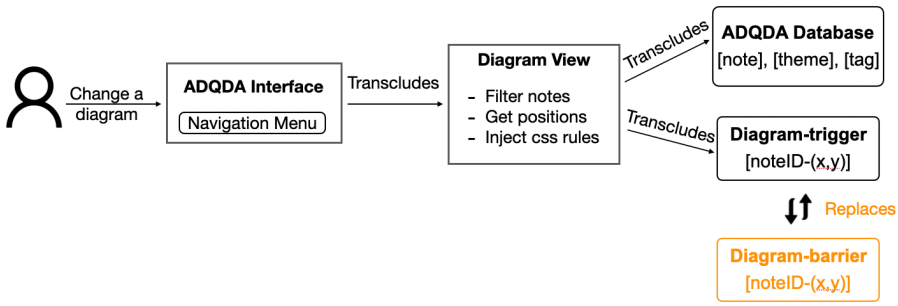


Figure 4.10. When a user switches the current diagram named "trigger" to another diagram "barrier", ADQDA dynamically transcludes corresponding diagram database to ensure proper representations of notes.

The interface detects the clicking input with the desired diagram name - barrier. It then changes the translusion source in diagram view to the desired one, by updating the "src" attribute of the corresponding <iframe> element. Once finish transcluding/loading, the diagram view filters notes and theme notes that belong to the "barrier" diagram with the aid of creator attribute of each note. It then gets the corresponding (x,y) positions of filtered notes from the "barrier" diagram database. Finally, diagram view injects proper style sheet rules to show notes in their correct locations and hide irrelevant information. Thus the two diagrams share same programming logic but show its proper analysis representation.

Another example demonstrates how two users can work on a same diagram with different devices, as in figure 4.11. User A and B both have the same diagram opened in their individual devices. When user A drags a note, diagram view detects the action and updates the style sheet rule on the position of that note. It further updates the (x,y) element stored in the underneath diagram database. Since changes to the database's DOM tree get persisted on the server and synchronized to all connected clients, the diagram view on the side of User B listens this change as well. The diagram view get the new (x,y) and update the location of note on B's side accordingly. Thus A and B see the

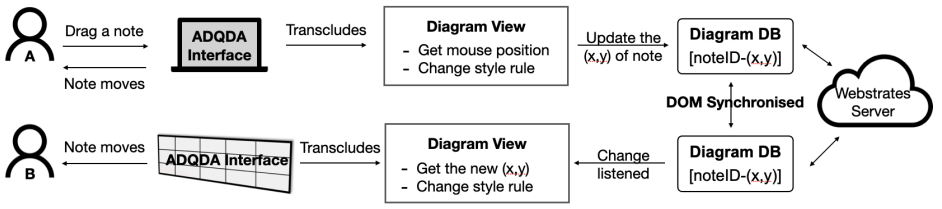


Figure 4.11. The different instances of same diagram database get synchronized to ensure two users collaborate on the same diagram with different displays.

same movements of note in real time.

4.5 Application Scenarios

ADQDA presents a vision and proof-of-concept implementation for affinity diagramming across the various analytic stages, conceptual methods, devices and collaboration modes. In this section, we use a set of application scenarios to probe the utility and novelty of the ADQDA prototype.

As discussed in section 4.2, there exist many related systems; we pick four representative systems of their types: Affinity+ [114], a research prototype for digital affinity diagramming; Distributed-Designer's Outpost [120], a digital whiteboard supports remotely organising notes in space; Miro[2], a commercial online collaborative whiteboard platform; and MaxQDA[4], a commercial computer-assisted qualitative data analysis software. We compare ADQDA with these current tools around the application scenarios and the design space to show ADQDA is able to solve previously unsolved problems, and reduce interaction viscosity during analysis [135].

Scenario 1. Alice and Bob meet in a lab room to analyze interview transcripts. The room has a wall-sized interactive display, an interactive table, and Alice and Bob's own laptops, phones, and tablets. They each open the transcripts on their laptops and begin coding them in ADQDA. They highlight pertinent snippets of the transcripts while they read. After going through a few participants, Alice moves to the wall while Bob continues to code transcripts. She loads the URL for their analysis session and creates a new affinity diagram, which automatically contains the extracted snippets in a pile in the corner of the display. While she organizes these snippets, Bob's new snippets

continue to be added to pile. Meanwhile, Charles, working from his office in another city, joins the analysis by opening Alice's diagram on his laptop. They work collaboratively, following each other's notes as they move around in real time on the shared diagram, Alice using her fingers and Charles his mouse. Charles decides he would rather not pan around in the diagram, so he walks down the hall to his studio and opens the diagram on its interactive board.

Discussion: This scenario illustrates multiple users working in different places and with different types of devices. Moreover, the users are able to choose the available devices best suited to their task at hand, extending the environment or switching devices by merely opening the shared session URL and choosing the shared artifact (transcripts or diagrams) to work on.

Comparison to Other Tools: Most tools, such as Affinity+, Miro, and Distributed-Designer's Outpost focus on only one type of analysis or artifact (coding or affinity diagramming). They may provide more refined interactions for these specific activities, but transitioning between these activities requires a cumbersome export-import and potential data transformation. MaxQDA does support both activities in the same tool, but it is neither collaborative nor does it support multiple device types or interactions beyond a single user on a single desktop system.

Similarly, few tools allow for users to bring different devices into an analysis. Affinity+ does let users add in multiple phones and tablets as personal input devices, but the main analysis view—affinity diagrams—can not be extended. Miro can be used on both mobile devices and larger displays, treats them as views of a single representation. We are not aware of any other tools besides paper that supports the ready addition of extra available displays to extend the analysis across phases and representations.

Scenario 2. Joining Alice on the wall, Bob realizes he is not sure of the meaning of one of the notes. He uses the menu on the note to send it to his phone, which he had previously registered with the session by opening its URL. His phone displays the pertinent transcript focused on the selected snippet. Meanwhile, Alice is not sure that a note is in the right category. She sends it to her tablet, finding in the transcript that the underlying meaning was misinterpreted without its surrounding context. She continues to read the rest of the transcript to make sure that the other snippets have not been similarly misconstrued. In the transcript, the diagrams' cluster information appear as tags on each snippet, so she can make sure that each tag makes sense. She re-tags several misidentified snippets and adds a few more snippets as she identifies new themes that had not emerged before.

Meanwhile, Bob and Charles continue working on their shared diagram. Alice's new notes now appear in the uncategorized pile on the diagram, while each of the notes she has re-tagged in the transcripts now appears in its new theme's "inbox" and its old theme's "outbox". Collectively, they walk through each of the items in inbox to make sure that they agree with this new categorization before either dragging the note into place in its new cluster, letting the system do so automatically, or sending it back to its old cluster.

Discussion: This scenario shows how analysts can smoothly mix different conceptual methods and iterate between different phases of analysis, from coding to diagramming and back again, throughout a given analysis session. Each analytic artifact reveals different contextual information through its spatial construct that the analyst needs to be able to fluidly navigate as the analysis evolves. Similarly, different analyses may be better suited to different devices, whether ephemeral in nature or more focused.

Comparison to Other Tools: Tools that focus on a single type of analysis cannot provide support for such iterative analyses. They may be able to provide links back to transcript context, as in MaxQDA,

but such references are effectively read-only. MaxQDA does support multiple analytic artifacts, but the analysis is effectively linear, with coding preceding affinity diagramming: changes made in coding alter the spatial positions of affinity diagrams. As one user in one of MaxQDA's own tutorial videos comments, "when MaxQDA brought the new coding results into the project, it re-sorted all the groups into alphabetical order for reasons I don't entirely understand."

Scenario 3. Some days later, Alice, Bob, and Charles decide to re-analyze the data, focusing on how participants use their artifacts rather than just on the types of artifacts under study. Alice creates a new affinity diagram from the same set of transcripts, which loads all of the notes into the initial pile. The three of them collectively build their new diagram, Alice and Bob in their lab and Charles in his home office. Once the diagram has taken shape, Alice and Bob gather around the table in their lab and load the initial diagram. As they select notes in the new diagram, they highlight in the initial one, helping them to compare the relationship between the different analyses. Charles does not have a board in his home office, so follows along in the transcripts view on his tablet. As Alice and Bob highlight notes in their diagram, the corresponding snippet highlights in Charles' view, showing any alternative tags it may have from the other diagrams.

Discussion: This scenario shows how analysts can carry out parallel analyses of their data from different analytic lenses, but still maintain a link between them. The analysts can treat these analyses as independent, focusing on just the one, as we see in the first part of the scenario, or they can compare across them, as in the second part. To help facilitate such comparison, the different users can bring in the different available devices to extend the space for their analysis. Similarly, they can collaborate asymmetrically, as when Charles works along from the transcript view.

Comparison to Other Tools: While most, if not all, tools allow the creation of multiple diagrams from the same data, they are typically treated as independent diagrams. As such, iteratively going back to a different phase of analysis, such as re-coding the data, would require manually updating the diagrams accordingly. MaxQDA does support associating multiple diagrams to the same data, but, in addition to the limitations shown in Scenarios 1 & 2, can only show one diagram at a time.

Scenario 4. Alice, Bob, and Charles agree that both diagrams reveal essential facets of the collected data. Alice and Bob begin a meta-analysis of the first diagram, looking for higher level concepts. They create a new meta-diagram, which transforms theme notes from the first diagram into regular notes in the new diagram. They further import the theme notes from the second diagram, allowing them to combine the themes from both analyses into a single higher-order analysis. As Alice is combining two notes into a common theme, she wants to make sure she is correctly interpreting the theme's label. She first expands the note to reveal its associated snippets to confirm before them grouping the two themes together. Meanwhile, Charles continues refining the second diagram on the board in his studio. To ensure a shared understanding of both diagrams and to follow along as his themes are organized into new higher-level concepts, Charles turns on the "show meta info" mode.

Discussion: The first part of this scenario illustrates how users can transition between analytic phases, while still maintaining links to higher or lower levels of abstraction. The second part of the scenario illustrates how multiple users can simultaneously iterate on different analyses at different levels of abstraction.

Comparison to Other Tools: We are not aware of any other tool that provides explicit support for meta-diagramming activities across

analytic phases. MaxQDA does support meta-diagramming, but for one device and one analysis at a time. Other tools support such analysis by extracting themes and creating a new diagram; as such they would be treated as independent artifacts.

4.6 ADQDA through the Lens of Alternatives Framework

Qualitative sensemaking involves a flexible mix of alternative methods, multiple representations, and the adding or removing of multiple diverse devices across three main analytic phases.

ADQDA considers alternatives at all three abstract levels - the cognitive, artifact, and execution alternatives. Cognitive alternatives are indirectly supported: they are often off-loaded and externalized through the various representations. ADQDA lacks direct support for cognitive alternatives, such as memos or annotations to record the reasoning process. Yet users could potentially appropriate theme notes for the same kinds of functionalities. ADQDA provides support for a variety of artifact alternatives, including highlights and notes, tags and theme notes, the different representations of them, as well as the heterogeneous displays. Some of these artifact alternatives are introduced by alternative execution methods, coding, and diagramming.

In terms of the "degree of attention", we can view the analysis in terms of different granularity levels: under different frames of reference, the active analysis focus can vary from a single artifact, such as a highlight, a note, to a group of artifacts, such as a cluster in a diagram, and to the entire diagram or even a combination of them. When a user focusing on categorizing a single note or highlight, it acts as a choice. The different clusters are multiples, while several are options

in case the user considers among which this note should belong. Once choosing a cluster, this cluster becomes a choice. After being arranged into a proper position inside the cluster, the note turns to a multiple as the user continues to pick another note.

The user might also focus on a particular cluster or theme, and goes through multiple notes, highlights, or even the raw materials to check which data item is of relevance. In this case, the cluster is a choice and the different kinds of data items flow between multiples, options and potentially become a choice. Moreover, the user could also focus on the entire diagram or a combination of them as a whole, such as when considering different mental models or constructing higher-level concepts. Here each singer artifact is a multiple since no special attention is paid to them. The different diagrams flow between multiples, options, and choices as the user changing the active view or switching attention from one display to another.

In the proposed alternatives framework, five high-level processes around alternatives are identified: generating, updating, reasoning, reducing, and managing. In ADQDA, the creation of alternatives is achieved by both users and the system itself. Users manually generate multiple diagrams and appropriate different devices. While the system automatically produces pairs of identical analysis artifacts for the alternate methods - coding and diagramming. Similarly, Updating a diagram requires users to manually moving notes around, while analysis artifacts related to coding and diagramming get updated automatically for mapping synchronization.

The asymmetric views, the cross-linking mechanism between views, and the fact that ADQDA can be easily deployed to combinations of devices facilitate reasoning alternatives, such as to compare two diagrams. Alternatives get reduced as users remove certain diagrams or clusters, or merge multiple diagrams into one. As affinity diagramming is highly subjective, ADQDA does not implement any algorithms to reduce the number of alternatives automatically. Considering man-

aging alternatives, ADQDA acts as a repository for all analysis artifacts and representations, and further provides linking and mapping mechanisms to maintain the interdependency among different kinds of alternatives.

4.7 Discussion

In this section, we present other design alternatives considered. We discuss the limitations of our work in terms of the prototype itself and conceptual limitations.

4.7.1 Design Options Considered

While designing ADQDA, several alternative solutions have been considered:

Integrating or Separating Hierarchical Diagrams: Instead of separating phase 2 and 3 into affinity diagram view and meta-diagram view, we could also put them in one representation, such as using a tree-like structure as in Klemmer et al.'s designers' outpost system [117], or other free structured diagrams where users can draw links between notes.

Through our interviews and experience, we find several reasons that practitioners prefer the separated representation, especially with relatively large amounts of notes. Firstly, it is hard to find a wall space to present these levels together. Secondly, in a tree structure, the operations are cumbersome when moving a note with all its subtrees. Thirdly, users might want to keep focused in each analytic stage, too much information is overwhelming and may distract their

attention. Last but not least, we might have users work on different phases at the same time, this separation can facilitate this type of work division. Based on these observations, we decide to design the basic diagrams and their meta diagrams as two different views.

In fact, each option is appropriate for certain analysis scenarios. This separation of views may not make sense with a smaller amount of notes. The all-in-one view can better provide an overview of the analysis, and a more flexible level assignment. In the current system, we have not yet implemented other types of views, but ADQDA could be extended to complimentary views in future work, such as through a plugin. Users should flexibly choose the analytic artifacts and representations as they see fit for their situated task or analytic goal. These artifacts should be a linked part of the entire analytical environment.

Tracking the Context of a Note: To trace the original context of a note when diagramming, ADQDA shows the source document on an extra display, such as a mobile phone or a tablet. To facilitate collaboration, users can further choose to send the content to other collaborators' devices, either in the same place or at distance.

Instead of introducing extra devices, we initially designed it as a pop-up window nearby the note. Users can further choose whether to synchronize the window with other remote instances. We find each of these two options has its trade-offs. The pop-up window saves users the efforts to reach a new device that needs to be previously registered. However, diagramming with a large amount of data items is highly information-intensive, where users often struggle with a lack of space. Pop-up windows could potentially obscure other notes nearby. By adding extra displays, the diagram stays the same and users can access both information by just switching their attention. It also seems similar to the natural practices with paper artifacts, where users grab related documents when needed and leave them by side while not in

use. ADQDA could be implemented to support both options and let users choose the way they want based on their contexts.

4.7.2 Prototype Limitations

The primary focus of the ADQDA prototype is to support qualitative data analysis through affinity diagramming by (1) appropriating available interactive devices to create ad-hoc cross-device analysis environments, and (2) providing various linking mechanisms to help track and coordinate alternative artifacts across different analytic phases and representations. As such, support for coding and affinity diagramming could be considered a minimum viable product: it provides basic support for coding textual transcripts and for rearranging snippets and groups of snippets, clustering, and annotating snippets in affinity diagrams.

Richer interactions for multi-touch gestures [136], provenance and history tracking [114], or large-scale interaction techniques such as Bring and Go [137] or portals [138], etc. are not currently implemented. There are currently no additional visualizations or color coding to convey, for example, the stability of clusters or to reinforce participant numbers or participant roles (e. g., managers vs. executives vs. workers).

While the implementation is designed with collaboration as an explicit design goal, there is little support for in-system coordination. All such coordination must currently be done out-of-band using teleconferencing or using ad-hoc social conventions (such as jiggling a note to convey a shared focus).

Finally, ADQDA is implemented using Webstrates [14], a research prototype environment. It can handle around a hundred simultaneous connections and tens of simultaneous edits to the same shared document. It is built on web technologies with their associated constraints.

4.7.3 Conceptual Limitations

Beyond the technical limitations of the current prototype, there are also conceptual limitations in the current approach and design, especially related to collaboration, device heterogeneity, and conceptual compatibility between different views.

Our system supports various collaboration modes, multiple users can be colocated or remote, on the same device or different devices, simultaneously or asynchronously. Conceptually, however, different interactions are better suited to each of these scenarios. For example, a technique that works well to highlight a given note across multiple views for a single user across multiple devices might distract or confuse other users. Similarly, different incompatible interaction techniques might be better suited to multiple users working on the same wall vs. multiple users collaborating across distant walls. Furthermore, the various collaboration scenarios might also rely on different kinds of awareness mechanisms [139, 140].

Similarly, the different views are currently only minimally adapted to the constraints of the device. For example, displaying a wall-sized diagram on a smaller display such as a laptop or a smartphone currently uses a mix of panning and zooming. It is both useful and possible, but awkward. Moreover, it assumes that the relative dimensions of the canvas will not vary across devices. For input, touch and click events are currently treated as equivalent, but little additional support is currently made to handle the different input modalities of different devices.

Finally, the underlying design of the approach assumes that it is possible to define a consistent mapping to maintain dynamic links between all related analysis artifacts. For example, a note in a transcript must maintain a link to its snippet in an affinity diagram, but under certain scenarios such a bijection may be untenable or impossible to

maintain across different layers of abstraction.

4.8 Future Work

In this chapter, we identify the kinds of alternatives within the specific context of qualitative data analysis through affinity diagramming. We present our vision and a proof-of-concept prototype, ADQDA, to explore how analysts can appropriate available digital devices as they fluidly migrate between analytic phases or adopt different methods and representations, all while preserving consistent analysis artifacts.

We have previously discussed ADQDA in terms of its support for alternatives, the prototype limitations, as well as the conceptual limitations. In this section, we discuss several directions for future work.

Conducting Empirical Evaluation In section 4.5, we validate ADQDA through a set of application scenarios. We show the different kinds of alternatives supported in ADQDA and how this support leads to more flexible sensemaking practices. This work could be further evaluated through various empirical studies to gather real user feedback on the design concepts.

For ADQDA, it is insufficient to consider the novelty of individual problems or individual technological solutions, nor are we aiming to provide empirical evidence that the specific design decisions or features in the prototype are *the* right solutions. Rather it is the vision where users flexibly flow between the integration of alternative tools, artifacts, representations, and displays that challenges the traditional application model for handling data analysis tasks.

Letting practitioners using ADQDA in their real analysis projects could bring inspiring observations and valuable feedbacks. Especially, we are interested to see how they appropriate ADQDA to create their

own analysis environments and workflows. Even with a small number of participants, we could potentially see the use of ADQDA in a variety of ways. We can also conduct laboratory studies to compare ADQDA with other existing analysis tools, such as MaxQDA or Mural, for given analysis tasks or goals. Instead of gathering quantitative measurements, we are more interested in qualitative feedback such as what features or functionalities of ADQDA they consider helpful or not helpful in the sensemaking process.

Enhancing support for alternatives Though ADQDA provides good support for alternatives at different abstraction levels and the high-level processes around them, it could be further improved in different aspects.

First, instead of creating a new diagram each time, generating alternative diagrams by forking or branching an existing diagram might be helpful. Better alternatives management might be achieved through various kinds of versioning techniques. To facilitate the practices around reasoning alternatives, the current version only supports tracking notes one by one across diagrams, further versions could add more features to help to compare, such as to show all intersections of two diagrams. Besides, the current linking mechanism follows a strict match. Yet fuzzy match might be helpful to compare data items with high similarity.

Apart from the current coding and diagramming views, ADQDA should allow extensions of other kinds of views. For example, to provide more explicit support for cognitive alternatives, views with free-form drawings, annotations, and memos could be added to preserve thoughts and ideas.

Enhancing Collaborative Sensemaking ADQDA only provides a rudimentary implementation for collaborative sensemaking. For ex-

ample, the synchronization among multiple instances of the same view provides basics for remote collaboration. Yet ADQDA has not put special attention to any particular collaborative scenario and design interactions and features to support them. As said, the "cross-linking" feature to track notes across multiple views for a single user might distract or confuse other users. How to balance the features that facilitate sensemaking for a single user and those for multiple collaborators remains an open issue.

When collaborating, the distinction between personal and collaborative space is proved important [108]. In ADQDA, except for the document view, all other views are treated as shared space where updates by all users get synchronized immediately. It could be further developed to provide this separation when needed, such as letting two users individually highlight transcripts before presenting these highlights for cross-validation.

CHAPTER 5

Computational Transclusion - Managing alternatives in the context of reuse

This chapter explores how to design tools to manage alternatives in the context of reuse. Analysts often reuse contents generated in computational notebooks to other places like presentation slides. We consider the original content and its derivative as "alternatives" that diverge from the moment of reuse. Based on various needs during sensemaking, these alternatives might be applied with the same updates or be modified differently. We introduce "computational transclusion" as a novel reuse approach that maintains the links between pairs of alternatives to facilitate tracking and coordinating changes. We further explore "computational transclusion" in the context of transcluding data visualizations.

Section 5.1 introduces our motivation for envisioning "computa-

tional transclusion". Section 5.2 reviews related research, including content reuse practices during sensemaking processes; the original concept of transclusion; and other existing reuse techniques. Section 5.3 explores transcluding data visualizations by examining various reuse scenarios using a sandbox system. Section 5.4 discusses the identified transclusion properties and how to reify them in the user interface to enable flexible (re-)configurations of transclusion. Finally, section 5.5 & 5.6 discuss the limitations of our work as well as what can be done in the future.

5.1 Introduction

Computational notebooks have been widely adopted by people from various domains, such as science, finance, and education, to perform data exploration tasks [141–144]. By integrating narrative texts, code pieces, data frames, and interactive visualizations in one single document, computational notebooks support both experimenting with ideas and presenting analysis results.

Rule et al. [26] point out that a single notebook can hardly support the overall sensemaking process: a notebook for exploring ideas often contains "messy" code that is hard to understand; while a notebook for presenting results often lacks of the details of the analytic process. Analysts thus often apply multiple notebooks in a single analysis project to support various analytic tasks; other environments, such as presentation slides or note-taking software, could also be adopted to compensate for the inadequacy of notebooks.

Contents that are initially generated in one notebook could be reused to these other places for different purposes. For example, a visualization could be moved from a "messy" notebook to a clean-up version to present analysis results. We consider the original content

and the derivative (or reused content) as "alternatives" that diverge from the moment of reuse. Instead of each being an independent entity, the alternatives often hold complicated links based on users' mental models during the sensemaking process. For instance, considering reusing a visualization, analysts might customize the styles for each visualization to suit different color themes but synchronize the data-binding logic to involve newly-collected data in both visualizations. Sensemaking is often messy and iterative, analysts often need to go back and forth between these places to update them or to coordinate changes.

Existing technologies provide a variety of ways for content reuse. For example, users could simply perform "copy & paste" commands or "export & import" to move an artifact across different places. However, these techniques tend to treat the alternatives as individual copies, leaving users to apply ad hoc strategies to track and coordinate changes that might appear on both sides. New techniques have also been invented to facilitate reusing portions of content from computational notebooks. For instance, the "embed cell" technique in Observable[1] allows users to put working contents of notebooks, like visualizations, inside another website. Despite that "embed cell" can keep the reused contents as updated as the original one, it is difficult for users to make changes from the reused side or to alter the relationship between the two alternatives based on their situated needs.

Back in the 1960s, Ted Nelson pictured a digital world where multiple instances of the same content dynamically connect as the way we think and use them [76]. Instead of each a separate copy, the instances across different places can be tracked and synchronized with one and another. He coined the term *transclusion* to describe "reuse with original context available, through embedded shared instancing" [77]. Built

[1] `https://observablehq.com/@observablehq/introduction-to-embedding`, accessed on 21-04-2021.

upon his idea, we believe that the original content from computational notebooks and its derivative in another place should stay connected as the way users expect. While different from the original "transclusion" that keeps the *same* content, we explore the asymmetric reuse where both the original and the reused content could evolve independently as the analysis continues.

We envision "computational transclusion" as a novel reuse approach that maintains various kinds of links between pairs of original and reused contents to facilitate tracking and coordinating changes in asymmetric reuse cases. With "computational transclusion", analysts can carry on these alternatives as they move back and forth between multiple sensemaking stages and different analysis places. Users can also flexibly alter the way the alternatives are linked based on their contextual needs during the sensemaking process.

In our current work, we explore "computational transclusion" in the context of transcluding data visualizations. Analysts often explore various kinds of data visualizations and interact with them to gain insights [145–147]. Visualizations can also help share insights, communicate findings, and present analysis results [26]. In the transclusion context, we refer the original content as being *transcluded* and the reused or derived content as *transcluding*. Our research questions are: (1) *What are the components of computational transclusion of visualizations?* (2) *What are the various kinds of links users might want between the transcluding and the transcluded?* and (3) *How to reify these links in the user interface to enable flexible manipulations of transclusion?*

To answer these questions, we explore various reuse scenarios using a sandbox system built upon the Webstrates platform [14] (section 5.3). Through our explorations, we explore the dependencies between different components of visualizations and how the alternative contents might be linked. We propose six properties of transclusion — *linking, role of artifact, application of changes, direction of propagation, linking*

granularity, and *linking depth* — to help clarify different kinds of links between the transcluding and the transcluded (section 5.4.1). We also explore how to reify the identified concepts in the user interface to enable flexible and fine-grained control over transclusion (section 5.4.2). Finally, we discuss the limitations of our work as well as what can be done in the future (section 5.5 & 5.6).

5.2 Related Work

Our work builds upon previous research on practices around artifact reuse in computational notebooks; painpoints of existing reuse techniques; and novel reuse concepts and techniques.

In this section, we first introduce computational notebooks as a medium for data workers to conduct sensemaking activities. We review the reuse practices conducted around computational notebooks. We then discuss existing reuse concepts and techniques around three groups of work: (1) the original concept of "transclusion"; (2) techniques like "intelligent copy & paste" or "linking & embedding" which considers the links between the original and the reused content; and (3) coordination of multiple instances of visualizations as distributed in different devices.

5.2.1 Sensemaking in computational notebooks

Computational Notebooks Dating back to Knuth's idea of "literate programming" in 1984 [64], computational notebooks consider narratives as first-class citizens as the code to prompt the capture of the reasoning behind the analysis processes.

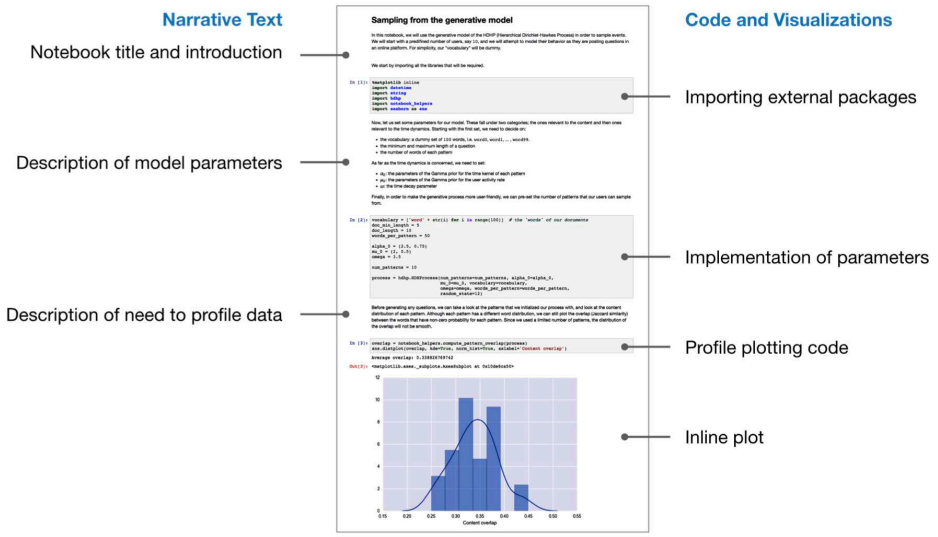


Figure 5.1. A computational notebook combining narrative text, code pieces, and the visualization result [26].

Computational notebooks integrate code, text narratives, as well as analysis outputs such as data frames and visualizations in a single document. As shown in figure 5.1, a notebook is based on a linear collection of units called "cells", each of which contains text or code that can be executed to compute results or generate visualizations.

There are many examples of today's computational notebooks, either as research prototypes or commercial products. For example, Jupyter Notebook[2], as an open-source web application, has been adopted by millions of users. Google Colab[3] enables synchronous editing of a notebook to facilitate collaboration. Observable[4] tracks the

[2] `https://jupyter.org/`, accessed on 29/04/2021
[3] `https://colab.research.google.com`, accessed on 29/04/2021
[4] `https://observablehq.com/`, accessed on 29/04/2021

dependencies between cells to facilitate code exploration and management. Codestrates [88] makes it possible to change the notbook environment from within. After analyzing 60 notebook systems in both academia and industry, Lau et al. [141] characterize these notebooks across four major stages of sensemaking: importing data, editing code and prose, running code to generate outputs, and publishing notebook outputs.

Artifacts Reuse around Notebooks Studies around computational notebooks show the general phenomenon of artifacts reuse. Rule et al. [26] find that analysts often "reuse snippets of code from prior or others' notebooks." They also "copy analytic results to other media for sharing and presenting." Kery et al. [8, 46] observe that their participants use copy & paste to create code variations or transfer "most successful parts" to a new notebook. Chattopadhyay et al. [148] find analysts continuously copy-paste code to customize visualizations. Subramanian et al. [149] find that data workers tend to move code segments across multiple applications.

However, the constant copy-pasting is considered frustrating. Analysts might "lose their mental model of the task" as a result of excessive switching between clones [149]. The excessive cloning also makes it more difficult to maintain the code. Chattopadhyay et al. [148] further point out that reusing small portions of a notebook out of the box is challenging because of the intertwined dependencies among cells. Focusing on the dependency issue, Head et al. [27] contribute the Code Gathering tools, as extensions to computational notebooks, to help analysts identify a minimal set of content dependencies that generate a specific result.

More generally, studies around laboratory notebooks show analysts' work as a complex web of interrelated repeated content. Tabard et al. [150] identify that biologists' notebook content is "repeated either

over time or related from one stream to another." Klokmose and Zander [151] report physicists' vision on future lab notebooks — flexible media that can be broken into pieces and be brought to other places. They give the example of reusing control instruments from a notebook to a mobile phone, where changes on the mobile side can be reverted to the original notebook.

Our work builds on these observations of content reuse around notebooks. We focus on asymmetric reuse cases where the original and reused content could be modified alternatively for different analytic goals. Based on explorations of various scenarios, we review different kinds of mental models that users might hold on the relationships between the original and reused content.

5.2.2 The original concept of transclusion

The network of repeated information contents was examined by Ted Nelson since the early 1960s. He argued that instead of emulating physical paper to handle information in a digital world, we should rather take advantage of computing and rethink the structure and model of texts in a digital context. He envisioned computing systems can help create, manage, and share information contents that align naturally with human thought and creativity.

Ted Nelson coined the term "transclusion" to describe "the same content knowably in more than one place" [76], or "reuse with original context available, through embedded shared instancing" [77]. Rather than separated individual copies, transclusion treats same contents over places as shared instances. It establishes a permanently live connection between fragments. As a result, these contents can be retrieved and changes made in one place are reflected in others as well.

Ted Nelson also discussed interesting application scenarios for transclusion, including creating alternative versions or alternative organi-

zations of documents; transcluding alternatives side-by-side to compare them; or creating composite documents [87, 152]. More recently, Krottmaier and Helic [153] demonstrate three application contexts of transclusion: in electronic publishing, since scholar papers often quote contents from published ones; in discussion forums to achieve more efficient communication; and in organizing courses materials, which are inherently related. They further discuss issues of transclusions, such as intellectual property — how to make the transcluded part visible as a "reuse" not a "stolen" artifact; and how to handle the changes made in the transcluded. Bernstein [154] proposes three different types of linking for transclusion: montage, transformation, and collage. Di and Lumley [155] focus on the transparency of a transclusion to both users and applications, as well as the operations users are allowed to perform.

Transclusion challenges how digital contents are organized and represented, as well as the way we manipulate and interact with them. Klokmose et al. [14,156] further apply transclusion in web technologies to generate "malleable, shareable, and distributable" software systems. They create the Webstrates platform, where different web documents, representing data or interaction logic, can be flexibly composed to handle different kinds of tasks. Webstrates dissolves the isolation between individual applications. Tchernavskij [78] defines this as "software transclusion" — "software systems as networks of information substrates applying transclusion as a fundamental composition mechanism and user interaction." Tchernavskij further discusses the potentials and challenges of this approach around three themes, reconfigurable software, asymmetrical collaboration, and ecology of software artifacts.

The concept and vision of transclusion inspires us to explore the interconnections between alternatives in the context of reuse. We propose "computational transclusion" as a novel reuse approach that links contents originally from computational notebooks to other reused

places like presentation slides. “Computational transclusion” can help users to track and coordinate changes between the alternatives in a flexible and fine-grained manner.

5.2.3 “Intelligent” copy-paste, linking & embedding techniques

Transclusion emphasizes permanently live connections among the same contents. Yet in other reuse cases, the original and the reused contents could be modified to different alternatives and might need different types of synchronization mechanisms.

In this section, we discuss reuse techniques around “intelligent”copy-paste and “linking & embedding” which provide various kinds of linking mechanisms between the original and the reused content.

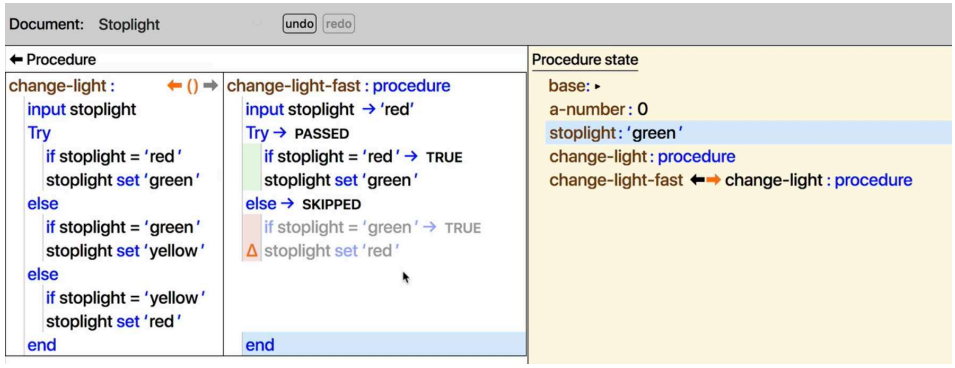


Figure 5.2. An example of “managed copy and paste” [157]: “change-light-fast” is copied from the “change-light” and modified to a variant. The system automatically links this copy to its origin. Using the pair of arrows in the interface, users can pull differences from the origin to the copy and vice versa.

Acknowledging the carrying cost of maintaining code copies in programming practices, Edwards introduces "managed copy and paste" [157]. As shown in figure 5.2, the system automatically tracks code copies and links them to the origin. For example, the function "change-light-fast" is copied from the "change-light" function and modified to a variant. Edwards reifies the links in the system's interface as a pair of arrows. The colors of arrows indicate whether there are new updates in each side. By clicking these arrows, users can pull differences from the origin to the copy and vice versa. This design saves users from "subsequently tracking down all copies and fixing them" during their coding practices. Our reification of transclusion as discussed in 5.4.2 applies similar design, but focuses on the granularity of cells and provides different models of synchronization.

Google Colab[5] provides a "code snippets" library where users can search and reuse well-established functions from other notebooks. Users can insert a code snippet directly into their own notebooks and track back to the source notebook in any time. Users can add their own code pieces to the Google Colab library as well [6]. Codestrates Packages allows users reusing both code functionalities and interface templates [158].

Observable develops various cell reuse techniques. The "import cell" allows users importing any named cells from other notebooks[7]. Compared to copy-paste, "import cell" automatically loads the dependencies of the cell to ensure its functionality in the new environment. By default, users get the most up-to-date version, they can choose to import a certain version of the cell as well.

"Import cell" is limited to reusing cells within the notebooks environment. Due to the large demands on reusing contents from note-

[5] `https://colab.research.google.com`, accessed on 29/04/2021

[6] urlhttps://www.youtube.com/watch?v=rcXrH8euKNA, accessed on 29/04/2021.

[7] `https://observablehq.com/@observablehq/introduction-to-imports`, accessed on 29/04/2021.

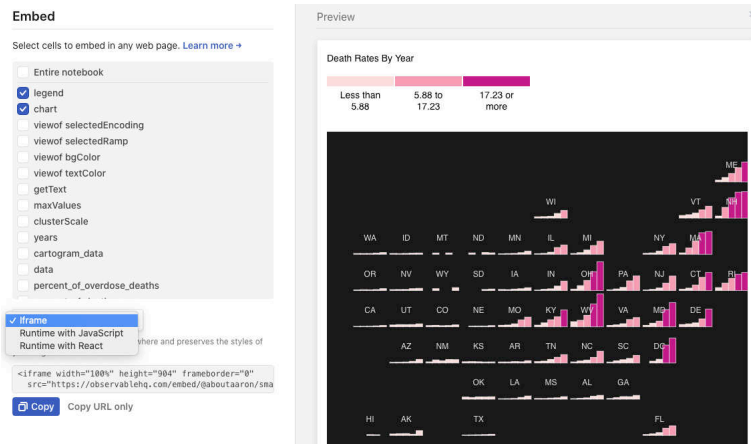


Figure 5.3. An example of "embed cells" feature in observable. Users can choose a combination of cells in one notebook to reuse in other websites. Taken from a published notebook [9].

books to other places, the "embed cell" is invented[8]. As shown in figure 5.3, every notebook has an "embed cells" option that lets users choose which contents to embed and which embedding methods to apply[9].

These techniques, implemented in different notebook systems, facilitate reusing portions of contents within or outside the notebook environments. Yet, they only provide basic tracking functionality that enables users to track back to the origin from the reused side. No further functionalities are provided to help users be aware of and manage the changes as made on both sides.

Besides the notebook environments, different reuse techniques are

[8] `https://observablehq.com/@observablehq/introduction-to-embedding`, accessed on 29/04/2021.

[9] `https://observablehq.com/@aboutaaron/small-multiple-chart-cartogram`, accessed on 29/04/2021.

also invented to facilitate reusing and coordinating contents across applications. Microsoft invents the "Object Linking and Embedding" (OLE) mechanism to allow objects created in one Microsoft application to be reused in another. Considering reusing a table generated in a Excel worksheet to a Word document, the user can either paste this table as an individual object or "paste link". For the former option, the reused table can be modified and the changes won't influence the original content. Whereas the "paste link" option inserts an image of the object and links to its origin. Users cannot modify the table directly on the reused side [10]. Once finishing updating the contents in the original side, the changes will be automatically synchronized in the copy.

Google deploys another linked-copy mechanism across its applications. As show in figure 5.4, the table and the graph are initially created in a Google spreadsheet and are linked-copied to this slide. When the objects get updated in the source document, update icons

[10] `https://www.youtube.com/watch?v=fDCF2elhMOE`, accessed on 29/04/2021.

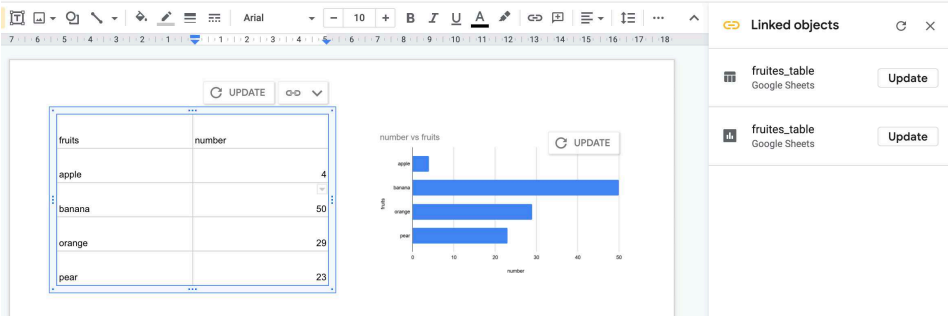


Figure 5.4. An example of linked objects in Google Slide. The table and the graph are initially created in a Google spreadsheet and are linked-copied to this slide. When the original objects get updated, update icons appear in the copy as a reminder for synchronization.

appear in the copies to indicate the changes. Users can choose to synchronize the local object with the origin or ignore these changes to keep its own version. Users can make changes in the reused contents, however, these changes are at risk of being overwritten when synchronizing to the origin.

These "linking & embedding" mechanisms in Microsoft and Google applications consider not only track between the original and reused content, but also synchronizing changes between them. However, unlike "computational transclusion" which allows users to flexibly define the type of synchronization they need, Microsoft and Google impose certain mechanisms which cannot be changed by end-users.

Our work also proposes six properties of transclusion that help to untangle the concepts involved in these different linking models. More detailed discussion on the differences between computational transclusion and the reviewed reuse techniques can be found in 5.5.

5.2.4 Distributed visualization & visualization sharing

Distributed or Shared visualizations also encounter the problem of managing and synchronizing changes across multiple instances of a visualization.

Badam and Elmqvist [159] propose a web application framework called PolyChrome to share and synchronize web visualization in collaborative settings. PolyChrome explores different interaction distribution mechanisms. As shown in figure 5.5, operations on visualization can be shared: (1) explicitly, where users decide when to share, which operation to share, and share to whom; (2) implicitly, where all operations are shared automatically with connected devices; or (3) unilaterally, where a leader shares its operations to other devices.

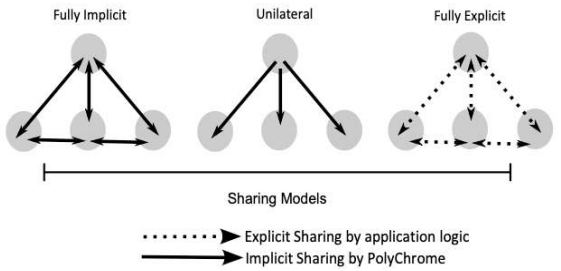


Figure 5.5. Hypothetical operation sharing models in PolyChrome [159]. Explicit model allows users to define application logic for sharing interaction (when, which, and to whom); implicit model shares operations automatically to all the others; unilateral model declares a leader and share automatically the operations on the leader to the others.

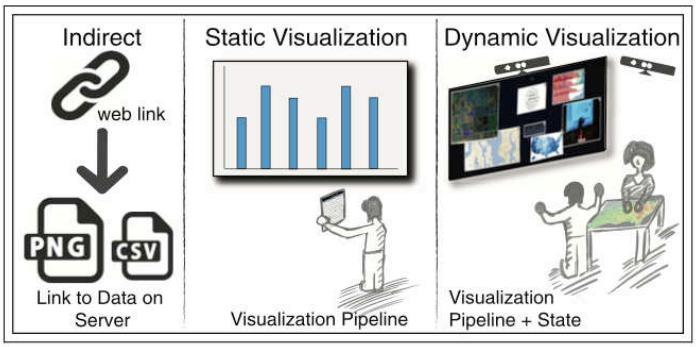


Figure 5.6. Three levels of visualization content transfer in Visfer [160]. From left to right: transferring data, visualization pipeline, and dynamic state of the visualization.

In another study by Badam and Elmqvist [160], they create Visfer framework to help transfer a visualization from one device to another by scanning the QR code embedded in the visualization. This technique can be applied in various reuse scenarios, like "allowing students to extract a visualization from the presentation and explore it on their personal device." As shown in figure. 5.6, they consider three levels of content transfer: data, visualization pipeline, and dynamic state. At the basic level, the QR codes link to the data driving the visualization; the second level is the visual representation, or rather the pipeline that generates it; the third level considers the visual representation as well as its dynamic state.

McGrath et al. [161] propose a Branch-Explore-Merge protocol to help users transition back and forth between collaboratively working on a shared visualization and individually exploring alternative paths. Branch-Explore-Merge allows users diverging from a shared visualization (branch) to explore data independently on their own devices (explore), and then propagate back their findings into the original display (merge). To resolve potential conflicts during the stage of merge, they apply different voting policies.

To facilitate remote collaboration on visual data exploration, Schwab et al. invent VisConnect, a web-based synchronous distributed collaborative visualization system [162]. Based on the typical update cycle of

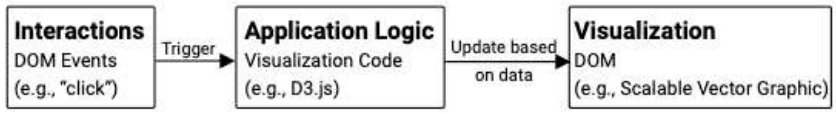


Figure 5.7. The typical update cycle of data visualizations [160]. When a user interacts with visualizations, the interactive events get detected by listeners. Based on the events, the application logic applies corresponding updates to the data and the visual rendering.

data visualizations (in figure 5.7), Schwab et al. consider visualization synchronization in terms of three layers: (1) visualization synchronization which synchronizes the visualization itself. Since web visualizations often directly deal with the document object model (DOM), this can be achieved by synchronizing the DOM elements; (2) application logic synchronization, synchronizing and executing the updated application logic would lead to the same updated data and DOM; and (3) interaction synchronization, which synchronizes the execution of low level DOM events. VisConnect applies interaction synchronization since it is considered the most suitable approach in terms of generalization and compatibility with data binding.

We explore computational transclusion in the context of transclusion visualization. Though our work does not specifically consider collaboration or cross-device visualization, we draw on the above research to identify visualization generation pipelines as well as the different approaches to synchronize multiple instances of a visualization. These studies tend to focus on replicating visual representations and synchronizing interactions. Our work explores transcluding "computational bundles" — both the visual representation and its generation pipeline or dependencies. We focus on linking and synchronizing these different visualization components in flexible ways.

5.3 Exploring Computational Transclusion

Instead of separate copies, the original and the reused content (or the alternatives) could be interconnected where changes on one side might have impacts on the other. As shown in 5.2, a number of techniques try to re-establish the missing links, however, they tend to impose a

certain way of linking and hide the links from end-users. It is thus difficult for analysts to access them and manipulate how their alternatives are linked to suit for their contextual needs.

We envision "computational transclusion" as a novel reuse approach that maintains various types of links between the original and the reused contents to facilitate tracking and coordinating changes during the iterative sensemaking process. We refer the original content as the *transcluded* and the reused or derived content as the *transcluding*.

"Computational transclusion" could be a broad concept that involves many different contexts. In this work, we consider specifically reusing visualizations from a computational notebook to other places, such as another notebook, or a presentation slide. To untangle the intertwined notions around transclusion, we ask the following questions:

- What are the components of computational transclusion of visualizations?
- What are the various kinds of links users might want between the transcluding and the transcluded?
- How to reify these links in the user interface to enable flexible manipulations of transclusion?

Preliminary study approach To study transcluding visualizations, we built a sandbox on top of Webstrates [14] — a web-based research platform that provides many prototype tools like whiteboards and computational notebooks and enables users to change the environment from within. These features of Webstrates make it possible for us to quickly test different ideas on transclusion.

Our sandbox system is a web-based collection of notebook and presentation interfaces that enable us to probe into a variety of visualization reuse scenarios. Using this sandbox as a tool, we aim to better

understand the various kinds of mental models users might hold when using transclusion, the notions and concepts involved, and how these models might break down with current technologies.

5.3.1 Components of visualization transclusion

Before exploring the possible relationships between the transcluding and the transcluded, we first need to understand the types of contents that might be reused when transcluding visualizations, i.e., the components of visualization transclusion.

In this section, we summarize various components that could influence the generation and the state of visualizations. We introduce "computational bundle" as a transclusion unit for visualizations — where all necessary dependencies to generate a visualization get transferred to the transcluding environment, and exposed to users so that they can modify them.

Visualization Components To generate a visualization, analysts often need go through a pipeline. Taking the bar chart in figure 5.8 as an example, it is the output of a combination of cells: (1) a code cell that loads the data set to be analyzed into the working environment, and potentially cleans or transforms the data into a desired form. (2) Another code cell that generates the bar chart, often with the aid of some visualization libraries or packages. Analysts write code to define their application logic, including how data items are mapped to visual elements, as well as the related interaction logic. (3) Finally, there is a CSS style cell that customizes the visual appearance of the bar chart, such as the size and color of the visualization.

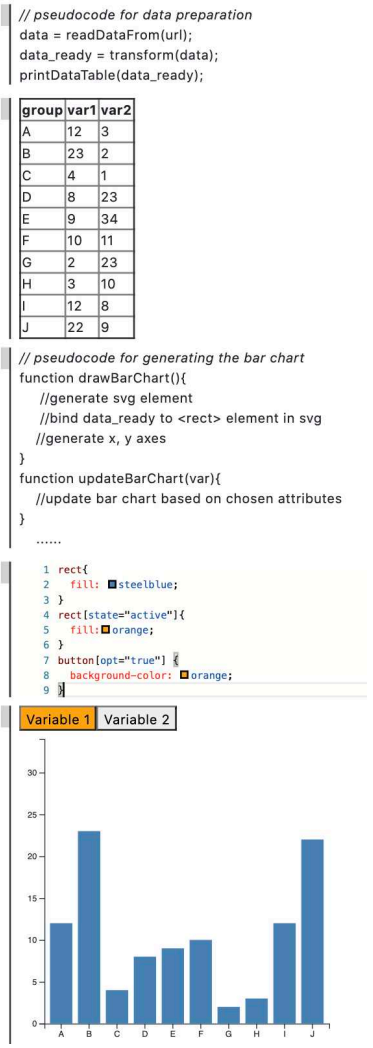


Figure 5.8. An example of a bar chart generated in our sandbox notebook. The generation of the bar chart relies on: a code cell to load and transform the data to be analyzed; another code cell to generate the visualization, including defining the application logic as well as the related interactions; and a style cell to customize the appearance the bar chart.

In lack of any of these cells, the bar chart cannot be completely replicated in another place. Any changes in these cells could influence the visualization result. We refer these cells engaged in generating a visualization as the *visualization dependencies* or *pipelines*.

Furthermore, the state of a data visualization could be influenced by the interactions performed on it, as well as the value changes of runtime variables. For example, interacting with the buttons in figure 5.8 changes the data attributes bound to the bar chart. Considering the case of running a simulation model like virus spreading. Once the simulation starts, it continuously updates the visualization to reflect the simulation result in real-time. Thus the visualization state also relies on the values of related runtime variables.

As shown in figure 5.9, we summerize the pipelines to create a visualization and the factors that could influence its state as a set of components, including: visualization packages, data sets, code, styles, UI widgets (if there is any), and runtime of the application. Though the concrete artifacts and steps engaged to generate visualizations rely on each single case, most visualizations involve some combinations of these kinds of components.

Computational Bundle We define "computational bundle" as a unit for transcluding visualizations — where all necessary dependencies for a given visualization in the computational notebook are transcluded to the new environment. Moreover, these dependency contents are also exposed to users so that they can freely modify them. As such, the transcluding visualization inherits the data, application logic, styles, and interactivity of the transcluded visualization. All components in both sides can also be further modified for alternative goals.

In our following explorations, we leave aside the technical challenges in identifying such visualization dependencies. We assume that

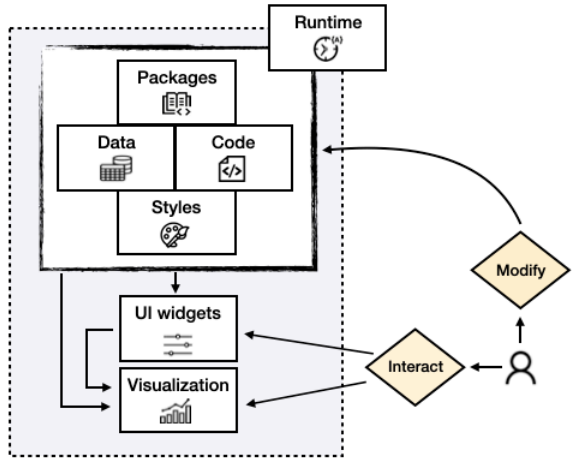


Figure 5.9. Components of transclusion of visualizations: A visualization is the visual representation of the underneath data. To generate a visualization, different kinds of dependencies are engaged; a visualization's state further relies on other factors like user interactions.

users can identify and transfer the necessary dependencies for a given visualization, either manually or automatically [11].

Depending on the context, users might prefer different kinds of reuse formats. Sometimes, they might just want to export and import the bar chart as a static image, like PNG or JPEG file. In the context of web visualization, the visual objects are stored as SVG elements in a website's DOM (Document Object Model — all elements in a web document are represented as nodes in a tree structure). For example, the bar chart is made up of <rect> elements inside a SVG element. To reuse the visualization, users can also copy these elements in another

[11] For example, the Code Gather tools [27] automatically identifies all dependent cells for a given visualization and duplicates them in a new notebook.

place. Yet we do not consider the static images or the collection of elements as computational artifacts. Since they lose the underling data, the application logic, and the interactivity of the visualization. Other reuse techniques, such as the "embed cell" in observable[12] automatically loads the dependencies for a visualization in the new environment. However, it hides the dependency contents from end-users, making it difficult to access and modify them.

Our Exploration Scope With the discussed concepts, we clarify the scope for our following explorations on computational transclusion: we explore transcluding dynamic web visualizations as "computational bundles". We consider scenarios where all components in both the transcluding and the transcluded environment could be modified alternatively based on analysts' contextual needs.

5.3.2 Exploring various transclusion scenarios

In this section, we pick up four concrete scenarios to demonstrate different kinds of visualization reuse cases and the challenges involved. We then discuss the design insights we learned from these explorations.

5.3.2.1 Transclusion scenarios

Scenarios 1, 2 and 3 demonstrate the reuse of an interactive bar chart, from one notebook to another notebook, as shown in figure 5.10. We use the convention that the interactive bar chart is initially generated

[12] `https://observablehq.com/@observablehq/introduction-to-embedding`, accessed on 21-04-2021.

in the light notebook and being transcluded to the notebook in dark theme. Scenario 4 demonstrates the reuse of a simulation model from a notebook to a presentation slide, as shown in figure 5.11.

We refer the initial contents as *the transcluded*, and the reused or the derivative as *the transcluding.*

Scenario 1 As shown in figure 5.10, after transcluding the interactive bar chart from the light notebook to the dark notebook, we first notice that the axes of the chart disappear since they are in similar color as the background of the transcluding notebook. We thus edit the style cell and change the axes' color to white. We intend to keep this change locally so that each visualization can suit its proper background.

We also find that the labels of these axes are too small, making it hard to read them. We thus continue to edit the style from the transcluding side, enlarging the labels' font by increasing its size. In this case, we want to synchronize this update to the transcluded notebook to ensure a proper size of axes in both visualizations.

Without any linking support, we must track back to the transcluded notebook and manually fix the changes. We thus start to implement a linking mechanism to facilitate this practice. While, we figure out that if we synchronize the entire transclusion content as a whole, the style for the transcluded visualization might risk being overwritten, meaning the color of its axes turns to white. We need a more fine-grained synchronization on the range that only affects the labels' font size, so that we can keep the black axes but have larger labels in the transcluded visualization.

Scenario 2 During the analysis, we need to update the bar chart to include additional data items or consider different data attributes. This kind of update often requires changing the corresponding code

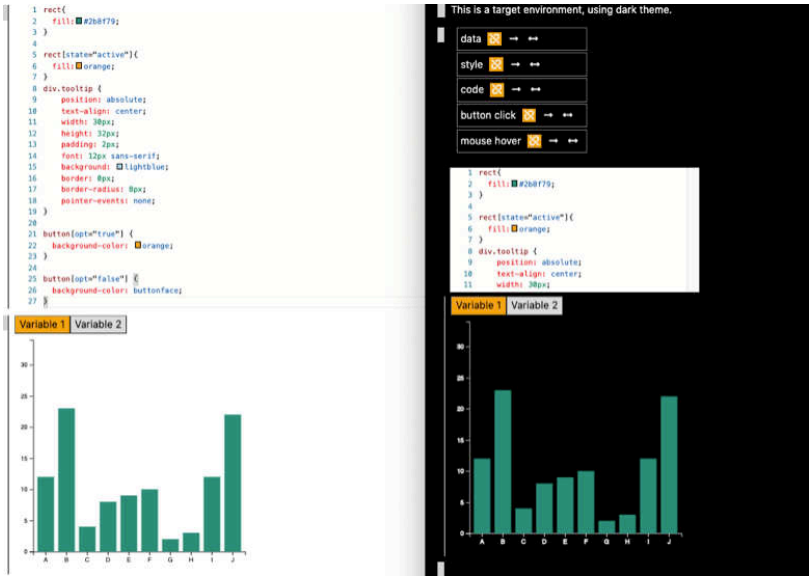


Figure 5.10. Transcluding an interactive barchart from one notebook (left) to another in a different theme (right).

pieces, including data loading, cleaning, or binding. To maintain analysis consistency, we expect to have the same updated visualizations on both the transcluding and transcluded notebooks.

There exist two approaches to synchronize the visualizations. As one option, we could synchronize the visual representation itself without updating the underling data and code logic. From the system's point of view, the change of the visualization is represented as the change of elements in the DOM. Assuming that we update the visualization on the transcluding side, to synchronize the update, the transcluding side could send the corresponding changes to the transcluded; or on the other hand, the transcluded side could actively fetch these changes and apply them to its local visualization.

However, this synchronization approach breaks the data binding on

the transcluded document. The data sets loaded in the transcluded notebook didn't get updated to review the actual data changes, neither does the code logic. The updated visualization thus disconnects from the data in the transcluded environment. Re-running the notebook will turn this visualization back to its previous state. The code might even break and fail to generate the visualization.

As another option, we could instead synchronize all corresponding pipelines. In this case, both the transcluding and transcluded notebooks will hold the same data and apply the same application logic. Yet it could be tricky since we won't observe a immediate change on the visualization until we re-run the code. Note that we consider most computational notebook environments that run code cells linearly from top to bottom. There are other notebooks like Observable which automatically tracks dependencies of cells to ensure instant feedback [13].

Scenario 3 To explore the data, analysts often need to interact with visualizations. For example, we can examine different data subsets or attributes by clicking the buttons in figure 5.10. There exist many kinds of interactions, some only affects the visual representations, like mouseovering on a bar highlights it; some might operate on the underlining data structures, like filtering the data to focus on a subset. Based on different situations, analysts may want to selectively synchronize or desynchronize these interactions between the transcluding and the transcluded side.

We explored different ways to synchronize interactions. As one option, we could synchronize the execution of low-level interaction events, such as *onclick*, *mousemove*, *mouseover*, etc. However, since the transcluding and transcluded content might evolve alternatively,

[13] `https://observablehq.com/@observablehq/reactive-dataflow`, accessed on 03/05/2021.

the underlining logic related to these events might not stay the same. For instance, the same resort button can be implemented differently, one based on time sequence, another on result values. In this case, synchronizing the clicking event will lead to different visualization updates.

As another option, we could directly synchronize the visualization itself, that is synchronizing the DOM elements, to imitate same interactive reactions. In this case, the two visualizations will stay the same. While similar to that discussed in scenario 2, this synchronization approach might lead to disconnection between the visualization and the underlining application logic. Without proper designs, both options could introduce confusions or conflicts with users' mental models.

Scenario 4 Considering the second example of reusing the simulation of virus spreading generated in a notebook to a presentation slide. As shown in figure 5.11, the population is represented as circles. The x and y attributes of a circle simulate a person's mobility, and the color

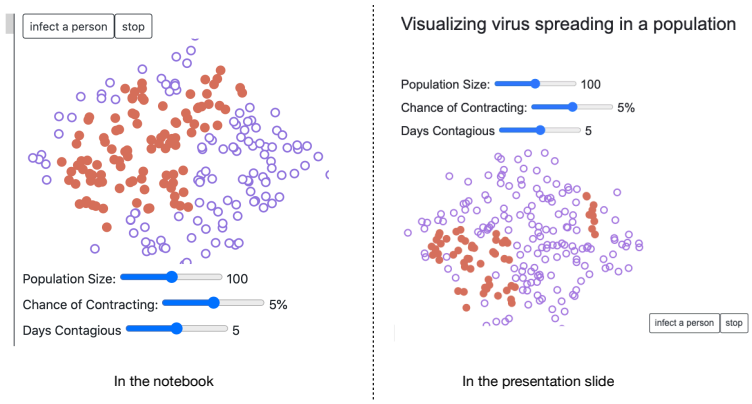


Figure 5.11. Transcluding a simulation model from a notebook (left) to a presentation slide (right).

attribute indicates whether a person is infected. Clicking the button "infect a person" starts the simulation. Users can further use the sliders to alter different parameters' values. During the process, the visualization constantly gets updated to reveal the simulation results.

We could apply different linking strategies for different kinds of analytic needs. For example, synchronizing the clicking button events but separating the changes on sliders' values allows analysts to compare different simulation cases simultaneously. We could also link two pairs of sliders and unlink one to compare the influence of one specific factor in the simulation.

While even with exactly same parameter configurations, these two visualizations could evolve differently. Since it is a non-deterministic model, the randomness in computing leads to two different results, thus two different visualizations. Sometimes, we might want to keep the two visualizations exactly the same.

To achieve this, we could synchronize the DOM elements. As previously discussed, this DOM synchronization could break the data-binding. For example, a circle in red represents that a corresponding person is infected, yet the underlying data didn't get updated to show this infection. Depending on the context, this disconnection might be ignored or need to be further solved.

We could write extra code to build a bi-directional binding of data and visual objects, so that the updates of visual elements will trigger corresponding data updates. As another option, we could execute the code on a single side and share the output and values of runtime variables to another.

5.3.2.2 Design insights

Though these four reuse scenarios are relatively rough and naive compared to real-life sensemaking cases, they still showcase the nuances

in how visualizations and related components could be reused asymmetrically during sensemaking processes.

Based on these explorations, we find that "computational transclusion" should offer various types of links to map diverse reuse models that users might require. First of all, the two alternative contents should be linked to enable users to go back and forth between them. Sensemaking is a messy and iterative process, this linking can save them the efforts to find and locate the alternative one in the transclusion pair.

Secondly, though the entire visualization bundle is transcluded as a whole, users might need more flexible and fine-grained control of the contents inside a bundle. Considering synchronizing changes, linking text-based contents like code pieces could happen in a cell level or in any arbitrary size. While linking the visual representations could be more complicated due to the multi-interpretations and implementations. It could be linked on the visual representation level, or on the data and application pipelines, or even on the runtime environments.

Thirdly, instead of keeping a permanently live synchronization, users might want to selectively synchronize or desynchronize the contents as they move between different sensemaking tasks. They might want to ignore the changes from the other side, or be aware of them to make proper reactions, or let the system automatically synchronize the updates. No matter the case, they should be able to figure out the current model that is applied and be able to switch to other models when needed.

5.4 Transclusion Properties & Reifications

In this section, we first introduce six identified properties of transclusion to help untangle the intertwined notions involved in transcluding visualizations. We then explore how to reify these notions in the user interface to facilitate flexible (re-)configurations of transclusions.

5.4.1 Transclusion Properties

To clarify the different roles of the transcluding and the transcluded contents as well as their relationships, we propose six properties of transclusion:

- *linking*
- *role of artifact*
- *application of changes*
- *direction of propagation*
- *linking granularity*
- *linking depth*

Linking The fundamental idea of transclusion is that the alternative contents in multiple places stay connected. This connection can be first reflected in whether it enables users to navigate back and forth between the transcluded and the transcluding contents. We call this connection as the linking property.

Linking can further have directions, we refer to moving from transcluding to transcluded as trace-back and from transcluded to transcluding as trace-forward. Many reusing techniques discussed in related work section enable trace-back but not trace-forward.

Role of Artifact Unlike Ted Nelson's transclusion which establishes a permanently live connection among *same* content in multiple places, we consider asymmetric reusing cases — where both the transcluding and the transcluded content could be modified alternatively.

Considering informing content updates, the transcluding and the transcluded can both be a *sender* that sends their local changes to the other side and a *receiver* that gets remote changes from the other side. In case the transcluding is not allowed to be further modified, it can only act as a receiver.

Application of Changes We consider three different models — *no application*, *manual application* and *auto application* — for applying changes from the view of both sender and receiver.

For a sender, the application models relate to how changes on the sender's side are sent to the receiver. In no application mode, changes on the sender's side will not be propagated. In manual application mode, users choose when and what to propagate. In auto application mode, changes are propagated automatically in real time.

For a receiver, the application models relate to how the received changes are applied locally. In no application mode, the received changes will be ignored. In manual application mode, users receive the changes immediately as the sender sends it, but the receiver can reacts to them in a later time. The receiver can either accept the changes to apply them locally or reject them to stay on its own version. In auto application mode, received changes will always be automatically applied.

Direction of Propagation In the above cases, a receiver acts passively — it first waits for the sender to send the changes and then perform proper reactions based on the different models applied. We call this model the "sender-push". There is still the "receiver-pull" model. Instead of passive and receptive, the receiver side deliberately requests the changes from the sender.

The "sender-push" model and "receiver-pull" model are not in conflicts. As a receiver, the setting for applications of changes define its reactions towards received changes, in the meanwhile, it can also actively request changes. For example, to concentrate on modifying the transcluded content, a user sets the "receive" as no application to ignore all changes sent from the transcluding. Afterwards, the transcluded can request and pull the changes from transcluding to examine its updates.

Linking Granularity When we transclude a visualization as a computational bundle, it contains many components. The linking and synchronization can be achieved in different granularity.

First, we can consider the bundle as a whole piece, where any changes inside the bundle could trigger the synchronization of the entire bundle. However, as shown in our scenarios, users often need more fine-grained controls. We can consider linking in the level of cells. Since cells are designed as the basic constructing units for computational notebooks, it seems natural to treat them as the linking units as well.

As discussed in scenario 1, even within a style cell, users might only want to synchronize a specific range like the font size. They can achieve this by splitting the original style cell into multiple smaller ones. On the other hand, we could design more flexible and fine-grained linking units, such as letting users define arbitrary ranges.

Linking Depth As discussed in our scenarios, there exist multiple interpretations in synchronizing visualizations. We use "linking depth" to describe how or to what degree the transcluding and the transcluded visualizations are linked. We identify three synchronization levels — *interaction level*, *document level*, and *runtime level*.

In interaction level, the corresponding interaction events get synchronized. For example, clicking buttons on one side will trigger same button click events on the other side. We further consider interactions as two types, those only affects the visual representations and those operate on the data structures that guild the visualizations. For the latter case, users might need to further synchronize changes on the underlying data and application logic.

In document level, it is the changes in the DOM that get synchronized. For example, the adding, deleting of visual elements or the changing of elements' attributes. We need to pay attention that this DOM synchronization often breaks the data-binding on the side to be synchronized. Extra operations are needed from users to solve the breaks. As tool designers, we should at least think mechanisms to indicate users about the potential disconnections.

In runtime level, the code is executed on a single side and the output and the values of runtime variables are shared to another in real time. Thus it updates the underneath data sets as well as its visual representations.

Conclusion We extract these six properties of transclusion — *linking, role of artifact, application of changes, direction of propagation, linking granularity*, and *linking depth* — from our own explorations on asymmetric visualization reuse from computational notebooks to other places. They constitute the basic space for designing transclusions. We believe by applying different combinations and settings of these properties, we can fulfill various kinds of transclusion cases based

on users' situated needs during their sensemaking processes.

Some of these properties might be more oriented for certain components inside the visualization bundle. For example, the *application of changes* property could be more suited for textual-based components such as code pieces or style sheets. Since the text contents might need to be updated both synchronously or asynchronously. But for interactions, it might make more sense to synchronize the actions in real time. The *linking granularity* property is also oriented for textual-based components. While it could be applied to visual representation itself, such as when only part of the visual objects need to be linked. Whereas the *linking depth* property focuses on the visual representation.

There might also exist many unexplored notions and challenges around transclusion. Note that this is our preliminary explorations and findings on computational transclusion, instead of providing clear-cut solutions, we hope our work can be a starting point for explorations and discussions to address the issues in managing alternatives in the context of reuse. We discuss the limitations of our work and the future directions in section 5.5 & 5.6.

5.4.2 Reify transclusion

With the identified concepts, we then explore how to reify them in the user interface to enable analysts to flexibly (re-)configure transclusions based on their situated needs.

Figure 5.12 demonstrates the process to transclude a bar chart from one notebook to another in our reification prototype. When a visualization gets selected, the related dependency components will be highlighted. In case of "quick transclusion", all these dependencies get transcluded to the new environment. Users can also manually assign a combination of them to transclude. The dependency components can

Figure 5.12. Transcluding an interactive bar chart from notebook A to notebook B. Clicking on the bar chart highlights all dependency cells. The user can either select "quick transclude" to move the entire visualization bundle or manually choose a combination of cells to transclude.

be folded as a block or be hided when not in use [14].

Once finish transcluding the content to the desired place, users can further configure how they want to synchronize the changes made in both the transcluded and the transcluding content. Based on the *linking granularity* concept, we apply a cell level linking in our prototype.

[14] As a simplified prototype, we manually predefined relevant cells as different types of dependency components. For example the data cell has an attribute named "data + (visualizationId)" to indicate this data is used to generate a particular visualization. The identification of dependencies and their types could be implemented in a more intelligent way in the future implementation.

Users can thus apply different linking models for each cell. They can decide how they want the changes in a cell to be propagated to the other side, and how the received changes to be applied locally.

Since each cell can have its configurations, it can quickly become confusing to figure out the exact settings for each cell. We need to reify the *application of changes* property to not only facilitate configuring users' mental models but also help them to maintain awareness of what models have been applied.

Figure 5.13 demonstrates a possible reification that we explored. In our prototype, each cell has a set of configuration widgets: 1) First, a linking icon to help users perform back tracing or forward tracing between the transcluding and the transcluded content. Clicking the icon can thus open the corresponding document and jump to the proper location. 2) The two arrow buttons represent a cell's role as both a sender and a receiver. By default, the left arrow for sending and the right arrow for receiving. Users can manipulate them to configure the

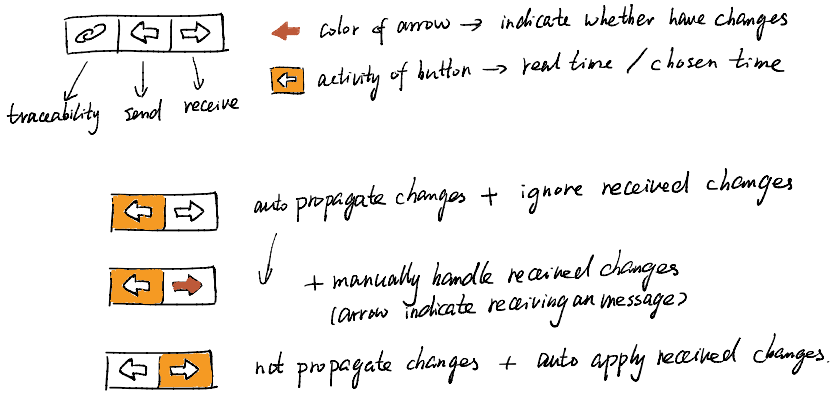


Figure 5.13. A possible reification on application of changes property of transclusion.

models of changes application.

Long clicking the left arrow button activates the auto model for sending local changes to the other side. The button's background turns to orange to indicate this activation. Long clicking the activated button will deactivate it and change the sender to no application model. Similarly, activating the right arrow button allows received changes to be automatically applied in the cell. Deactivating it ignores all received changes. Users can also choose to manually send a change or apply a change by a simple click on the corresponding arrows. The color of the right arrow indicates whether there are upcoming changes from the other side; the color of the left arrow indicates whether there are local changes that need to be propagated to the other side.

In terms of *direction of propagation* property, this reification considers the "sender-push" model. It provides no direct support for "receiver-pull". Yet users can achieve an active request by tracing to the other side to verify its updates (using the linking icon).

In terms of *linking depth* property, if the user manipulates the icons for the visualization cell, the synchronization happens in document level where the changes of the DOM elements get synchronized. We also extract all supported interactions as different components that can be manipulated, such as button click and mouse hover are treated as two separate components in figure 5.10. Users can configure the links separately for each type of interactions.

In this reification example, users can only see the local configurations but not those on the other side. Since this work considers the single-user case, where the same person updates the transcluding and transcluded content and manages them on both sides, we believe this design might be feasible for them to understand and coordinate the configurations on two sides.

5.5 Discussion

In this chapter, we present "computational transclusion" as a novel reuse approach that maintains the links between the original and the reused contents to facilitate tracking and coordinating changes. We demonstrate various reuse scenarios and discuss the mental models that users might hold and the challenges involved. Based on our explorations, we propose six pertinent properties of translcusion to clarify the kinds of links between the transcluding and the transcluded content. Finally, we explore how to reify the identified concepts on the user interface to provide users flexible and fine-grained control over transclusion.

In this section, we discuss how the proposed "computational transclusion" concept is different from existing reusing techniques. This is still a preliminary work based on our own observations and intuitions, we discuss both its conceptual and method limitations.

5.5.1 Computational transclusion vs. other reuse techniques

"Computational transclusion" fill in the gaps between users' needs of maintaining alternatives in the context of reuse and the missing links in existing techniques. To clarify how "computational transclusion" is different from other reuse techniques, we compare it with three categories of techniques discussed in section 5.2: the original concept of transclusion by Ted Nelson; the "linking & embedding" techniques; and the synchronization in distributed visualization.

Ted Nelson's transclusion Ted Nelson's transclusion focuses on the *same* content in multiple places. He considers these contents as a linked network where users can trace between multiple instances and changes in one place can be automatically synchronized to the others in real-time. Whereas, the transclusion that we propose handles pairs of alternatives in *asymmetric* reuse — where the transcluding and the transcluded content can be modified alternatively.

Unlike Ted Nelson's transclusion which describes the network of information in multiple places, our transclusion only considers two places, the computational notebook where the content is generated; and the transcluded environment where it is reused.

During sensemaking processes, analysts iteratively go back and forth to update, compare, and manage the transcluding and the transcluded content for different kinds of analytic tasks. Instead of keeping a permanently live synchronization among multiple instances, computational transclusion enables pairs of alternatives to be linked or synchronized in a fine-grained manner that can be flexibly manipulated based on users situated needs.

Computational transclusion also differs from Ted Nelson's concept in terms of the type of content. Ted Nelson basically considers texts or images. While computational notebooks integrate narrative texts, data sets, code pieces, as well as interactive widgets and visualization results. These contents often hold complicated dependencies, changes in one place could influence the others.

We explore transcluding data visualizations, which are the outputs of a combination of other contents and can be interacted with during runtime. These pipelines and interactions introduce both conceptual and technical challenges while transcluding. Our study primarily focuses on conceptual challenges. We break down the different components of visualization transclusion and explore various types of links between these components.

Linking & embedding "Linking & embedding" techniques also focus on the reuse of content from one place to another while still preserving their links. Unlike "computational transclusion" which lets users define the types of links, existing "linking & embedding" techniques tend to impose certain types of linking mechanisms. Users are often not authorized to modify them.

In Microsoft Office applications, the transcluding contents can not be modified. Users can only update them in the transcluded environment. When the transcluding document is open, the changes in the transcluded will be automatically synchronized to the transcluding in real-time; in case the transcluding document is closed, an update message will show up the next time users open it.

Using the proposed properties of transclusion, we describe Microsoft Office applications only allow the transcluding as a receiver. Both transcluding and transcluded apply auto propagation mode. In other words, it is a single direction, real-time synchronization from the transcluded to the transcluding.

In Google applications, changes in the transcluded side are automatically propagated to the transcluding place. The transcluding side receives reminders of updates and leaves users to decide how to handle the changes. Using the proposed properties of transclusion, we describe the transcluded side as a sender which applies auto propagation mode, and the transcluding side as a receiver which applies manual propagation mode.

Unlike Google and Microsoft which encapsulate the code to generate visualizations, computational transclusion considers both the transclusing and the transcluded visualization content can be modified. Computational transclusion allows users to flexibly manipulate how the pair of transcluding and transcluded is linked based on multiple granularities.

Distributed visualization As discussed in the related work (section 5.2), research on distributed visualization also studies the transfer and synchronization between multiple instances of a visualization.

One group of work is on collaborative visual exploration. It focuses on synchronizing the explorative interactions performed by multiple users on distributed devices [159,161–163]. Synchronizing interactions among multiple visualization instances is also one of the problems that we tackle in transclusion. Yet we explore it from a single user perspective in the asymmetric reuse context. As a result, we do not handle issues such as concurrency control or group awareness.

Similar to Schwab et al.'s VisConnect [162], we consider synchronizing a visualization in terms of three levels: visual representation synchronization, application logic synchronization, and interaction synchronization. VisConnect takes the approach of synchronizing interaction events due to the considerations on compatibility with data binding and easiness to develop with. Our work, instead, reifying these different layers of synchronization and let users define the way they want based on their contextual needs.

Another group of work on distributed visualization focuses on leveraging various displays for visual sensemaking. They study how a visualization can be easily transferred from one device to another [160], or how to combine multiple devices to support various analytic needs, such as to extend display space, filter data items, or manipulate overview & detail views [164]. They focus on the usage side of visaulizations, but not the generation. In computational transclusion, we focus both on the interactions with visualizations as well as the visualization generation pipeline. We identify various components of visualization and allow users to modify them in both the transcluding and the transcluded place (could be on the same device, or different devices).

5.5.2 Limitations

This chapter presents our preliminary work on computational transclusion. In this section, we discuss both the conceptual and method limitations of our work.

Conceptual limitation Though computational transclusion could be applied across different users, in our study, we explore transclusion from a single user perspective as a first step. Therefore, the six proposed properties of transclusion only consider the involvement of a single user.

Introducing collaboration in transclusion could potentially bring many extra problems. For example, in terms of the property of *application of changes*, a single user can only work on one transclusion side at a time, thus we do not need to handle concurrency issues. In the proposed concepts, we assume actions such as tracing or requesting for changes are always approved. Whereas if multiple users are engaged, these actions might need extra authorization for coordination or security concerns. For instance, when a reader explores the visualization results on a blog, he might ask the original author whether he can access to the original notebook for details.

In our reifications, we do not provide awareness mechanisms for transclusion configurations on the other side. We assume that a single user could maintain a consistent mental model on both transclusion sides. It takes relatively less efforts for them to check the configurations for the other side. Yet for multiple users, they might need more explicit indications.

Moreover, our current reification of transclusion is constrained by the cell structure as implemented in most current computational notebooks. However, the computational media could also be more flexible and reconstructible. For example, instead of following a linear cell

representation, Boxer [165] represents computational objects as boxes that can be freely moved in space and be composed within other boxes. These differences in how computational objects are represented and structured could lead to different reification of transclusion.

Note that transclusion can become extremely complicated when considering the entire network of reused artifacts or a chain of transclusion — transcluding a content that is originally transcluded from another place. In our work, we only explore a pair relationship, where a visualization get transcluded from a computational notebook to another place, and study the interplay between them.

Method limitation In our current work, we explore various notions and issues around transclusion by simulating reuse cases based on our own experience and observations. These explorations all happened in the sandbox system that we built on top of the Webstrates [14].

As a preliminary stage, the sandbox explorations provide us various insights on how visualization and its dependencies might be reused. Various issues are exposed when we try to coordinate the changes made in both transclusion sides. Yet, these discoveries need to be strengthened through the involvement of end-users — the practitioners who use computational notebooks for visual sensemaking. For example, we could apply various qualitative methods to collect their painpoints on visualization reuse. We will discuss the future steps in the next section.

In our sandbox prototype, we leave aside some technical barriers which could be challenging to solve: Firstly, how to identify all dependent components for a given visualization result. Secondly, when loading the "computational bundle" to a new environment, either automatically or manually, how to make sure the transcluded contents are compatible with the original content in the new environment. After being transcluded, whether the system immediately evaluates (runs)

the transcluded code functions or lets users manually run each cell. Last but not least, how to identify and inform the potential inconsistency between data, code logic, and visual representations.

These technical issues are all pertinent for transcluding. In our work, we focus on the conceptual notions around transclusion rather than the actual implementations.

5.6 Future Work

In the future, we can strengthen our contributions through the following aspects.

First, we could conduct various studies to collect and verify the painpoints on visualization reuse from computational notebooks to other places. For example, we could conduct interviews with analysts to gather their requirements on reusing visualizations, the problems they encounter, and their coping strategies. We could also design questionnaires to get both quantitative and qualitative feedback on related painpoints. We can check whether they align well with those identified in our explorations and intuitions.

Second, to better reify various concepts around transclusion, we could further conduct participatory design workshops with analysts or computational notebook practitioners. This approach helps us to elicit insights from participants. In the workshops, we can start by presenting several concrete reuse cases to demonstrate the concept of transclusion. Through these cases, we could either explicitly or implicitly explain the identified properties of transclusion. Then we can ask participants to brainstorm and generate diverse designs to reify transclusion.

Finally, to validate the proposed concepts and reification designs, we could further implement the reification as extensions to current

notebook systems and observe how analysts adapt them to their real analytic tasks.

CHAPTER 6

Conclusion

This dissertation explores sensemaking through the lens of "alternatives" and addresses both better understanding and supporting alternatives in data workers' sensemaking processes.

To make sense of data, data workers often explore different kinds of alternatives: they might examine multiple data sources, explore diverse sets of hypotheses, try out different types of methods, experiment with a broad space of possible solutions, and combine a variety of tools. These alternatives influence each other within a dynamic and complex sensemaking process.

However, current sensemaking tools poorly support the exploration and management of alternatives in sensemaking. First, they tend to serve a specific purpose with some predefined functionalities and workflows. It is hard for users to appropriate them to explore or combine alternatives as they arise in the sensemaking process. Second, they tend to treat the explored alternatives as disconnected pieces. Analysts thus waste a significant amount of time and mental resources in trying to reconnect them and coordinate them as they go back and forth between different analytic contexts and stages.

In the preceding chapters, we presented the results of the qualitative studies conducted to better understand the role alternatives in sensemaking; and the explorations on designing more flexible tools to support them. This chapter revisits the contributions of this dissertation and highlights directions for future research.

6.1 Contributions

In this dissertation, we focus on sensemaking as the intersection of human cognitive processes and the tools that enable them. The key research questions, as introduced in Chapter 1, are: (1) *How do alternatives fit within the sensemaking process?* (2) *how can tools better support the exploration and management of alternatives?*

Chapter 3 addresses RQ1 by exploring the kinds of alternatives that arise in sensemaking and how data workers cope with them. Chapter 4 & 5 address RQ2 by exploring how to design tools to support alternatives in two specific sensemaking contexts. Chapter 4 explores how tools can link related alternatives and be appropriated to suit different contextual needs in qualitative sensemaking processes. Chapter 5 probes different types of linking mechanisms within the context of reuse to facilitate tracking and coordinating changes in alternatives.

We conclude that digital tools can facilitate sensemaking by (1) applying various linking mechanisms to help track and manage associated alternatives; and (2) being re-configurable in terms of devices and features to let the user adapt these tools to their contextual needs during the overall sensemaking process.

We consider this dissertation brings the following primary contributions:

Characterization of alternatives Before our investigation, the HCI in visual analytics literature lacked a deep, systematic understanding of the kinds of alternatives data workers consider and how they fit into their general sensemaking processes. Most research work studies certain types of alternatives in a specific sensemaking context; or characterize sensemaking from from different angles, They do not consider the ensemble types of alternatives and how they fit within the overall sensemaking process.

Our first key contribution is a rich characterization of the role of alternatives as they fit within the sensemaking process. In chapter 3, we presented our semi-structured interviews with 12 data workers, we show the kinds of alternatives they consider; why they consider them; the triggers and barriers to dealing with alternatives; and the strategies applied by our participants.

Drawing upon our analyses and findings, we characterized alternatives using a theoretical framework based on participants' 1) degree of attention, 2) abstraction level, and 3) analytic processes. This framework can help describe and reason the kinds of alternatives considered in sensemaking, the related processes, and how tool designers might create more flexible tools to better support them.

Design Space for Qualitative Analysis Tools In chapter 4, we present a design space for qualitative analysis tools, drawn from the literature, our own experience, and that of our informal interview participants. In qualitative data analysis (with affinity diagramming technique), there are five main dimensions to be considered: (A) analysis phases, (B) conceptual methods, (C) analytic lens, (D) modes of collaboration, and (E) types of devices used.

Various kinds of alternatives exist along these identified dimensions, including two alternative conceptual methods — coding and diagramming, multiple alternative analytic lenses, as well as diverse devices engaged. We find that: first, these alternatives are frequently disconnected in existing tools, leaving users to apply ad-hoc strategies to manage the mass of intertwined information; second, current systems are often designed to suit some specific analytic stages, methods, displays, or collaboration modes, making it hard for users to appropriate them for different contextual needs.

These dimensions enrich our understanding of qualitative sensemaking processes, the kinds of alternatives involved, as well as where

current tools break down.

Vision & Proof-of-Concept Prototype for QDA Tools In chapter 4, we presented a vision and proof-of-concept prototype, ADQDA, for qualitative analysis tools based on the identified design dimensions and pain points on existing tools.

To better support the exploration of alternatives in sensemaking, ADQDA embeds different types of links: 1) a synchronization-link that enables a flexible mix of alternative analysis methods; and 2) a cross-linking mechanism to facilitate search and compare alternatives across multiple views and displays. ADQDA further enables users to appropriate multiple pertinent displays to customize and extend their analysis environment based on their analytic tasks at hand.

Using the ADQDA prototype, we explored how analysts can appropriate available digital devices as they fluidly migrate between analytic phases or adopt different methods and representations, all while preserving consistent analysis artifacts.

We are not aware of other work that has articulated the need for such qualitative data analysis tools or demonstrated how new-ish technologies can be combined in a way to realize that need.

Series of Demonstrative Scenarios In chapter 4, we presented a series of application scenarios that show how ADQDA realizes our vision and can potentially facilitate richer, deeper qualitative data analyses. We validated ADQDA by comparing it with other related systems using these scenarios as a lens and drawing upon Olsen's criteria [135] — focusing on whether the problem is previously solved and interaction viscosity.

Greenberg and Buxton [166] argue that usability testing is not always appropriate for evaluating complex systems, and can be even harmful for novel systems or interaction techniques. Salovaara et

al. [167] point out the present-future gap in HCI's evaluation methodology — the way we design our evaluative studies has not taken the future-orientedness of prototypes into account.

We thus presented "design rationale, a vision of what could be, expected scenarios of use, reflections, [and] case studies" as more appropriate forms of validation [166]. This approach can serve as inspiration or open up issues for reflection and future research that explore how to evaluate complex systems for future context.

Vision & Prototype for Reuse Techniques In chapter 5, we envisioned "computational transclusion" as a novel reuse technique. "Computational transclusion" provides more explicit links between the original and the reused alternatives to facilitate tracking and coordinating changes. We implemented a sandbox prototype to help us explore the "computational transclusion" concept in the context of transcluding data visualizations.

Instead of offering clear-cut solutions, our work opens up issues for reflection and further research. The prototype is intended as a concrete reification of this concept to permit further explorations. We hope that our vision and prototype can be a starting point for more comprehensive and thorough explorations within the context of alternatives in reuse.

6.2 Future Work

In section 3.7, 4.8 and 5.6, we have discussed separately the future work for each of our research projects. In this section, we propose future research directions based on the insights and implications that we gained throughout our work.

Exploring Different Types of Linking for Alternatives As previously discussed, we find that the disconnection between alternatives in sensemaking is one of the major pain points facing by data workers.

Chapter 3 characterized different kinds and levels of alternatives, and discussed the intertwined relationships they might hold on each other. Chapter 4 explored how sensemaking tools can implicitly link related alternatives to preserve consistent analysis artifacts while analysts fluidly migrating between multiple analytic phases or adopt different methods and representations. Chapter 5 explored how different kinds of explicit links can be applied to facilitate coordinating changes in alternatives in the context of reuse.

Besides the two specific contexts, we encourage future research to study alternatives under different concrete analysis contexts and explore how various linking concepts and mechanisms can help to manage alternatives. An auto, bi-directional synchronization link as implemented in ADQDA is effective to handle the alternative artifacts which are conceptually identical. Yet, this bi-directional synchronization is not always suitable in asymmetric reuse cases in "computational transclusion". Chapter 5 provides a starting point to discuss various kinds of links in the context of asymmetric reuse. Future research can identify more comprehensible design space for tool designers by studying different linking concepts and mechanisms.

Furthermore, as discussed in chapter 3, alternatives at different abstraction levels can also influence each other. We encourage future research to explore these cross-abstraction-level links to facilitate sensemaking. For example, wang et al. find that the discussions between collaborators often contain valuable insight of alternative paths explored in computational notebooks [100]. However, these discussions are disconnected from the notebooks. They propose Callisto, an extension to computational notebooks, that maintains contextual links between discussion messages and notebook elements [100].

Considering Alternatives in Collaborative Sensemaking In chapter 3, we characterized alternatives by observing and studying individual sensemaking process. However, data analysis is often not performed alone. We encourage future research to characterize alternatives in a collaborative context.

Though our ADQDA prototype in chapter 4 can support various collaboration modes, our primary focus is on the other four dimensions, including multiple analytic phases, conceptual methods, analytic lenses, and types of devices. ADQDA thus only provides rudimentary support for collaboration. The current linking mechanisms might break down or hinder the analysis when multiple users are engaged. For example, the 'cross-linking" technique that works well to highlight a given note across multiple views for a single user across multiple devices might distract or confuse other users. Future research is required to examine different sensemaking scenarios and propose interaction techniques and awareness mechanisms are better suited to each of these scenarios.

Similarly, in chapter 5, the identified properties of transclusion as well as their reification consider single user cases. Future research is required to explore "computational tranclusion" in the collaborative sensemaking context.

Exploring Evaluation Methods for HCI Prototypes During our work, one of the biggest challenges is to understand how to properly evaluate our research prototypes. Real-world sensemaking is often complex and messy. It can be conducted throughout a long period of time and across multiple activities and contexts. Both ADQDA and "computational transclusion" are envisioned to facilitate such sensemaking process.

The value of ADQDA should be considered holistically: it is insufficient to consider the novelty of individual problems or individual

technological solutions. Rather, we combined multiple solutions that might already exist in a new and interesting way to support qualitative sensemaking and the alternatives engaged. Traditional measurements like task completion time or number of insights gained within a certain amount of time are not our primary concerns.

Ideally, we could conduct field trails where we introduce our prototype to participants' day-to-day life and collect data on its appropriation [168, 169]. Yet, our vision for a device rich analysis environment, including multiple interactive displays from the size of mobile phone to the size of wall, is still not pervasive for participants in the current world. Moreover, considering the context of the COVID pandemic, it is also not feasible to bring participants into our own facilities.

As Salovaara et al. point out, "while there are many methods for envisioning technologies, and sketching and prototyping them, the way we design our evaluative studies has not taken their future-orientedness into account" [167]. We hope future research can enrich the methods to evaluate complex systems and better solve this present–future gap.

Bibliography

[1] P. Pirolli and S. Card, "The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis," in *Proceedings of international conference on intelligence analysis*, vol. 5, pp. 2–4, McLean, VA, USA, 2005.

[2] D. M. Russell, P. Pirolli, G. Furnas, S. K. Card, and M. Stefik, "Sensemaking workshop chi 2009," in *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '09, (New York, NY, USA), pp. 4751–4754, ACM, 2009.

[3] B. Hartmann, L. Yu, A. Allison, Y. Yang, and S. R. Klemmer, "Design as exploration: creating interface alternatives through parallel authoring and runtime tuning," in *Proceedings of the 21st annual ACM symposium on User interface software and technology*, pp. 91–100, 2008.

[4] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer, "Enterprise data analysis and visualization: An interview study," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, pp. 2917–2926, Dec. 2012.

[5] J. Liu, N. Boukhelifa, and J. R. Eagan, "Understanding the role of alternatives in data analysis practices," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 1, pp. 66–76, 2019.

[6] J. W. Tukey and M. B. Wilk, "Data analysis and statistics: An expository overview," in *Proceedings of the November 7-10, 1966, Fall Joint Computer Conference*, AFIPS '66 (Fall), (New York, NY, USA), pp. 695–709, ACM, 1966.

[7] S. van den Elzen and J. J. van Wijk, "Small multiples, large singles: A new approach for visual data exploration," in *Computer Graphics Forum*, vol. 32, pp. 191–200, Wiley Online Library, 2013.

[8] M. B. Kery, A. Horvath, and B. Myers, "Variolite: Supporting exploratory programming by data scientists," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, (New York, NY, USA), pp. 1265–1276, Association for Computing Machinery, 2017.

[9] M. B. Kery and B. A. Myers, "Interactions for untangling messy history in a computational notebook," in *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pp. 147–155, IEEE, 2018.

[10] W. Mackay, "Users and customizable software : a co-adaptive phenomenon," 1990.

[11] N. Boukhelifa, A. Bezerianos, I. C. Trelea, N. M. Perrot, and E. Lutton, "An exploratory study on visual exploration of model simulations by multiple types of experts," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, (New York, NY, USA), pp. 1–14, Association for Computing Machinery, 2019.

[12] M. Beth Kery and B. A. Myers, "Exploring exploratory programming," in *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pp. 25–29, 2017.

[13] D. S. McLellan, "Presidential decisionmaking in foreign policy: The effective use of information and advice. by alexander l. george. (boulder, colo.: Westview press, 1980. pp. xviii 267. 24.00, *cloth*;10.00, paper.)," *American Political Science Review*, vol. 74, no. 4, pp. 1082–1083, 1980.

[14] C. N. Klokmose, J. R. Eagan, S. Baader, W. Mackay, and M. Beaudouin-Lafon, "Webstrates: Shareable dynamic media," in *Proceedings of the 28th Annual ACM Symposium on User Interface Software Technology*, UIST '15, (New York, NY, USA), pp. 280–290, Association for Computing Machinery, 2015.

[15] S. Bødker and C. N. Klokmose, "The human–artifact model: An activity theoretical approach to artifact ecologies," *Human–Computer Interaction*, vol. 26, no. 4, pp. 315–371, 2011.

[16] S. Bødker and C. N. Klokmose, "Dynamics in artifact ecologies," in *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design*, pp. 448–457, ACM, 2012.

[17] J. O. Wobbrock and J. A. Kientz, "Research contributions in human-computer interaction," *Interactions*, vol. 23, pp. 38–44, Apr. 2016.

[18] D. M. Russell, M. J. Stefik, P. Pirolli, and S. K. Card, "The cost structure of sensemaking," in *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, (New York, NY, USA), pp. 269–276, ACM, 1993.

[19] G. Klein, B. Moon, and R. R. Hoffman, "Making sense of sensemaking 1: Alternative perspectives," *IEEE Intelligent Systems*, vol. 21, pp. 70–73, July 2006.

[20] G. Klein, B. Moon, and R. R. Hoffman, "Making sense of sensemaking 2: A macrocognitive model," *IEEE Intelligent Systems*, vol. 21, pp. 88–92, Sept. 2006.

[21] B. Dervin, L. Foreman-Wernet, and E. Lauterbach, *Sensemaking methodology reader: Selected writings of Brenda Dervin*. Hampton Pr, 2003.

[22] K. E. Weick, *Sensemaking in organizations*, vol. 3. Sage, 1995.

[23] P. Zhang and D. Soergel, "Towards a comprehensive model of the cognitive process and mechanisms of individual sensemaking," *J. Assoc. Inf. Sci. Technol.*, vol. 65, pp. 1733–1756, Sept. 2014.

[24] G. Grolemund and H. Wickham, "A cognitive interpretation of data analysis," *International Journal of Statistics*, vol. 82, no. 2, pp. 184–204, 2012.

[25] "Merriam-webster's dictionary," 2019. https://www.merriam-webster.com/dictionary/alternative.

[26] A. Rule, A. Tabard, and J. D. Hollan, "Exploration and explanation in computational notebooks," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, (New York, NY, USA), pp. 1–12, Association for Computing Machinery, 2018.

[27] A. Head, F. Hohman, T. Barik, S. M. Drucker, and R. DeLine, "Managing messes in computational notebooks," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–12, 2019.

[28] H. Erhan, I. Y. Wang, and N. Shireen, "Harnessing design space: a similarity-based exploration method for generative design," *In-*

ternational Journal of Architectural Computing, vol. 13, no. 2, pp. 217–236, 2015.

[29] R. Woodbury *et al.*, "Elements of parametric design," 2010.

[30] M. Tohidi, W. Buxton, R. Baecker, and A. Sellen, "Getting the right design and the design right," in *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pp. 1243–1252, 2006.

[31] M. Terry and E. D. Mynatt, "Side views: persistent, on-demand previews for open-ended tasks," in *Proceedings of the 15th annual ACM symposium on User interface software and technology*, pp. 71–80, 2002.

[32] L. Zaman, W. Stuerzlinger, C. Neugebauer, R. Woodbury, M. Elkhaldi, N. Shireen, and M. Terry, "Gem-ni: A system for creating and managing alternatives in generative design," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 1201–1210, 2015.

[33] J. R. Guenther, *Shiro-A language to represent alternatives*. PhD thesis, Communication, Art & Technology: School of Interactive Arts and Technology, 2016.

[34] Y. V. Chen, *Alternatives in visual analytics and computational design*. PhD thesis, Communication, Art & Technology: School of Interactive Arts and Technology, 2011.

[35] N. Shireen, "Paraxplore: Demystifying the exploration of large design spaces," in *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–6, 2019.

[36] L. Zaman, "User interfaces and difference visualizations for alternatives," 2015.

[37] R. J. Heuer, *Psychology of intelligence analysis*. Center for the Study of Intelligence, 1999.

[38] Y.-a. Kang and J. Stasko, "Characterizing the intelligence analysis process: Informing visual analytics design through a longitudinal field study," in *2011 IEEE conference on visual analytics science and technology (VAST)*, pp. 21–30, IEEE, 2011.

[39] E. C. Barrett, *The interpretation and exploitation of information in criminal investigations*. PhD thesis, University of Birmingham, 2009.

[40] M. K. Dhami, I. K. Belton, and D. R. Mandel, "The "analysis of competing hypotheses" in intelligence analysis," *Applied Cognitive Psychology*, vol. 33, no. 6, pp. 1080–1090, 2019.

[41] M. Golfarelli, S. Rizzi, and A. Proli, "Designing what-if analysis: towards a methodology," in *Proceedings of the 9th ACM international workshop on Data warehousing and OLAP*, pp. 51–58, 2006.

[42] S. Guo, F. Du, S. Malik, E. Koh, S. Kim, Z. Liu, D. Kim, H. Zha, and N. Cao, "Visualizing uncertainty and alternatives in event sequence predictions," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–12, 2019.

[43] C. Hill, R. Bellamy, T. Erickson, and M. Burnett, "Trials and tribulations of developers of intelligent systems: A field study," in *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pp. 162–170, IEEE, 2016.

[44] J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. Viégas, and J. Wilson, "The what-if tool: Interactive probing of machine learning models," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 1, pp. 56–65, 2019.

[45] H. Strobelt, S. Gehrmann, M. Behrisch, A. Perer, H. Pfister, and A. M. Rush, "Seq2seq-vis: A visual debugging tool for sequence-to-sequence models," *IEEE transactions on visualization and computer graphics*, vol. 25, no. 1, pp. 353–363, 2018.

[46] M. B. Kery, M. Radensky, M. Arya, B. E. John, and B. A. Myers, "The story in the notebook: Exploratory data science using a literate programming tool," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, (New York, NY, USA), pp. 1–11, Association for Computing Machinery, 2018.

[47] J. Stasko, C. Görg, and Z. Liu, "Jigsaw: supporting investigative analysis through interactive visualization," *Information visualization*, vol. 7, no. 2, pp. 118–132, 2008.

[48] L. Bavoil, S. P. Callahan, P. J. Crossno, J. Freire, C. E. Scheidegger, C. T. Silva, and H. T. Vo, "Vistrails: Enabling interactive multiple-view visualizations," in *VIS 05. IEEE Visualization, 2005.*, pp. 135–142, IEEE, 2005.

[49] J. Wood, A. Kachkaev, and J. Dykes, "Design exposition with literate visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 759–768, 2019.

[50] A. Kale, M. Kay, and J. Hullman, "Decision-making under uncertainty in research synthesis: Designing for the garden of forking paths," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, (New York, NY, USA), pp. 1–14, Association for Computing Machinery, 2019.

[51] H. Guo and D. H. Laidlaw, "Topic-based exploration and embedded visualizations for research idea generation," *IEEE transactions on visualization and computer graphics*, 2018.

[52] P. J. Guo, *Software tools to facilitate research programming*. Stanford University, 2012.

[53] P. J. Huber, *Data analysis: what can be learned from the past 50 years*, vol. 874. John Wiley & Sons, 2012.

[54] Y. S. Yoon and B. A. Myers, "A longitudinal study of programmers' backtracking," in *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pp. 101–108, IEEE, 2014.

[55] M. P. Robillard, "What makes apis hard to learn? answers from developers," *IEEE software*, vol. 26, no. 6, pp. 27–34, 2009.

[56] A. Lunzer and K. Hornbæk, "Subjunctive interfaces: Extending applications to support parallel setup, viewing and control of alternative scenarios," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 14, no. 4, pp. 1–44, 2008.

[57] M. Toomim, A. Begel, and S. L. Graham, "Managing duplicated code with linked editing," in *2004 IEEE Symposium on Visual Languages-Human Centric Computing*, pp. 173–180, IEEE, 2004.

[58] M. Terry, E. D. Mynatt, K. Nakakoji, and Y. Yamamoto, "Variation in element and action: supporting simultaneous development of alternative solutions," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 711–718, 2004.

[59] S. Kolarić, R. Woodbury, and H. Erhan, "Cambria: a tool for managing multiple design alternatives," in *Proceedings of the 2014 companion publication on Designing interactive systems*, pp. 81–84, 2014.

[60] M. Hofmann and R. Klinkenberg, *RapidMiner: Data mining use cases and business analytics applications*. CRC Press, 2016.

[61] K. Patel, N. Bancroft, S. M. Drucker, J. Fogarty, A. J. Ko, and J. Landay, "Gestalt: integrated support for implementation and analysis in machine learning," in *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pp. 37–46, 2010.

[62] R. Fiebrink, P. R. Cook, and D. Trueman, "Human model evaluation in interactive supervised learning," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 147–156, 2011.

[63] W. Cancino, N. Boukhelifa, and E. Lutton, "Evographdice: Interactive evolution for visual analytics," in *2012 IEEE Congress on Evolutionary Computation*, pp. 1–8, IEEE, 2012.

[64] D. E. Knuth, "Literate programming," *The Computer Journal*, vol. 27, no. 2, pp. 97–111, 1984.

[65] M. B. Kery, B. E. John, P. O'Flaherty, A. Horvath, and B. A. Myers, "Towards effective foraging by data scientists to find past analysis choices," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, (New York, NY, USA), pp. 1–13, Association for Computing Machinery, 2019.

[66] A. Mathisen, T. Horak, C. N. Klokmose, K. Grønbæk, and N. Elmqvist, "Insideinsights: Integrating data-driven reporting in collaborative visual analytics," in *Computer Graphics Forum*, vol. 38, pp. 649–661, Wiley Online Library, 2019.

[67] Y.-a. Kang, C. Gorg, and J. Stasko, "Evaluating visual analytics systems for investigative analysis: Deriving design principles

from a case study," in *2009 IEEE Symposium on Visual Analytics Science and Technology*, pp. 139–146, IEEE, 2009.

[68] Y.-a. Kang, C. Gorg, and J. Stasko, "How can visual analytics assist investigative analysis? design implications from an evaluation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 5, pp. 570–583, 2010.

[69] Y.-a. Kang and J. Stasko, "Examining the use of a visual analytics system for sensemaking tasks: Case studies with domain experts," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2869–2878, 2012.

[70] Y. B. Shrinivasan and J. J. van Wijk, "Supporting the analytical reasoning process in information visualization," in *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 1237–1246, 2008.

[71] C. Andrews, A. Endert, and C. North, "Space to think: large high-resolution displays for sensemaking," in *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 55–64, ACM, 2010.

[72] M. Beaudouin-Lafon, S. Huot, M. Nancel, W. Mackay, E. Pietriga, R. Primet, J. Wagner, O. Chapuis, C. Pillias, J. Eagan, *et al.*, "Multisurface interaction in the wild room," *Computer*, vol. 45, no. 4, pp. 48–56, 2012.

[73] H. Chung, C. North, S. Joshi, and J. Chen, "Four considerations for supporting visual analysis in display ecologies," in *2015 IEEE conference on visual analytics science and technology (VAST)*, pp. 33–40, IEEE, 2015.

[74] H. Chung and C. North, "Savil: cross-display visual links for sensemaking in display ecologies," *Personal and Ubiquitous Computing*, vol. 22, no. 2, pp. 409–431, 2018.

[75] V. Bush *et al.*, "As we may think," *The atlantic monthly*, vol. 176, no. 1, pp. 101–108, 1945.

[76] T. H. Nelson, "Complex information processing: a file structure for the complex, the changing and the indeterminate," in *Proceedings of the 1965 20th national conference*, pp. 84–100, 1965.

[77] T. H. Nelson, "The heart of connection: hypermedia unified by transclusion," *Communications of the ACM*, vol. 38, no. 8, pp. 31–33, 1995.

[78] P. Tchernavskij, *Transclusion in Software for the Complex, the Changing, and the Indeterminate*. PhD thesis, Aarhus Universitet, Datalogisk Institut, 2016.

[79] C. Chronicles, "Hypercard," 1990.

[80] C. Chronicles, "Hypercard," 2004.

[81] B. Myers, S. E. Hudson, and R. Pausch, "Past, present, and future of user interface software tools," *Transactions on Computer-Human Interaction*, vol. 7, no. 1, pp. 3–28, 2000.

[82] M. Beaudouin-Lafon, "Instrumental interaction: an interaction model for designing post-wimp user interfaces," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 446–453, 2000.

[83] B. Shneiderman, "1.1 direct manipulation: a step beyond programming languages," *Sparks of innovation in human-computer interaction*, vol. 17, p. 1993, 1993.

[84] M. Beaudouin-Lafon and W. E. Mackay, "Reification, polymorphism and reuse: three principles for designing visual interfaces," in *Proceedings of the working conference on Advanced visual interfaces*, pp. 102–109, 2000.

[85] G. Leiva, N. Maudet, W. Mackay, and M. Beaudouin-Lafon, "Enact: Reducing designer–developer breakdowns when prototyping custom interactions," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 26, no. 3, pp. 1–48, 2019.

[86] M. Ciolfi Felice, N. Maudet, W. E. Mackay, and M. Beaudouin-Lafon, "Beyond snapping: Persistent, tweakable alignment and distribution with stickylines," in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pp. 133–144, 2016.

[87] P. Tchernavskij, C. N. Klokmose, and M. Beaudouin-Lafon, "What can software learn from hypermedia?," in *Companion to the First International Conference on the Art, Science and Engineering of Programming*, Programming '17, (New York, NY, USA), Association for Computing Machinery, 2017.

[88] R. Rädle, M. Nouwens, K. Antonsen, J. R. Eagan, and C. N. Klokmose, "Codestrates: Literate computing with webstrates," in *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, UIST '17, (New York, NY, USA), pp. 715–725, Association for Computing Machinery, 2017.

[89] C. N. Klokmose, C. Remy, J. B. Kristensen, R. Bagge, M. Beaudouin-Lafon, and W. Mackay, "Videostrates: Collaborative, distributed and programmable video manipulation," in *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, UIST '19, (New York, NY, USA), pp. 233–247, Association for Computing Machinery, 2019.

[90] S. K. Badam, A. Mathisen, R. Rädle, C. N. Klokmose, and N. Elmqvist, "Vistrates: A component model for ubiquitous analytics," *IEEE transactions on visualization and computer graphics*, vol. 25, no. 1, pp. 586–596, 2018.

[91] S. Bødker, "Through the interface-a human activity approach to user interface design," *DAIMI Report Series*, no. 224, 1991.

[92] V. Kaptelinin and B. A. Nardi, *Acting with technology: Activity theory and interaction design*. MIT press, 2006.

[93] T. Horak, S. K. Badam, N. Elmqvist, and R. Dachselt, "When david meets goliath: Combining smartwatches with a large vertical display for visual data exploration," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, p. 19, ACM, 2018.

[94] F. Brudy, S. Houben, N. Marquardt, and Y. Rogers, "Curationspace: Cross-device content curation using instrumental interaction," in *Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces*, pp. 159–168, ACM, 2016.

[95] K. Charmaz, *Constructing grounded theory: A practical guide through qualitative analysis*. Sage, 2006.

[96] J. Lave, *Cognition in practice: Mind, mathematics and culture in everyday life*. Cambridge University Press, 1988.

[97] R. Peng, "Divergent and convergent phases of data analysis," 2018.

[98] H. Fangohr, "A gallery of interesting jupyter notebooks," 2018. https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks.

[99] A. Singh and S. Malhotra, "A researcher's guide to running diary studies," in *Proceedings of the 11th Asia Pacific Conference on Computer Human Interaction*, APCHI '13, (New York, NY, USA), pp. 296–300, Association for Computing Machinery, 2013.

[100] A. Y. Wang, Z. Wu, C. Brooks, and S. Oney, "Callisto: Capturing the "why" by connecting conversations with computational narratives," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, (New York, NY, USA), pp. 1–13, Association for Computing Machinery, 2020.

[101] H. Beyer and K. Holtzblatt, *Contextual design: defining customer-centered systems*, vol. 1. Morgan kaufmann, 1998.

[102] R. Scupin, "The kj method: A technique for analyzing data derived from japanese ethnology," *Human organization*, pp. 233–237, 1997.

[103] S. M. Winchip, "Affinity and interrelationship digraph: A qualitative approach to identifying organizational issues in a graduate program," *College Student Journal*, vol. 35, no. 2, pp. 250–250, 2001.

[104] G. Harboe and E. M. Huang, "Real-world affinity diagramming practices: Bridging the paper-digital gap," in *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, pp. 95–104, ACM, 2015.

[105] I. Larsen-Ledet, H. Korsgaard, and S. Bødker, "Collaborative writing across multiple artifact ecologies," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–14, 2020.

[106] T. Wynn, "Layers of thinking in tool behavior. in (kr gibson and t. ingold eds.) tools and cognition in human evolution," 1993.

[107] A. Lucero, "Using affinity diagrams to evaluate interactive prototypes," in *IFIP Conference on Human-Computer Interaction*, pp. 231–248, Springer, 2015.

[108] T. K. Judge, P. S. Pyla, D. S. McCrickard, S. Harrison, and H. R. Hartson, "Studying group decision making in affinity diagramming," tech. rep., Department of Computer Science, Virginia Polytechnic Institute & State University, 2008.

[109] E. Brady, M. R. Morris, Y. Zhong, S. White, and J. P. Bigham, "Visual challenges in the everyday lives of blind people," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2117–2126, ACM, 2013.

[110] C. Harrison, J. Horstman, G. Hsieh, and S. Hudson, "Unlocking the expressivity of point lights," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1683–1692, ACM, 2012.

[111] T. Olsson, E. Lagerstam, T. Kärkkäinen, and K. Väänänen-Vainio-Mattila, "Expected user experience of mobile augmented reality services: a user study in the context of shopping centres," *Personal and ubiquitous computing*, vol. 17, no. 2, pp. 287–304, 2013.

[112] A. Spiridonidou, I. Kampi, and K. Chorianopoulos, "Exploring everyday life in remote schools: A large-scale study with cultural probes and affinity diagrams," in *IDC 2010 Digital Technologies and Marginalized Youth Workshop*, Citeseer, 2010.

[113] N. Boukhelifa, M.-E. Perrin, S. Huron, and J. Eagan, "How data workers cope with uncertainty: A task characterisation study," in *Proceedings of the 2017 CHI Conference on Human Factors in*

Computing Systems, CHI '17, (New York, NY, USA), pp. 3645–3656, ACM, 2017.

[114] E. R. Burtner, R. A. May, R. E. Scarberry, R. R. LaMothe, and A. Endert, "Affinity+: Semi-structured brainstorming on large displays," in *Powerwall international workshop on ultra-high-resolution displays, CHI '13 Extended Abstracts*, p. 6, ACM, 2013.

[115] F. Geyer, U. Pfeil, A. Höchtl, J. Budzinski, and H. Reiterer, "Designing reality-based interfaces for creative group work," in *Proceedings of the 8th ACM conference on Creativity and cognition*, pp. 165–174, ACM, 2011.

[116] S. Klemmer, M. Newman, R. Farrell, R. Meza, and J. A. Landay, "A tangible evolution: System architecture and participatory design studies of the designers 'outpost," *NCSTRL. UCB/CSD-00*, vol. 1117, 2000.

[117] S. R. Klemmer, M. W. Newman, R. Farrell, M. Bilezikjian, and J. A. Landay, "The designers' outpost: A tangible interface for collaborative web site," in *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, UIST '01, (New York, NY, USA), pp. 1–10, Association for Computing Machinery, 2001.

[118] F. Geyer, U. Pfeil, J. Budzinski, A. Höchtl, and H. Reiterer, "Affinitytable-a hybrid surface for supporting affinity diagramming," in *IFIP Conference on Human-Computer Interaction*, pp. 477–484, Springer, 2011.

[119] P. Pirolli and S. Card, "The sensemaking process and leverage points for analyst technology as identified through cognitive task

analysis," in *Proceedings of international conference on intelligence analysis*, vol. 5, pp. 2–4, McLean, VA, USA, 2005.

[120] K. M. Everitt, S. R. Klemmer, R. Lee, and J. A. Landay, "Two worlds apart: Bridging the gap between physical and virtual media for distributed design collaboration," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, (New York, NY, USA), pp. 553–560, Association for Computing Machinery, 2003.

[121] S. Carpendale, S. Knudsen, A. Thudt, and U. Hinrichs, "Analyzing qualitative data," in *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*, pp. 477–481, ACM, 2017.

[122] J. Saldaña, *The coding manual for qualitative researchers*. Sage, 2015.

[123] J. Corbin and A. Strauss, *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage publications, 2014.

[124] B. F. Crabtree and W. F. Miller, "A template approach to text analysis: developing and using codebooks.," 1992.

[125] T. J. Schmidlen, L. Wawak, R. Kasper, J. F. García-España, M. F. Christman, and E. S. Gordon, "Personalized genomic results: analysis of informational needs," *Journal of genetic counseling*, vol. 23, no. 4, pp. 578–587, 2014.

[126] A. Rafalovich, "Broken and becoming god-sized: Contemporary metal music and masculine individualism," *Symbolic Interaction*, vol. 29, no. 1, pp. 19–32, 2006.

[127] V. Elliott, "Thinking about the coding process in qualitative data analysis," *Qualitative Report*, vol. 23, no. 11, 2018.

[128] E. Blair, "A reflexive exploration of two qualitative data coding techniques," *Journal of Methods and Measurement in the Social Sciences*, vol. 6, no. 1, pp. 14–29, 2015.

[129] S. Chandrasegaran, S. K. Badam, L. Kisselburgh, K. Ramani, and N. Elmqvist, "Integrating visual analytics support for grounded theory practice in qualitative text analysis," in *Computer Graphics Forum*, vol. 36, pp. 201–212, Wiley Online Library, 2017.

[130] J. M. Corbin and A. Strauss, "Grounded theory research: Procedures, canons, and evaluative criteria," *Qualitative sociology*, vol. 13, no. 1, pp. 3–21, 1990.

[131] F. Scharf, C. Wolters, M. Herczeg, and J. Cassens, "Cross-device interaction definition, taxonomy and applications," 2013.

[132] J. Grubert, M. Kranz, and A. Quigley, "Challenges in mobile multi-device ecosystems," *mUX: The Journal of Mobile User Experience*, vol. 5, no. 1, p. 5, 2016.

[133] T. Judge, R. Kopper, S. Ponce, M. Silva, and C. North, "Babes: Brushing+linking, attributes, and blobs extension to storyboard," tech. rep., Department of Computer Science, Virginia Polytechnic Institute & State University, Jan 2008.

[134] M. Mateescu, C. Pimmer, C. Zahn, D. Klinkhammer, and H. Reiterer, "Collaboration on large interactive displays: a systematic review," *Human–Computer Interaction*, vol. 36, no. 3, pp. 243–277, 2021.

[135] D. R. Olsen Jr, "Evaluating user interface systems research," in *Proceedings of the 20th annual ACM symposium on User interface software and technology*, pp. 251–258, 2007.

[136] E. Tse, S. Greenberg, C. Shen, C. Forlines, and R. Kodama, "Exploring true multi-user multimodal interaction over a digital table," in *Proceedings of the 7th ACM conference on Designing interactive systems*, pp. 109–118, 2008.

[137] T. Moscovich, F. Chevalier, N. Henry, E. Pietriga, and J.-D. Fekete, "Topology-aware navigation in large networks," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2319–2328, 2009.

[138] A. Bezerianos and R. Balakrishnan, "View and space management on large displays," *IEEE Computer Graphics and Applications*, vol. 25, no. 4, pp. 34–43, 2005.

[139] A. Prouzeau, A. Bezerianos, and O. Chapuis, "Awareness techniques to aid transitions between personal and shared workspaces in multi-display environments," in *Proceedings of the 2018 ACM International Conference on Interactive Surfaces and Spaces*, pp. 291–304, 2018.

[140] P. Dourish and V. Bellotti, "Awareness and coordination in shared workspaces," in *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, pp. 107–114, 1992.

[141] S. Lau, I. Drosos, J. M. Markel, and P. J. Guo, "The design space of computational notebooks: An analysis of 60 systems in academia and industry," in *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pp. 1–11, IEEE, 2020.

[142] A. Y. Wang, A. Mittal, C. Brooks, and S. Oney, "How data scientists use computational notebooks for real-time collaboration," *Proc. ACM Hum.-Comput. Interact.*, vol. 3, Nov. 2019.

[143] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. B. Hamrick, J. Grout, S. Corlay, *et al.*, "Jupyter notebooks-a publishing format for reproducible computational workflows.," in *ELPUB*, pp. 87–90, 2016.

[144] I. R. Brian Granger, Chris Colbert, "Jupyterlab:the next generation jupyter frontend," *JupyterCon*, 2017.

[145] J. J. Thomas, *Illuminating the path:[the research and development agenda for visual analytics]*. IEEE Computer Society, 2005.

[146] T. Munzner, *Visualization analysis and design*. CRC press, 2014.

[147] N. Elmqvist, A. V. Moere, H.-C. Jetter, D. Cernea, H. Reiterer, and T. Jankun-Kelly, "Fluid interaction for information visualization," *Information Visualization*, vol. 10, no. 4, pp. 327–340, 2011.

[148] S. Chattopadhyay, I. Prasad, A. Z. Henley, A. Sarma, and T. Barik, "What's wrong with computational notebooks? pain points, needs, and design opportunities," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, (New York, NY, USA), pp. 1–12, Association for Computing Machinery, 2020.

[149] K. Subramanian, I. Zubarev, S. Völker, and J. Borchers, "Supporting data workers to perform exploratory programming," in *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–6, 2019.

[150] A. Tabard, W. E. Mackay, and E. Eastmond, "From individual to collaborative: The evolution of prism, a hybrid laboratory notebook," in *Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work*, CSCW '08, (New York, NY, USA), pp. 569–578, Association for Computing Machinery, 2008.

[151] C. N. Klokmose and P.-O. Zander, "Rethinking laboratory notebooks," in *Proceedings of COOP 2010* (M. Lewkowicz, P. Hassanaly, V. Wulf, and M. Rohde, eds.), (London), pp. 119–139, Springer London, 2010.

[152] T. H. Nelson *et al.*, "Literary machines: The report on, and of, project xanadu, concerning word processing, electronic publishing, hypertext, thinkertoys, tomorrow's intellectual revolution, and certain other topics including knowledge, education and freedom," 1981.

[153] H. Krottmaier and D. Helic, "Issues of transclusions," in *E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, pp. 1730–1733, Association for the Advancement of Computing in Education (AACE), 2002.

[154] M. Bernstein, "Collage, composites, construction," in *Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pp. 122–123, 2003.

[155] A. Di Iorio and J. Lumley, "From xml inclusions to xml transclusions," in *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, pp. 147–156, 2009.

[156] C. N. Klokmose, J. Eagan, S. Baader, W. Mackay, and M. Beaudouin-Lafon, "Webstrates: Demonstrating the potential

of shareable dynamic media," in *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion*, CSCW '16 Companion, (New York, NY, USA), pp. 61–64, Association for Computing Machinery, 2016.

[157] J. Edwards, "Managed copy & paste," 2018. https://vimeo.com/287270545.

[158] M. Borowski, R. Rädle, and C. N. Klokmose, "Codestrate packages: An alternative to "one-size-fits-all" software," in *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI EA '18, (New York, NY, USA), pp. 1–6, Association for Computing Machinery, 2018.

[159] S. K. Badam and N. Elmqvist, "Polychrome: A cross-device framework for collaborative web visualization," in *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces*, ITS '14, (New York, NY, USA), pp. 109–118, Association for Computing Machinery, 2014.

[160] S. K. Badam and N. Elmqvist, "Visfer: Camera-based visual data transfer for cross-device visualization," *Information Visualization*, vol. 18, no. 1, pp. 68–93, 2019.

[161] W. McGrath, B. Bowman, D. McCallum, J. D. Hincapié-Ramos, N. Elmqvist, and P. Irani, "Branch-explore-merge: facilitating real-time revision control in collaborative visual exploration," in *Proceedings of the 2012 ACM international conference on Interactive tabletops and surfaces*, pp. 235–244, 2012.

[162] M. Schwab, D. Saffo, Y. Zhang, S. Sinha, C. Nita-Rotaru, J. Tompkin, C. Dunne, and M. A. Borkin, "Visconnect: Distributed event synchronization for collaborative visualization,"

IEEE Transactions on Visualization and Computer Graphics, 2020.

[163] R. Neogy, J. Zong, and A. Satyanarayan, "Representing real-time multi-user collaboration in visualizations," *arXiv preprint arXiv:2009.02587*, 2020.

[164] R. Langner, T. Horak, and R. Dachselt, "V is t iles: Coordinating and combining co-located mobile devices for visual data exploration," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 1, pp. 626–636, 2017.

[165] A. A. diSessa and H. Abelson, "Boxer: A reconstructible computational medium," *Communications of the ACM*, vol. 29, no. 9, pp. 859–868, 1986.

[166] S. Greenberg and B. Buxton, "Usability evaluation considered harmful (some of the time)," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 111–120, 2008.

[167] A. Salovaara, A. Oulasvirta, and G. Jacucci, "Evaluation of prototypes and the problem of possible futures," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pp. 2064–2077, 2017.

[168] B. Brown, S. Reeves, and S. Sherwood, "Into the wild: challenges and opportunities for field trial methods," in *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 1657–1666, 2011.

[169] H. Hutchinson, W. Mackay, B. Westerlund, B. B. Bederson, A. Druin, C. Plaisant, M. Beaudouin-Lafon, S. Conversy,

H. Evans, H. Hansen, *et al.*, "Technology probes: inspiring design for and with families," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 17–24, 2003.

DOCTORAL SCHOOL

Titre : Data Expression: Comprendre et soutenir les alternatives dans les processus d'analyse des données

Mots clés : analyse des données, sensemaking, alternatives, visualisation, interaction multisurface

Résumé : Pour bien comprendre les données, les analystes considèrent différents types d' alternatives: Ils explorent diverses hypothèses, essayent différents types de méthodes et expérimentent un large éventail de solutions possibles. Ces alternatives s'influencent mutuellement dans un processus dynamique et complexe de "sensemaking". Pourtant, les outils d'analyse actuels considèrent rarement ces alternatives comme une partie intégrante de l'analyse, ce qui rend le processus lourds et cognitif exigeants. En appliquant diverses méthodes empiriques et conceptions d'outils, nous répondons aux questions : (1) Quelles sont les alternatives et comment s'intègrent-elles dans le processus de création de sens ? et (2) comment les outils peuvent-ils mieux soutenir l'exploration et la gestion des alternatives ?
Cette thèse comprend trois parties : La partie I explore le rôle des alternatives à travers des entretiens et des observations avec des analystes. Sur la base des résultats et de notre analyse, nous apportons des caractérisations des alternatives et un cadre pour aider à les décrire et à les raisonner. La partie II se concentre sur le soutien des alternatives dans le contexte du "affinity diagramming" pour l'analyse des données qualitatives. Sur la base des entretiens avec des praticiens et à notre propre expérience, nous proposons un design space pour caractériser les différents types d'alternatives engagées dans un tel processus de sensemaking. Nous fournissons une vision et un système de preuve de concept, ADQDA, pour montrer comment les analystes peuvent effectuer des transitions fluide entre des phases d'analyse, des méthodes et des représentations alternatives, et comment ils peuvent s'approprier de manière flexible divers dispositifs pour s'adapter aux tâches à accomplir ou pour étendre l'espace d'analyse. La troisième partie traite des alternatives dans le contexte de la réutilisation. Nous envisageons une nouvelle technique de réutilisation, la "computational transclusion", qui maintient divers liens dynamiques entre le contenu original et le contenu réutilisé (les alternatives) pour faciliter le suivi et la coordination des changements. Nous avons construit un système pour sonder différents scénarios de réutilisation et explorer les différents "links" entre les alternatives et leurs réifications possibles dans les interfaces utilisateurs.

Title : Data Expression: Understanding and Supporting Alternatives in Data Analysis Processes

Keywords : Data Analysis, sensemaking, alternatives, visualization, multi-surface interaction

Abstract : To make sense of data, analysts consider different kinds of alternatives: they explore diverse sets of hypotheses, try out different types of methods, and experiment with a broad space of possible solutions. These alternatives influence each other within a dynamic and complex sensemaking process. Current analytic tools, however, rarely consider such alternatives as an integral part of the analysis, making the process cumbersome and cognitively demanding. Applying various empirical methods and tool designs, we address the following questions: (1) What are alternatives and how do they fit within the sensemaking process?And (2) how can tools better support the exploration and management of alternatives?
This dissertation contains three parts: Part I explores the role of alternatives through interviews and observations with analysts. Based on the results and our analysis, we contribute characterisations of alternatives and a framework to help describe and reason about them. Part II focuses on supporting alternatives in the context of affinity diagramming for qualitative data analysis. Through interviews with practitioners and combined with our own experience, we propose a design space to characterise the various kinds of alternatives engaged in such sensemaking process. We further provide a vision and proof-of-concept system, ADQDA, to show how analysts can fluidly transition between alternative analysis phases, methods, representations, and how they can flexibly appropriate various devices to suit for the tasks at hand or to extend the analysis space. Part III discusses alternatives in the context of reuse. We envision a novel reuse technique, "computational transclusion", which maintains various dynamic links between the original and the reused contents (the alternatives) to facilitate tracking and coordinating changes. We built a sandbox system to probe into different reuse scenarios and explore the various links between alternatives and their possible reifications in notebook-ish user interface.

Institut Polytechnique de Paris
91120 Palaiseau, France