UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

STATISTICAL ANOMALY DISCOVERY THROUGH VISUALIZATION

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR of PHILOSOPHY

By

Chenguang Xu

Norman, Oklahoma

2023

Downloaded from SHAREOK. Persistent Handle URL: https://hdl.handle.net/11244/337584

STATISTICAL ANOMALY DISCOVERY THROUGH VISUALIZATION

A DISSERTATION APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY THE COMMITTEE CONSISTING OF

Dr. Chris Weaver, Chair

Dr. Christan Grant

Dr. Amy McGovern

Dr. Dean Hougen

Dr. William Endres

© Copyright by CHENGUANG XU 2023

All rights Reserved.

Acknowledgements

I would like to thank my advisor Dr. Chris Weaver for his support and encouragement. He brought me to the research area that I am strongly passionate about. He nominated me as the volunteer chair for the IEEE Conference on Visualization (VIS) conference 2022 in Oklahoma City. That brought me closer to the VIS community, and I had a chance to meet a lot of great researchers. I enjoy discussing new research ideas with Dr. Weaver, and his valuable advice and critical feedback guided me to complete this thesis.

I also would like to thank my advisor Dr. Christan Grant. He helped me explore the road on my research. He created many opportunities for me to do my research. He supported me to attend the 2018 SIGMOD conference in Houston, the 2018 FLAIRS Conference, and the 2018 technique meeting hosted by COE TTHP in Philadelphia. I had invaluable experience sharing research ideas and had a chance to talk to mature researchers in those venues.

I am also very grateful to my dissertation committee—Dr. Amy McGovern, Dr. Dean Hougen, Dr. Bill Endres—for their time and advice. Their encouraging words inspired me to continue when there was so much uncertainty in my life.

A special thanks to Dr. Sarah Brown for her collaboration and suggestions. She provided significant advice and technical support for building Wiggum. I appreciate her valuable feedback.

Grateful thanks to the students, library staff, and professors who participated in the user study assessing the Wiggum system.

I also thank Dr. Robert Dauffenbach and Shawn Lam for their support throughout this process.

Most of all, I would like to thank my family and friends for their love and support in my life.

Table of Contents

1 Introduction **1**

1.1 Overview 1

1.2 Motivation 3

1.3 Research Path 4

1.4 Thesis Statement and Research Contributions 10

1.5 Research Questions and Scope 11

1.6 Organization of the Dissertation 12

2 Background and Related Work **14**

2.1 Mix Effects and Simpson's Paradox 14

2.2 Visualization of Mix Effects 16

2.3 Fairness Forensics Tools 17

2.4 Visualization 22

2.4.1 Visual Perception 22

2.4.2 Visual Comparison 24

2.4.3 Visual Linking 25

2.4.4 Multiple Views 26

2.4.5 Design Space of Visualization 27

2.4.6 Visualization of Hierarchies 28

3 Automatically Detecting Simpson's Paradox **30**

3.1 Overview . . . 30
3.2 Detecting Simpson's Paradox . . . 31
3.3 Example Applications . . . 33
3.3.1 Synthetic Data Set . . . 33
3.3.2 Iris Data Set . . . 34
3.3.3 Auto MPG Data Set . . . 37
3.4 Performance Evaluation . . . 38
3.5 Discussion . . . 40
3.6 Simpson's Paradox in Partial Data . . . 42
3.7 Summary . . . 43

4 Representing Simpson's Paradox with Bivariate Colors 44
4.1 Overview . . . 44
4.2 Example Data . . . 45
4.2.1 Rank Trend Simpson's Paradox . . . 45
4.2.2 Regression Trend Simpson's Paradox . . . 46
4.3 Tool Design . . . 47
4.3.1 Bivariate Color Scheme . . . 47
4.3.2 Bivariate-Scaled Heatmaps . . . 48
4.4 Visual Analysis Workflow . . . 50
4.4.1 Rank Trend Simpson's Paradox . . . 50
4.4.2 Regression Trend Simpson's Paradox . . . 51
4.5 Summary . . . 53

5 Wiggum: Interactive Visual Analytics for Examining Mix Effects 54
5.1 Overview . . . 54
5.2 Illustrative Example . . . 57
5.3 Data Preparation and Processing . . . 59

5.3.1 Data Preparation . . . 59
5.3.2 Trend Extraction . . . 61
5.3.3 Trend Measurement . . . 64
5.4 Wiggum: Discovering Mix Effects . . . 66
5.4.1 Design Goals . . . 67
5.4.2 Preparation Page . . . 68
5.4.3 Result Table View . . . 68
5.4.4 Distance Heatmap View . . . 69
5.4.5 Detail Views . . . 71
5.4.6 The Control Panel Area . . . 73
5.4.7 Implementation . . . 75
5.5 Summary . . . 76

6 Evaluation of Wiggum **77**
6.1 Overview . . . 77
6.2 Use Cases . . . 77
6.2.1 Regression Analysis . . . 78
6.2.2 Binary Ranking Analysis . . . 79
6.2.3 Multiple Ranking Analysis . . . 80
6.3 Evaluation . . . 83
6.3.1 Participants . . . 83
6.3.2 Procedure . . . 84
6.3.3 Data and Tasks . . . 84
6.3.4 Correctness . . . 86
6.3.5 Time vs. Task Score . . . 87
6.3.6 Qualitative Results . . . 87
6.4 Limitations and Future Work . . . 89
6.4.1 Trends Types and Comparisons . . . 89

6.4.2 Visualization Design 90
6.4.3 Causality 91
6.4.4 Scalability 91
6.5 Summary 92

7 A Model of Visual Correspondence 93
7.1 Overview 93
7.2 A Model of Visual Correspondence 95
7.2.1 Information Aspects 95
7.2.2 Visual Channels 97
7.2.3 Visual Cardinality 99
7.2.4 Degree of Correspondence 100
7.3 Applying the Model 101
7.3.1 General Method 101
7.3.2 Populating the Model 102
7.3.3 Observations and Patterns 103
7.3.4 Applying the Results 105
7.4 Discussion 109
7.5 Guidelines 112
7.6 Toward Systematic Evaluation of Correspondence 116
7.6.1 Visual Presence within a Visual Encoding 117
7.6.2 Visual Correspondence between Visual Encodings 118
7.6.3 Discussion 119
7.7 Summary 119

8 PatternTree: A Hybrid Tree Visualization for Hierarchical Patterns 121
8.1 Overview 121
8.2 Virtual Layering 123

8.2.1 Identity Portion . . . 123
8.2.2 View Portion . . . 124
8.2.3 Visual Compositions in the Virtual Layer . . . 124
8.2.4 Positioning View Portion . . . 126
8.2.5 Association Between Identity and View Portions . . . 128
8.3 PatternTree Design . . . 132
8.3.1 Hierarchical Pattern Structure . . . 132
8.3.2 Visual Design . . . 132
8.4 Use Case: Gerrymandering . . . 133
8.5 Patterns in Gerrymandering . . . 135
8.6 Color Selection . . . 137
8.7 Implementation . . . 138
8.8 Analysis Examples . . . 139
8.8.1 Evaluating the Efficiency Gap . . . 139
8.8.2 Assessing Electoral Competition . . . 142
8.9 Visual Correspondence Assessment . . . 143
8.10 Limitations and Future Work . . . 148
8.10.1 Design Space . . . 148
8.10.2 Scalability . . . 149
8.11 Summary . . . 149

9 Conclusion **151**

9.1 Contributions . . . 151
9.2 Future Work . . . 153
9.2.1 Summary Statistics . . . 153
9.2.2 Tree Visualization . . . 153
9.2.3 Evaluation . . . 154
9.3 Conclusions . . . 154

List of Tables

1.1 The output from the algorithm in Chapter 3 for the Auto MPG data set [90]. 5

3.1 Per-group correlation matrices for a synthetic data set. 32
3.2 Result from our algorithm for the synthetic data set. 34
3.3 The output from our algorithm for the Iris data set. 35
3.4 The output from our algorithm for the Auto MPG data set, in the same form as Table 3.3. 37
3.5 Running time (in seconds) of the detection algorithm. 40

5.1 User-defined types and roles of features in instances of mix effects, for each of the two types of trends. 60

6.1 Summary of the three use cases. 78
6.2 The predefined roles—dependent, independent and splitby—of variables, the number of distance heatmaps, and their internal dimensions generated in the user study. 84
6.3 Tasks and questions in the user study. 85

8.1 Example: Precinct-Level Data for the 2016 and 2020 Presidential Elections in Oklahoma with Congressional Districts for the 2010 and 2020 Cycles. . . . 135

List of Figures

1.1 Simpsons' paradox occurring in an example data set with two subgroups [164]. 2

1.2 The application user interface for regression trend Simpson's paradox detection in the Auto MPG data set, using a bivariate color scheme. 6

1.3 The preparation page for the Auto MPG data set in Wiggum. 7

1.4 The visualization page for the Auto MPG data set in Wiggum. 8

1.5 Example of a PatternTree for the 2016 and 2020 presidential elections in Oklahoma. 9

3.1 Simpsons' paradox occurring in an example data set with two subgroups. . . 33

3.2 Visualization of Simpson's paradox in the Iris data set. 36

3.3 Simpson's paradox in the Auto MPG data set. 39

3.4 Running time of the algorithm for different samplings of the synthetic data set. 41

3.5 The F_1 score when running the Simpson's paradox over random samples of data. 42

4.1 Bivariate color legends for rank trend Simpson's paradox. 46

4.2 The application user interface for rank trend Simpson's paradox detection. . 51

4.3 The application user interface for regression trend Simpson's paradox detection. 52

4.4 The tree layout for regression trend Simpson's paradox. 53

5.1 Selecting a subgroup and examining trends on the Wiggum visualization page. 57

5.2 Subset generation. 62

5.3 The Auto MPG data set [90] in Wiggum. 66

5.4 Internal pipeline to generate a distance matrix heatmap from a result table. 70
5.5 A screenshot of a distance heatmap and its detail view. 71
5.6 A screenshot of a distance heatmap and its detail view. 72

6.1 The visualization page for the UC Berkeley Graduate Admissions data set. . 81
6.2 The visualization page for the adult data set. 82
6.3 Correct answers for each task-question-dataset combination. 86
6.4 Time vs. task score in each task level for the different data sets. 88

7.1 Three basic visual designs for considering visual correspondence. 96
7.2 Three-level degree of likeness for three different information aspects over common visual encoding channels. 97
7.3 Examples of populated matrices and three cases. 100
7.4 The combination matrices populated by our method. 104
7.5 The 12 example visualizations used to assess natural instances of correspondence. 106
7.6 Guidelines for considering visual correspondence. 112
7.7 An example of Guideline 3. 114
7.8 An example of Guideline 4. 115

8.1 Basic virtual layering concept illustrated in a node-link diagram. 124
8.2 Composite visual designs for the identity and view portion of the virtual layer. 125
8.3 Positioning the view portion for three different situations involving different view heights. 127
8.4 Visual manipulations and their applicability to common visualization types. 129
8.5 The hierarchical view design for the PatternTree. 133
8.6 An overview of the PatternTree system architecture. 139
8.7 An example demonstrating the basic design of the PatternTree technique for evaluating the efficiency gap. 140

8.8 Example of a PatternTree to assess electoral competition. 142
8.9 An example of refining choices of host and embedded visualizations in PatternTree based on the theory of visual correspondence. 144
8.10 Designs for embedding a heat map in the first level of a PatternTree. 145
8.11 Designs for embedding a scatterplot in the second level of the tree visualization. 146
8.12 Designs for embedding a map view in the leaf level. 147

Abstract

Developing a deep understanding of data is a crucial part of decision-making processes. It often takes substantial time and effort to develop a solid understanding to make well-informed decisions. Data analysts often perform statistical analyses through visualization to develop such understanding. However, applicable insight can be difficult due to biases and anomalies in data. An often overlooked phenomenon is mix effects, in which subgroups of data exhibit patterns opposite to the data as a whole. This phenomenon is widespread and often leads inexperienced analysts to draw contradictory conclusions. Discovering such anomalies in data becomes challenging as data continue to grow in volume, dimensionality, and cardinality. Effectively designed data visualizations empower data analysts to reveal and understand patterns in data for studying such paradoxical anomalies.

This research explores several approaches for combining statistical analysis and visualization to discover and examine anomalies in multidimensional data. It starts with an automatic anomaly detection method based on correlation comparison and experiments to determine the running time and complexity of the algorithm. Subsequently, the research investigates the design, development, and implementation of a series of visualization techniques to fulfill the needs of analysis through a variety of statistical methods. We create an interactive visual analysis system, *Wiggum*, for revealing various forms of mix effects. A user study to evaluate *Wiggum* strengthens understanding of the factors that contribute to the comprehension of statistical concepts. Furthermore, a conceptual model, *visual correspondence*, is presented to study how users can determine the identity of items between visual representations by interpreting the relationships between their respective visual encodings. It is practical to build visualizations with highly linked views informed by visual correspondence theory. We present a hybrid tree visualization technique, *PatternTree*, which applies the visual corre-

spondence theory. *PatternTree* supports users to more readily discover statistical anomalies and explore their relationships. Overall, this dissertation contributes a merging of new visualization theory and designs for analysis of statistical anomalies, thereby leading the way to the creation of effective visualizations for statistical analysis.

Chapter 1

Introduction

1.1 Overview

Developing a deep understanding of data is an essential part of decision-making processes. It often takes substantial time and effort to develop enough understanding to make well-informed decisions. Data analysts often perform statistical analyses to develop such understanding. This is especially common for large, multi-attribute data. A variety of statistical phenomena can manifest in data. The inability to examine them, or even mere ignorance of the phenomena themselves, can restrict analysis and lead to incorrect conclusions [18, 110].

Data analysts often use visualizations to explore data and examine trends in data. For purposes of discussion in this dissertation, we will define a *trend* as a tendency of a variable in relation to another variable that is changing in the data set, and focus on two *trend types*. A linear relationship between two variables defines a *regression trend* (short for *linear regression trend*). The ranking groups defined by one variable according to a statistic calculated on a second variable defines a *rank trend*. Both of these types of trends can be identified in a whole data set or in *subgroups* created by grouping instances based on the variable's distinct categories. A *mix effect* [13] occurs when a trend reverses in subgroups of data relative to the aggregate trend.

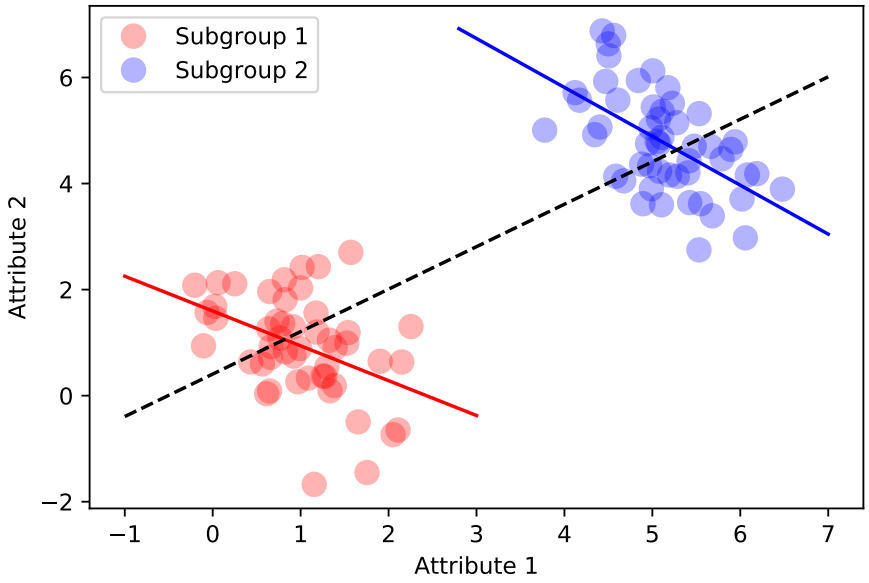


Figure 1.1: Simpsons' paradox occurring in an example data set with two subgroups [164]. Each subgroup has a full weight trendline. The dashed line shows the reversed trend for the entire data set.

A prominent case of mix effects arose in an audit of gender bias in graduate school admissions at the University of California, Berkeley [18]. University officials observed that, university wide, men were admitted at a higher rate than women; however, upon further inspection, this was not true within individual departments. If every department shows the reverse trend, it is a special case of mix effects known as *Simpson's paradox*, in which *all* subgroups of the data partitioned by a certain condition have trends opposite to the aggregate data.

Figure 1.1 shows a synthetic example of Simpson's paradox for a regression trend. A black-dotted line shows a regression line over the full data set, indicating a positive correlation. However, two subgroups of the data set, shown as red circles and blue circles, individually have negatively sloped regression lines. Grouping the data by the color attribute, we see an opposite relation between the overall trend and each of the two subgroup trends. If analysts draw incorrect conclusions based on data that has such disparities, lives and livelihoods may be affected.

1.2 Motivation

Data are used to assist the decision making process through multiple analytical approaches. Data mining and machine learning systems deployed in real world situations can reproduce and amplify biases [118, 87, 169]. Many of these quantitative disparities are related to differences in the makeup or structure of partitions of the data. These phenomena have fueled rapidly growing interest in fair machine learning, producing both a plethora of definitions and algorithmic solutions [15, 138]. However, fairness is not a uniquely quantifiable concept, so improving social outcomes of data mining systems requires subjective human intervention and reframing of the problem [127]. Kate Crawford first coined "Fairness Forensics" in her 2017 NeurIPS (Conference on Neural Information Processing Systems) Keynote "The Trouble with Bias" calling for more techniques to discover instances of unfairness in data [40].

Statistical analysts need better exploratory tools to draw reliable conclusions from high-dimensional data sets. Subject-area experts are used to looking for particular information in data and are accustomed to working with small data sets. As data grows not only in the number of samples, but more importantly in the richness of each sample, tools to assist with investigations become increasingly critical. The goal is to equip trained data scientists and analysts with powerful visual systems for conducting fairness forensics investigations in their exploratory data analysis and post-algorithm analysis workflows. Studies on statistical phenomena, such as Simpson's paradox, can be applied to detect bias in data for fairness forensics.

Simpson's paradox is a trend reversal in partitions of a data set. A linear relationship between two variables defines a *regression* (short for *linear regression*) trend and ranking groups defined by one variable according to a statistic calculated on a second variable defines a *rank* trend. Although analysts can study trends via summary statistics, they may not see important patterns in data since dramatically different data sets could produce the same statistical properties [96]. Good visualizations allow analysts to observe the patterns behind summary statistics. Taking this as a starting point, we endeavor to develop an

effectively designed visualization system to investigate instances of Simpson's paradox across each trend type. Unfortunately, the combinatorial explosion of dimensions is challenging in visualization design. To overcome this challenge, we sought to develop a visualization model for visualization designs to manage combinations of dimensions. The need for dimensional management led to the development of the visual correspondence theory and the PatternTree technique.

1.3 Research Path

The research has three stages. In the first stage, we develop the basic detecting algorithm for Simpson's paradox (Chapter 3), and use bivariate colors to represent Simpson's paradox visually (Chapter 4). Case studies on empirical data sets show the effectiveness of the algorithm and demonstrate the utility of our visualization design. In the second stage, we design the visualization system integrated with multiple detection approaches (Chapter 5), and perform an evaluation of the system (Chapter 6). We introduce a set of measures to support examination of mix effects for multiple trend types. The visualization design of the system aims to reduce the limitations of the bivariate color technique. In the third stage, we present a visualization model (Chapter 7) for addressing the issues that we found from the evaluation of the system, and apply the model to design an improved visualization technique for statistical anomaly detection (Chapter 8). The details of the research path follow.

This dissertation addresses the problem of discovering and examining statistical anomalies through a data analysis pipeline from statistical computation to interactive visualization for multidimensional data. The goal is to gain knowledge about the problem and potential solutions through a visual analysis system that combines back-end computation and front-end visualization. Therefore, this research intersects with both statistics and visualization as research fields. We deal with statistics first and start with one of the most well-studied, surprising trends in data, Simpson's paradox. Little work has been done to study how to

allCorr	attr1	attr2	revCorr	catAttr	subgroup
0.423	MPG	acceleration	-0.819	cylinders	3
0.423	MPG	acceleration	-0.341	cylinders	6
0.423	MPG	acceleration	-0.051	model year	75
0.423	MPG	acceleration	-0.051	model year	79
-0.778	MPG	horsepower	0.621	cylinders	3
-0.778	MPG	horsepower	0.013	cylinders	6

Table 1.1: The output from the algorithm in Chapter 3 for the Auto MPG data set [90]. The Auto MPG data set has 6 reverse trends. Each row is an instance of a reverse trend. For example, the regression trend between MPG and Acceleration for the entire data is positive (0.432), but the subgroup trend for Cylinder 3 is negative (-0.819) [164].

detect Simpson's paradox in multidimensional data. Simpson's paradox has been studied in two main forms: (1) ranking trends on relative rates and (2) linear trends. We focus on the latter and use linear correlation to measure a trend between two variables. We developed an algorithm for detecting regression based Simpson's paradox, and the rest of the work in this dissertation is based on it. The algorithm returns the result as tabular data, as shown in Table 1.1.

We build a back-end computational library for extending the detection algorithm. We add more algorithms for detecting ranking trend into the library, and explore visual techniques for visualizing Simpson's paradox. Choosing a bivariate color scheme can be a good start since it helps to see the relationship between two variables: a variable for the subgroup's statistical result (e.g., correlation coefficient), and a variable for the aggregate's statistical result. Users can perceive the relationship by identifying a color rather than comparing two numbers. Additionally, when we design the visualization for exploratory data analysis, it is crucial to provide context and explanation for revealing patterns behind summary statistics. We apply an overview plus detail interface design [130]. The overview shows contextual information, and the detail view serves to explain the statistical result. Since the result of our algorithm is derived from a correlation matrix, a correlation heatmap is used to represent the correlation between different variables by color. Figure 1.2 shows the visualization for

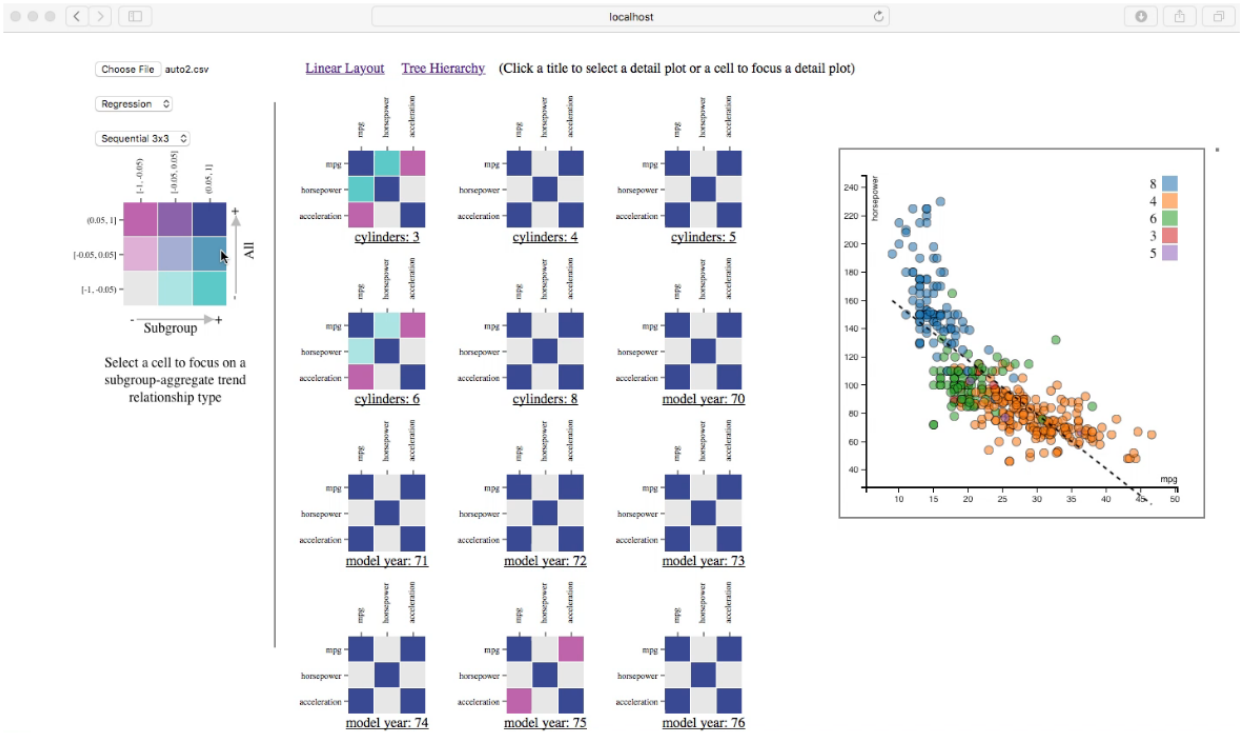


Figure 1.2: The application user interface for regression trend Simpson's paradox detection in the Auto MPG data set, using a bivariate color scheme.

the Auto MPG data set. This visualization is a precursor of our later visualization system for exploring Simpson's paradox. However, the bivariate color approach has limitations. Users suffer from the cognitive effort required to match one color to two different values. Furthermore, the differences between the two values represented by the bivariate color can be hard to perceive and compare, yet the differences are crucial for data analysis. Last but not least, the organizations of the heatmaps for rank trend and regression trend are different from each other.

An empirical solution to the limitations of the bivariate color scheme is to use direct encoding of data differences [103]. We introduce a practical set of measures to support discovering mix effects (i.e., trend strength and trend distance). After the user sets the types and roles for each variable and selects the trend type on the preparation page (Figure 1.3), the visualization page (Figure 1.4) is presented by clicking the Visualize Trends button. At this

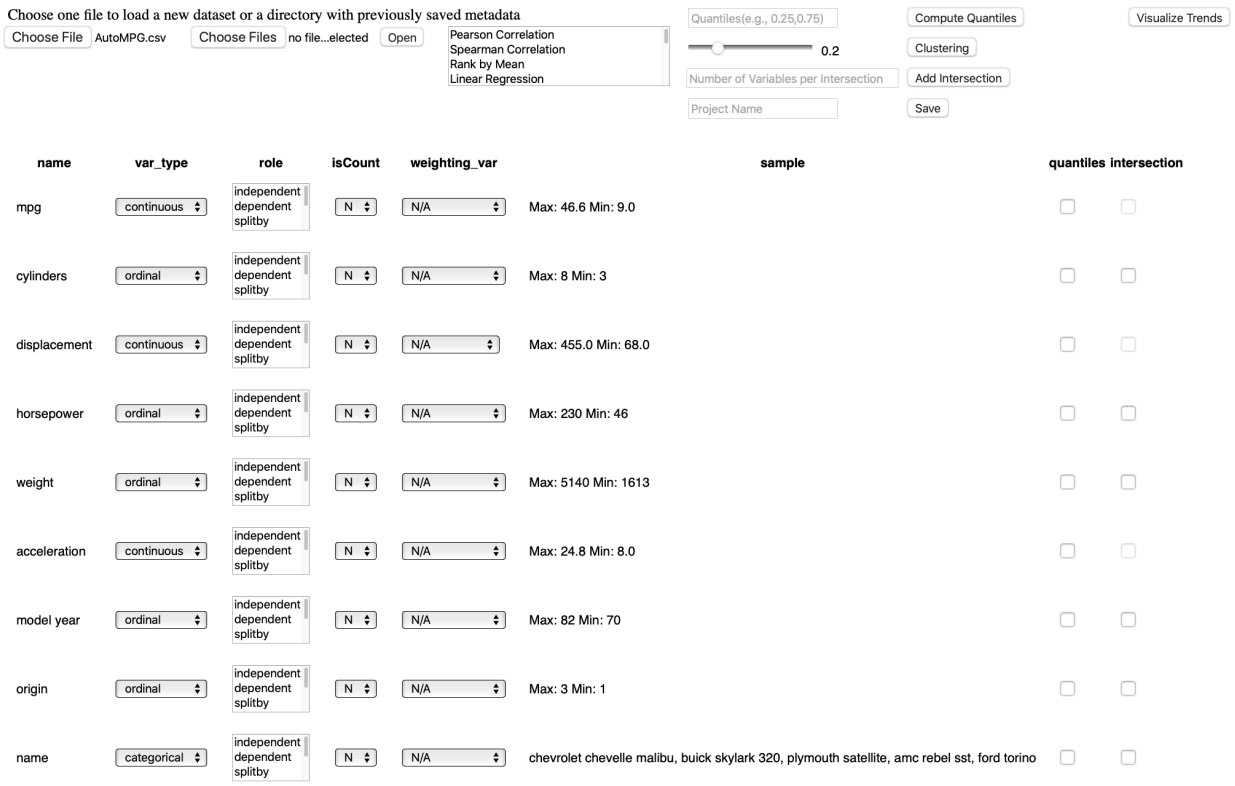


Figure 1.3: The preparation page for the Auto MPG data set in Wiggum. The user selects the types and roles for data attributes. The type and role associated with each variable are used together in rank and regression trend calculations.

point, the heatmap in the overview uses color to represent the trend distance, which measures the difference between two trends. We name our visual analysis system *Wiggum*. (Wiggum is a fictional character from the animated television series The Simpsons; he is the police chief of Springfield, and as a detective, he is a kind of analyst.) Wiggum makes improvements to the earlier tool components that we studied and developed. The improvements include a set of trend measurements that help to compare subgroup trends, a compatible overview design of subgroup trends for different trend types, and interactive features to facilitate exploration.

To assess the utility and usability of Wiggum, we conduct a user study. We also learn the limitations of Wiggum from the user study. A major limitation is that the structure of the information in the data becomes hard to analyze visually due to a combinatorial explosion of data dimensions. The combinatorial explosion occurs because the number of

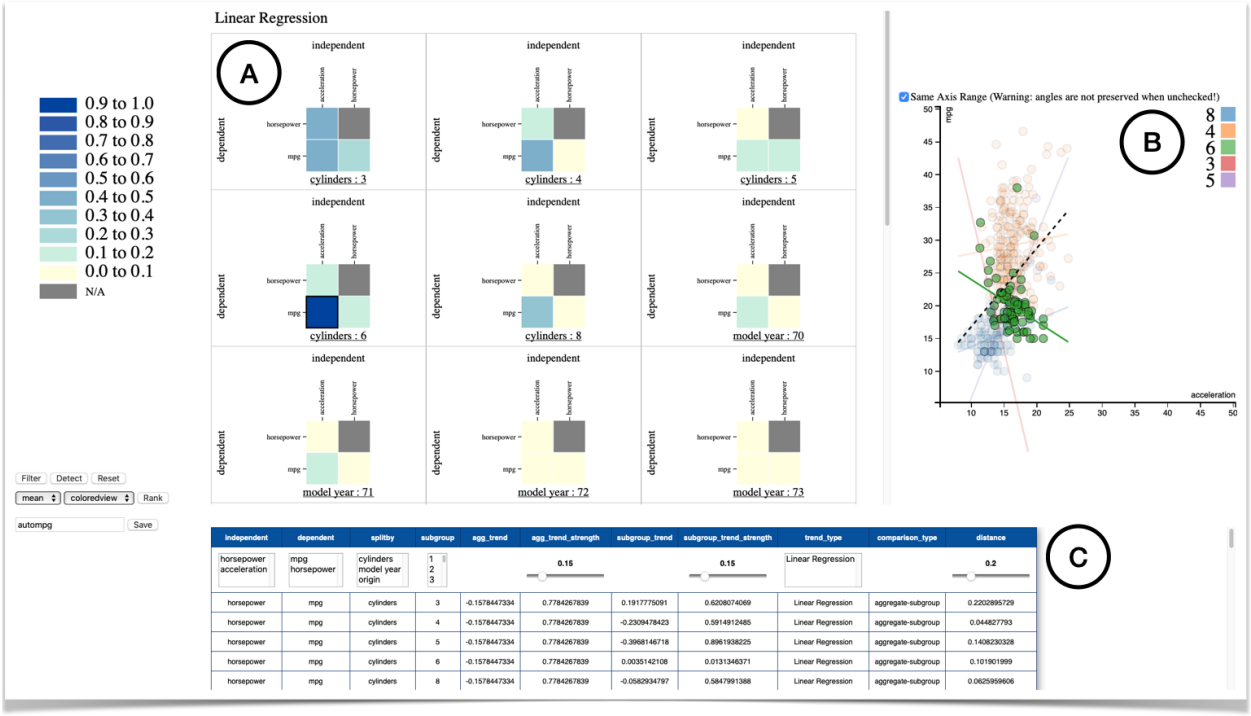


Figure 1.4: The visualization page for the Auto MPG data set in Wiggum. (A) The heatmap view provides an overview of the subgroup trends in terms of trend measurement. (B) The scatterplot shows the detail of a selected instance. (C) Users can also check the detailed statistics in the table view.

possible combinations of data dimensions rapidly increases with the number of dimensions. As a result, it is difficult to build visualizations that support identifying relationships in the data. During our investigation of the issue with dimensional combinations from the user study, we realize that the variables conveying hierarchical information (aggregate vs. subgroups) are crucial for users to perform tasks correctly and comprehend relationships among variables. This leads us to design a hybrid tree visualization, *PatternTree*, which conveys hierarchical information and explicitly shows the variables which partition data into subgroups. We apply PatternTree to the presidential election data set for gerrymandering analysis. Figure 1.5 shows the PatternTree for the 2016 and 2020 presidential elections in Oklahoma. Building PatternTree allows us to realize that there are clear issues of being able to determine the identity of graphical items in two places as we browse the hybrid tree visualization. Visualization tasks, such as identification and comparison, may require

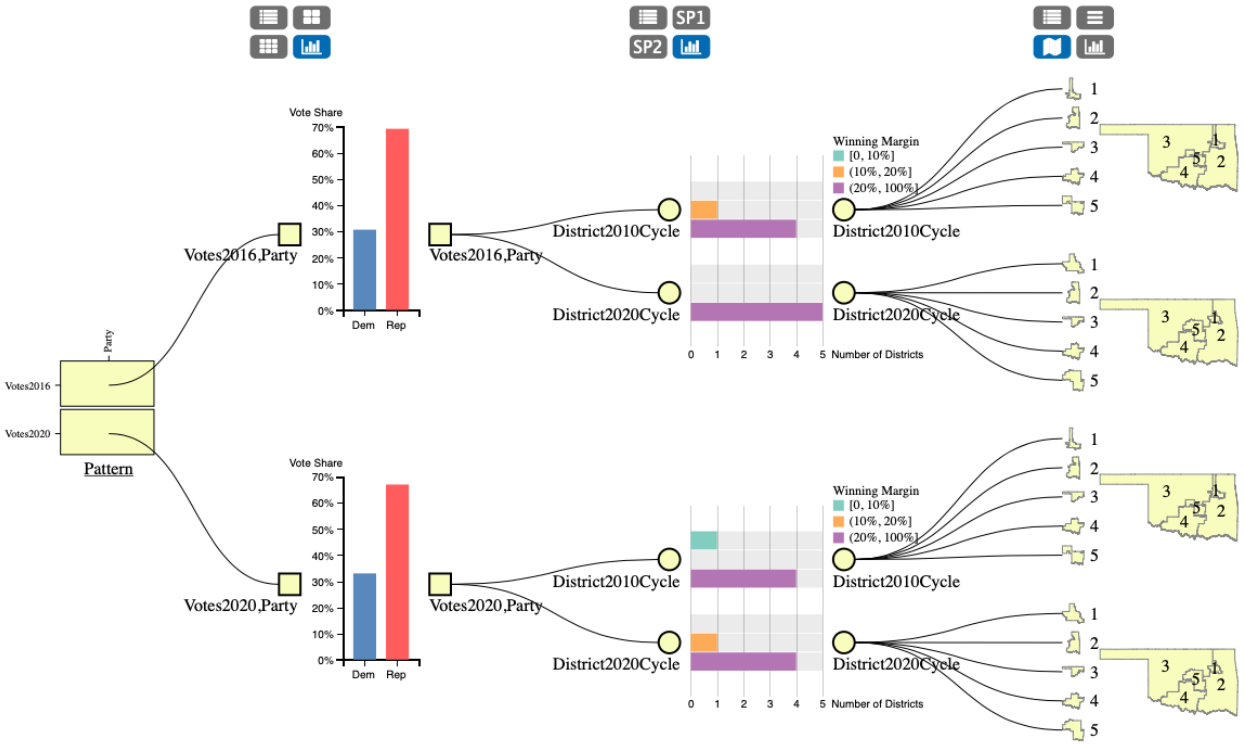


Figure 1.5: Example of a PatternTree for the 2016 and 2020 presidential election in Oklahoma. The Republican party has a higher voting share than the Democratic party in the entire data set. The trend in each congressional district is the same, but the number of highly competitive districts decreases to 0 in the 2020 presidential election data.

users to determine the identity of graphical items by interpreting the relationships between their respective visual encodings (e.g., color hue, shape, size). Consequently, we develop a model of *visual correspondence* to help associate graphical items which encode the same data through pairwise visual channels (e.g., color hue to color hue, shape to shape) in order to connect different views to each other. We apply our visual correspondence model to refine the design of PatternTree for revealing the relationship of graphical items between the host view (node-link tree diagram) and the embedded views (e.g., maps and plots). We provide an analysis of the utility of the visual correspondence theory through examples of design cases utilizing the PatternTree technique.

1.4 Thesis Statement and Research Contributions

This dissertation presents the following thesis: *Combining composite visualizations for drill down exploration with interactions into a visualization system supports effective comprehension and analysis of statistical anomalies in multidimensional data for the user with basic statistical knowledge. Adding capabilities to drill down into dimensional combinations hierarchically allows the user to see correspondences between dimensional information visually. These capabilities allow the user to more readily discover statistical anomalies and explore their relationships in multidimensional data.*

This dissertation describes five contributions to the field of visual analytics in general and to the analysis of statistical anomalies in multidimensional data specifically. The first contribution is an algorithm to detect Simpson's paradox for the regression trend case. A performance evaluation of the algorithm reveals important factors that influence the run time of the algorithm: the total number of attributes, the total number of records, and the number of levels for each categorical attribute.

The second contribution is a visual technique in which instances of Simpson's paradox in both rank trends and regression trends can be emphasized and examined via a bivariate color scheme.

The third contribution is *Wiggum*, an interactive visual analysis system that facilitates the discovery and interpretation of mix effects involving multiple trend types in multidimensional data. An evaluation of Wiggum's usability and utility via a user study provides insights into users' ability to comprehend statistical concepts, identify the corresponding visual patterns, and perform common analysis tasks correctly and efficiently.

The fourth contribution is a conceptual framework, *Visual Correspondence*, in which graphical items encoding the same data in different parts of a visualization are associated through their respective visual channels. A set of guidelines is provided to account for visual correspondence in the design and evaluation of visualizations that can show and relate data in new ways.

The fifth contribution is a hybrid tree visualization technique, *PatternTree*, in which the data items are smoothly transitioned from a host visualization to embedded visualizations for exploring hierarchical patterns in a composite visualization with nested views. An example visualization demonstrates its utility for analyzing gerrymandering in election data. We validate the theory of visual correspondence by assessing the utility of a PatternTree example application for exploring and analyzing that data.

1.5 Research Questions and Scope

The research presented in this dissertation must answer several questions raised by the thesis.

First, Simpson's paradox is generally discussed in terms of binary variables, but studies for the exploration of it for continuous variables are relatively rare. How can one detect Simpson's paradox for the trend of a pair of continuous variables in multi-dimensional data? What is the running time at different scales of data volume and dimensionality?

Second, how can statistics and visualization be integrated to find anomalies? How can visualization tools support interactive exploration to help users discover mix effects in high dimensional data? How can data be visually represented to reveal different trend types for mix effects?

Third, how can one support identification, comparison, and navigation over disparate visual features that represent the same or equivalent information? Which pairings of visual channels are perceptually effective for conveying such correspondence of information between visual features? What perceptual factors influence such effectiveness, and how? What are useful guidelines for creating visual correspondence?

Fourth, Wiggum focuses on the design of an overview-to-detail interface. As such, it is less concerned with hierarchical relationships between data dimensions. A nested tree visualization can help to represent such hierarchical information and still supporting an overview-plus-detail design. How can visualization design support a smooth transition from

host visualization to embedded visualizations in a hybrid node-link tree visualization? What is the design space for building visualizations supporting smooth transitions in the hybrid tree visualization? What are the challenges and limitations in designing the node-link layout tree visualization?

Fifth, gerrymandering is related to Simpson's paradox or mix effects. How can the computation of mix effects be applied to study gerrymandering? How can a hybrid tree visualization be designed for gerrymandering analysis?

Additional research topics are relevant to visual statistical anomaly detection, but are beyond the scope of this thesis. Among these are causality, uncertainty visualization, 3D graphics, progressive visualization, visual aggregation, and animations. They are not discussed in detail.

1.6 Organization of the Dissertation

The remaining chapters in the dissertation are organized as follows.

Chapter 2 provides an overview of previous studies in mix effects and Simpson's paradox, visual exploration of mix effects, fairness forensics tools, gerrymandering, and visualization in particular, including visual perception, visual comparison, visual linking, multiple views, design spaces of visualization, and visualization of hierarchies.

Chapter 3 presents a Simpson's paradox detection algorithm based on the correlation of two continuous variables. It discusses several empirical studies for revealing different aspects of the algorithm.

Chapter 4 describes a visual technique to detect Simpson's paradox for both rank trends and regression trends, using a bivariate color scheme.

Chapter 5 introduces the Wiggum visual analysis system, describes its corresponding processing pipeline, introduces a practical set of measures to support discovering mix effects,

and proposes new view designs for understanding and efficiently detecting mix effects.

Chapter 6 presents use cases and user studies for evaluating Wiggum's utility and usability.

Chapter 7 describes the conceptual framework of visual correspondence, applies the framework to characterize correspondences between pairs of visual encoding channels, provides foundational mathematics for analyzing visual analysis tasks in terms of interactive identification and comparison steps, and proposes a set of guidelines for the design and evaluation of visualizations that account for visual correspondence.

Chapter 8 describes the concept of virtual layering, discusses a design space for virtual layering inside node-link tree visualizations, describes a hierarchical pattern structure for statistical analysis, explores possible visual designs of a PatternTree technique for visualizing patterns in hierarchical data structures, and assesses how visual correspondence in PatternTree applications supports such visualization tasks.

Chapter 9 concludes with a discussion of the benefits and limitations of visual analytics for statistical anomaly discovery, an outline of possible extensions and future work, and a summary of contributions.

Chapter 2

Background and Related Work

This research focuses on combining statistical analysis and visualization for discovering and examining anomalies in multidimensional data. In this chapter, we review the research most relevant to mix effects and Simpson's paradox and existing visualization techniques for exploring them. We also review existing fairness forensics tools for investigating possible bias in algorithmic systems. For purposes of visualization design for analyzing such statistical anomalies, we survey the existing literature in visualization including visual perception, visual comparison, visual linking, multiple views, design spaces of visualization, and visualization of hierarchies.

2.1 Mix Effects and Simpson's Paradox

Undetected cases of mix effects can lead an unaware analyst to draw incorrect conclusions about data. A central objective of mix effects detection is to search for a diversity of trends between aggregate data and partitions of that data, and account for that diversity in one's analysis. The phenomenon has many names, but this most popular one is attributed to Blyth, crediting one of the earliest formal accounts of the paradox [20]. Two popular examples are gender disparities in university admission rates that reverse at the department level (the Berkeley Admissions Example [18]), and average income increasing over time but decreasing

at every level of education [102].

In the Berkeley Admission Example, the relative rank of the genders by admission rate is flipped, and we refer to this form of mix effects as a *rank trend* mix effects. Other examples of this type include tuberculosis mortality rates by race [55] and the on-time departure rate of flights per airline and airport [122]. In addition, mix effects can be found in real-world sports data sets [154, 111] used to rank players on statistical measures.

In the average income example [102], the summary statistic is the correlation between time and income conditioned on educational attainment. By considering the direction of the correlation, this form relies on assuming a linear relationship between the variables, and is referred to as a *regression trend* mix effects. Other examples of this form include the positive relationship between coffee and neuroticism by gender [80], and petal width and sepal of irises by species [164].

As a special example of mix effects, Simpson's paradox is of interest and often of significant concern in many disciplines including epidemiology [71, 141], social science [7, 8, 87], psychology [80], and sports analytics [45, 154]. In one famous example, researchers quantified the success rate of a surgical treatment [30]. Error arising from the paradox was not discovered until years later [77]. Simpson's paradox has also been studied in its impact on association rules in databases [57]. In the statistics community, it is a well-known phenomenon, widely understood through causal explanations [11, 109]. The *HypDB* [121] system was developed to detect, explain, and resolve bias leading to statistical anomalies like Simpson's paradox. The *HypDB* system is designed to help users detect bias using a causal directed acyclic graph (DAG). It can be difficult to create and interpret a causal DAG, calling for additional work to develop techniques and tools for analysis of Simpson's paradox and other kinds of mix effects.

2.2 Visualization of Mix Effects

In visualization research, mix effects' impact on causality has been studied with respect to the reliability of correlation analysis [149, 150]. In causal inference, a confounding variable that predicts both treatment and outcome can be closely linked to mix effects [108]. Existing visualization techniques mostly fall into two categories based on how trends, including rank trends and regression trends, manifest in data. Tabular techniques usually focus on rank trend mix effects [14, 18, 45, 107, 53]. Scatterplots are more appropriate when the purpose of the visualization is to present regression trends of two variables [7, 8, 80, 118, 31]. A line graph that consists of only two points on the x-axis has also been proposed to illustrate mix effects visually [118]. Causal network visualizations [166, 162, 76, 34, 51] offer additional means to explore and interpret mix effects.

Efforts to visualize mix effects demonstrate how detection can be included in visual exploration environments [65]. *Vizdom* allows users to observe instances of mix effects by using multiple bar charts to compare aggregate and partition trends [43]. A grouped bar chart can be used to explain mix effects by observing trend reversal in each subgroup [122]. The *comet chart* [13] supports the detection of mix effects, approaching the problem with an explicit goal to explain the phenomenon mathematically. However, it does not address effectiveness for exploring multidimensional data sets, is inherently unsuited to exploring more than two subgroups—such as race in the adult income example—and doesn't support regression trend mix effects.

Building on static visualization [13, 80], interactive approaches hold clear promise to facilitate the mix effects detection process [65]. While automated methods for detection can facilitate discovery [164], their black-box nature may not be suitable for sense-making. The relatively new area of auditing fairness in machine learning [28, 159, 5, 151] brings visual analytics techniques into the domain of interpretable machine learning [91, 119], concerned with understanding the results of ML-based decisions. The work in this dissertation applies interactive visualization techniques to create a human-in-the-loop design that improves in-

terpretability as a part of sense-making by giving analysts access to and control over the detection process.

2.3 Fairness Forensics Tools

FairVis is an interactive visualization tool that emphasizes discovering disparities with respect to both user supplied and discovered subgroups [28]. The target user for FairVis is the model developer who needs to evaluate models for biases in their outputs. To extend this tool to work with a user supplied data set, a model must be trained outside of the tool and the results must be appended as the last column, then the tool's JavaScript code must be edited to add the new data set to the start menu of the application. A deployed demo is available, but only to be used on pre-provided data sets. FairVis focuses on auditing intersectional bias by examining unfairness in a population defined by multiple features. FairVis allows users to generate subgroups in several ways, including combining different attributes or values from different attributes. In addition, suggested subgroups and similar subgroups are generated to help users discover potential biases. However, FairVis can only assess fairness for binary classification. The user can choose from 10 common fairness metrics. An interactive visual interface allows users to explore, compare, and investigate multiple subgroups.

Aequitas is an open support bias report generator designed explicitly for use by both data scientists and policy makers [120]. It is focused on the evaluation of risk assessment tools and generates a report after a model exists, but possibly prior to deployment or in the model selection phase. Aequitas is implemented as a Python library and a web application that visualizes group metrics derived from multiple fairness definitions and disparities between groups. After uploading data, the user selects protected attributes that need to be audited for bias, and specifies reference groups. Next, a "fairness tree" guides the user to select fairness metrics. Meanwhile, the user can enter a disparity intolerance value to filter out the detected results within a range of disparity values that can be considered fair. A bias

report is generated for use in decision making. Aequitas enables users to visualize group metrics and disparities between groups. It implements multiple fairness metrics, is designed explicitly to be accessible to non-technical parties, and supports reproducibility by offering relatively few settings for the user to determine on each use.

IBM's *AI Fairness 360* (*AIF360*) toolkit is an open source Python package that implements fairness metrics and algorithmic interventions [15]. The primary audience for AIF360 is developers who are building AI systems; it is primarily a programming API and is focused on the implementations of the interventions that are included in understanding and diagnosing bias. The tool is open source and provides clear contributing guidelines including the criteria on which contributions will be evaluated for inclusion. That it is developed by a major corporation is an advantage in that there are paid maintainers and effort behind growing awareness of the tool. The package implements both groupwise and individual notions of fairness to inspect various biases in many forms, thus meeting the criterion for definition agnostic analysis. The package is published with demos and tutorials, so while it is catered toward those who are building, and enables evaluation of any ML model, it is relatively accessible. It serves as a useful tool for teaching these concepts in programming as it has a standardized API with the popular machine learning package *scikit-learn*. Its interactive web application is designed to help users understand the stages of the fairness pipeline, and interactive demos are viewable and runnable on cloud tools without writing code or setting up a programming environment, increasing accessibility further. As a fairness forensics tool, AIF360 is a bias measurement tool, since it provides implementations of a number of metrics. A report comparing original biased metrics and mitigated results is generated for assessing the effectiveness of the chosen fairness metrics and the chosen mitigation algorithm.

FairTest employs a novel principled methodology for discovering discriminatory or otherwise unwanted behaviors in machine learning systems called the *unwarranted associations framework* [139]. FairTest is both a Python application and a library that provides a Python API on top of the R programming language. Multiple metrics for measuring association are

available and classified into three categories: frequency distribution metrics, correlation, and regression. In order to discover association bugs in sub-populations, FairTest generates an association-guided decision tree that finds sub-populations with strong associations between the protected variable and the application output. After finding strongly affected sub-populations, FairTest validates the association bugs on an independent test data set by performing hypothesis testing. In addition, FairTest ranks the sub-populations on confidence intervals of the association metric. An association bug report is generated for further investigation. It also audits application outputs along with the user's input data set.

Silva is a tool implemented as an interactive web application that visualizes a causal graph with bias metrics to help users assess machine learning fairness [166]. Investigations through Silva occur after a model has been built. The tool integrates three machine learning models and five fairness metrics to study and compare fairness during the model validation stage. Supportive approaches, such as recommendations on sensitive attributes' nodes and paths in the causal graph, are provided to guide the exploration of fairness. Informative tips in the application assist users from different backgrounds in effectively navigating the application with the ultimate goal of assessing fairness. The interactive visualization facilitates the exploration of connections between causality and fairness metrics. Moreover, the causal graph helps users inspect the relationship among attributes having different roles in machine learning fairness. Silva can be utilized in concert with existing fairness tools and machine learning pipelines to mitigate biases.

Google's *What-If Tool* is an open source interactive visual interface for gaining a better understanding of machine learning models [159]. Users can run minimal code from many platforms, making it accessible to a wide variety of programmers. It implements multiple performance metrics disaggregated by both supplied and computed groups, allowing a user to simultaneously consider multiple notions of fairness. The tool can benefit a wide variety of users from novice machine learning users to experienced machine learning researchers. It consists of three interfaces with features to compare two models applied to the same data

set. The first interface allows users to inspect and visualize individual data points with their model predictions. It can also offer the nearest counterfactual example for users to compare its similarity to the selected data point. Users can edit a selected data point, and rerun the model to visualize the change in the new inference result. The second interface shows two performance measurements. Users can slice the data, and apply fairness optimization strategies to inspect how they affect positive classification thresholds across data slices. It also supports the exploration of intersectional subgroups. In the final interface, users can visualize each feature's distribution and its summary statistics.

FairSight is a visual analysis tool for promoting fairness in decision making, especially in ranking decisions [5]. It is based on the FairDM framework, a decision making framework that combines three machine learning phases—data, model, and outcome—in a fairness pipeline. The fairness pipeline consists of four required actions: understand, measure, identify, and mitigate. For understanding bias, FairSight introduces input space, output space, and mapping. FairSight measures distortion to enable assessment of fairness by comparing pairwise distance in the input space with pairwise distance in the output space. Two different distance metrics are used for the input and output spaces, respectively. FairSight includes a comprehensive set of fairness metrics to support individual and group fairness assessments. It provides various strategies to investigate sensitive features (e.g., gender or race) that can potentially identify bias. It also includes methods to mitigate bias in pre-processing, in-processing, and post-processing stages. Users can select features to improve fairness in pre-processing. In-processing supports fair machine learning models, such as the Additive Counterfactually Fair model [84]. Post-processing applies a re-ranking algorithm [168] for any given ranking outcome to achieve fairer decision making.

Fairlearn is a Python toolkit for fairness assessment and unfairness mitigation in machine learning models, such as classification and regression models [19]. For different machine learning tasks, users can choose different metrics. Fairlearn focuses on assessing group fairness in which groups of individuals are defined using sensitive features (e.g., gender, age, or disabil-

ity status). Users can specify sensitive features to compute fairness metrics for a population and its subgroups. Fairlearn enables users to generate an interactive dashboard for visualizing fairness metrics and model comparison. It applies mitigation algorithms during training and post-processing. During training, a mitigation algorithm adjusts the machine learning model to fit fairness constraints. During postprocessing, a mitigation algorithm transforms the output to achieve fairness.

FAT Forensics is an open source Python package for inspecting fairness, accountability, and transparency in the machine learning pipeline [132]. It provides a basic visualization module and implements various fairness measurements for use in different stages of the pipeline. It provides both a "research" mode and a "deployment" mode. The research mode is designed for FAT (Fairness, Accountability, and Transparency) researchers who use it to create new fairness metrics and compare them with existing metrics. The deployment mode is meant for data scientists and machine learning model developers who need to evaluate a model or assess fairness in a data set. The package includes tutorials, examples, user guides, and an API reference.

Wiggum is an interactive visual analysis system for uncovering both mix effects and special cases known as Simpson's paradox. It is implemented as a Python library and a web application. Users provide metadata such as variable types and roles. Wiggum computes a set of measures in terms of trends to support discovering Simpson's paradox and presents them in an overview plus detail visualization. Interactive filtering and ranking features allow users to efficiently organize trends for exploratory analyses. Wiggum is accessible to a wide variety of users. It does not require any programming skills to use the web application. The Python library includes all of the computational features and runs in the back-end server of the Wiggum application. It has been designed as a modular framework, allowing for each individual component to be modified or extended.

2.4 Visualization

This section first covers visual perception and visual comparison. We also discuss visual linking and multiple views. Since our work investigates the design space for connecting views in a composite visualization of data that includes hierarchical information, we review prior work with a focus on design spaces of visualization and visualization of hierarchies.

2.4.1 Visual Perception

Much of the foundational work in visualization focuses on understanding the relationship between visual encoding channels and graphical perception. Bertin [17] considers firstly the ordering of visual variables associated with four levels of organization: associate, selective, ordered, and quantitative. Cleveland and McGill [37] identify and order the visual encoding channels for representing quantitative information in a variety of chart types. Numerous perceptual experiments in cognitive psychology research and otherwise validate elements of these foundations and suggest a refinement of understanding of visual channels. Mackinlay [94] proposed rankings of visual channels for three different types of information—quantitative, ordinal, nominal—which later developed through a series of visualization systems culminating in the Tableau tools [137, 95].

Heer and Bostock [68] demonstrate the viability of conducting graphical perception experiments on a crowdsourced platform. They analyze the cost and performance of such experiments. Demiralp, et al. [47] estimate perceptual distances for building perceptual distance matrices (perceptual kernels) through crowdsourced experiments. The perceptual distance measures the perceptual similarity between a pair of elements in a single visual channel (univariate perceptual kernel) or in a pairwise combination of visual channels (bivariate perceptual kernel), such as color, shape, and size. The perceptual kernels can be integrated into automated visualization design for optimizing perceptual discriminability. McColeman et al. [98] challenge assumptions in the existing rankings of visual channels. They find that

prominent factors, such as the number of marks, have an impact on task performance and could change the ranks of visual channels. Kim and Heer [81] suggest interactions between visual encoding channels, task types, and data characteristics as factors to consider in determining the effectiveness rankings of visual encoding channels. They measure the impact of task (summary task, value task) and data distribution characteristics (e.g., cardinality, entropy) on encoding effectiveness rankings. Davis, et al. [44] investigate individual differences in graphical perception performance. Their experimental results reveal patterns deviating from the canonical rankings of visualization effectiveness.

Some research focuses on particular techniques and tasks in studying the effectiveness of specific visual encodings. Heer, et al. [69] study the effect of space-efficient techniques for increasing data density through graphical perception experiments of time series charts. Based on their experimental findings, they propose design implications with regard to chart size and layered bands for value comparison tasks in time series visualizations. Nothelfer and Franconeri [103] investigate the visual perception of relations among data value pairs across different comparison tasks. Data values are encoded in pairwise combinations following a set of data encoding approaches (individual value encoding, delta value encoding) applied to particular visual encoding channels (slope, length, position). Chung, et al. [35] present crowdsourcing empirical studies which investigate the perception of orderedness with different visual channels. Their results indicate that shape can be orderable by the number of spikes or edges, in contrast with Bertin [17] and Mackinlay [94], neither of which relates shape to ordered perception or ordinal information.

The need for effective utilization of visual encoding channels in visualization designs calls for ongoing study of human perceptual capabilities. We argue that the state of the art in visualization design has not yet teased apart understanding of visual channels sufficiently. In particular, there is very little formal understanding of how visual channels combine in perceptually effective ways. Our model of visual correspondence considers visual encoding channels pairwise across views to model their effectiveness for showing the same information

in multiple places in visualization designs.

2.4.2 Visual Comparison

The study of supporting comparison visually has attracted considerable attention in the visualization community. Gleicher, et al. [61] survey works in information visualization related to comparison, and categorized visual designs for comparison into three types. A top-down perspective raises four considerations in conceptualizing comparison [60]: comparative elements, comparative challenges, scalability strategies, and comparative designs. Wu proposes a view composition algebra (VCA) that supports ad hoc visual comparisons [161]. VCA focuses on comparison safety and data transformations from the viewpoint of data itself. While these efforts provide invaluable insights for visual comparison at the level of entire views of data, they are less concerned with comparison at the level of visual perception, including the effectiveness of visual encoding channels and the visual correspondence between compared marks/glyphs. We argue that a better understanding of visual perception at the level of visual encoding channels across views is crucial for the support of comparison tasks in visualization designs.

Ondov, et al. [104] present an empirical study to explore the perceptual factors in visual comparison. They focus on comparative layouts of views [61, 93] and a narrow set of visual channels (length, slope, angle) represented by different charts for specific tasks. Jardine, et al. [74] extend empirical evaluation with two new comparison tasks to understand which visual features are adopted. They propose perceptual proxies for the interactions among tasks, marks, and spatial arrangements in the visual comparison of means and ranges. Xiong, et al. [163] empirically evaluate comparison affordances of four spatial arrangements for bar charts. They find that viewers tend to compare bars that are visually aligned and spatially proximate. Their work is confined to bar charts, and narrows the scope of factors that could affect the perception of visual grouping and visual comparison. These studies focus on comparing patterns of a target set of items. In contrast, our work is target-agnostic, and

focuses on understanding the relationship between corresponding objects as represented in multiple ways.

2.4.3 Visual Linking

Visual linking is a way to connect objects in visualization. It explicitly indicates the correspondence between connected objects. VisLink [38] places multiple 2D layouts in 3D space and draws inter-plane edges to reveal direct or indirect relationships between the visualizations. Flexible Linked Axes (FLINA) [36] draws links between pairs of axes to indicate relationships between pairs of attributes. ConnectedCharts [148] explicitly draws curves either between corresponding tuples or between corresponding axes to display relationships between multiple charts.

Steinberger, et al. [134] present context-preserving visual links that aim to minimize the occlusion of important information and reduce visual interference. LineUp [63] uses lines to bridge multiple alternative rankings on the same set of items for easier tracking of changes. More closely related to our work is Domino [62], a multiform visualization technique for showing subsets and the relationships between them. Domino represents subsets as blocks categorized into three types: partitioned, numerical, and matrix. It visually links visualizations to show relationships at multiple levels of detail: items, groups, and entire data sets. Connecting lines and alignment indicate corresponding items in adjacent blocks.

Visual linking is not always suitable to connect related graphic items, such as in visualizations in tabs [137, 23] or multiple views on large displays [86]. Our theory of visual correspondence describes how one can perceive associated graphic items through a pair of visual encoding channels, based on Gestalt principles of design including alignment, proximity, and similarity, regardless of whether explicit graphical associations like visual linking are suitable for use in multiple view visualization designs.

2.4.4 Multiple Views

Multiple views are commonly used to visually represent multi-dimensional data in information visualization. Baldonado, et al. [153] provide design guidelines for selecting, presenting, and interacting with multiple views, and discuss basic trade-offs for design. Javed and Elmqvist [75] present a design space for composing multiple views, and identify benefits and drawbacks for each of five patterns: juxtaposed, integrated, superimposed, overloaded, and nested views. More recent work focuses on spatial arrangement of view layouts [6, 128]. For example, Chen, et al. [33] explore correlations between view types and popular layouts based on images of multiple view visualizations in publications in the visualization community.

Multiple coordinated views is a technique that allows users to interact with presented data through multiple visual representations (views) in a single composite visualization [117]. Improvise [155] presents an environment for building and browsing multiple-view visualizations [156, 157, 158] interactively through a shared-object coordination mechanism. Whereas active interactions, such as dynamic queries or brushing, show view-to-view relationships in coordinated multiple views, passive interaction like visual correspondence also can help to see such relationships through pairwise visual channel encodings. Multiple view consistency [113, 114] explores effective constraints on position and color encoding channel pairings across views. GraphScape [82] uses a directed graph model to reason about visualization sequencing. Multiple view consistency and visualization sequencing are ways of looking at visual correspondence, either very narrowly at the perceptual level or more broadly at the task level. Our work considers visual correspondence at the intermediate level of task steps that involve visual identification of sameness or equivalence.

Brehmer et al. [23] describe visual encoding design choices in terms of the matches and mismatches between abstract tasks and visual representations, with consideration of multiple view workflows. They provide interactive linked highlighting between relevant elements in juxtaposed views. Van den Elzen, et al. [145] enable users to explore a visualization's state space in terms of parameters and parameter values by offering multiple alternative views

with small multiples. Their later work [146] considers multivariate network visualization via selections and aggregation operations in two juxtaposed, coupled views. They use color to achieve visual coherence between the different components. This is one of many examples for which our theory of visual correspondence can help to explain why known approaches to coordinating multiple views work well (or not).

2.4.5 Design Space of Visualization

Visualization "design spaces" are widely discussed in the visualization community. Card and Mackinlay present a guide to the design of information visualization based on the space of possible mappings between data types and graphical contexts [29]. Design spaces are typically described in terms of their dimensions, such as the dimensions of chart types, combinations, and enhancements used to classify data visualizations in genomics [42, 41]. Spatial relations and data relations are often emphasized in designing a composite visualization [75]. Guidelines for the design of multiple view tools to support the investigation of data consider cognitive factors including the effort required for comparison, context switching, working memory, and learning [153]. Gleicher, et al. consider fundamental designs for visual comparisons [61] and propose an abstract framework to aid in designing visualizations for comparison tasks [60].

Modeling the dimensionality of visualization design spaces is more elaborate as the purpose of the designs becomes more specific. In one design space for geospatial networks, four dimensions capture key aspects of geospatial network visualization, namely geography representation, network representation, composition, and interactivity [123]. Frishman and Tal minimize visual changes in clustered graphs by introducing invisible vertices [56]. Their work uses virtual nodes to harmonize the layout of graphs, but it differs from our proposed virtual layering technique both in looking at only a single view of the graph representation, and in that it addresses only the layout of the vertices.

2.4.6 Visualization of Hierarchies

There are many known ways to visualize hierarchies [125]. Common representations include node-link diagrams and treemaps. More complex techniques for representing hierarchies include elastic hierarchies, hierarchical edge bundling, and hierarchical aggregation. Two common tree representations, node-link diagrams and treemaps [129], can be mixed to form a hybrid visualization for investigating *elastic hierarchies* representing trees [170]. Holten [73] proposed hierarchical edge bundling of adjacency edges to reduce visual clutter in tree visualizations. Elmqvist and Fekete [50] presented a model of hierarchical aggregation based on linking aggregation in data space with the visual entity of the aggregates in visual space. Our work involves a tree of dimension combinations and embeds nested views where nodes would be in a node-link diagram.

There exist many composite visualizations for hierarchical data. Both tuples and attributes in tabular data can be divided to form an aggregation hierarchy. *Breakdown visualization* supports drill-down from overview to details across hierarchies through small multiple views [39]. In geographic visualization, a phylogenetic tree visualization superimposed on a map links geographic locations through a linear geographic axis [105]. TreeVersity2 [64] presents a space-filling visualization called StemView that nests bars at each level of an icicle tree. A web-based tool, *VEHICLE* [97], embeds hierarchical stacked bar charts into a node-link diagram for exploring conflict event data, and uses a radial tree layout with glyphs as nodes to visualize hierarchical information. A node-link diagram is juxtaposed with heatmap views for visualizing sequences of transactions in information hierarchies [27]. The DimLift system, presented by Garrison, et al. [58], represents hierarchical data through dimensional bundling, and applies multiple composite and interactive visualization techniques to facilitate exploration.

Despite a large amount of work on the visualization of hierarchies, none of these solutions addresses the problem of transitioning graphical items with the same information from the host view to the embedded view. Our work introduces a new visualization design pattern

that facilitates smooth visual transitions in composite visualizations.

Chapter 3

Automatically Detecting Simpson's Paradox

3.1 Overview

This chapter presents a Simpson's paradox detection algorithm based on the correlation of two continuous variables. It discusses several empirical studies for revealing different aspects of the algorithm. Simpson's paradox is a phenomenon in which a trend in the entire population reverses within the subpopulations defined by a categorical variable. Detection of Simpson's paradox can suggest surprising and interesting phenomena in a data set to an analyst. It is generally discussed in terms of binary variables; studies involving continuous variables are relatively rare.

In this chapter, we describe an algorithm to detect Simpson's paradox in trends consisting of a pair of continuous variables. A correlation coefficient is used to measure the association between each pair of continuous variables. Categorical variables partition the entire data set into groups. The algorithm looks for sign reversals between the correlation coefficients measured within the group relative to the original entire data set. We show that the algorithm detects cases in real data sets as well as synthetic data sets, and demonstrate that the

approach can uncover hidden patterns by detecting occurrences of Simpson's paradox. We benchmark the running time of the algorithm for typical combinations of varying conditions. We also propose an approach to exploit sampled data for opportunistic Simpson's paradox detection.

3.2 Detecting Simpson's Paradox

Simpson's paradox has been studied in two main forms: through relative rates and through trends. We focus on the trend-based case and use linear correlation to measure a trend between two variables.

Given a data set, we first determine the type of each column, then assign the columns to one of two lists: (1) *splitby attributes* for conditioning over; and (2) *candidate attributes* for computing relationships. Columns of integer and non-numerical types are added to the splitby attribute list. Common types include binary (e.g., male/female), categorical (e.g., red/blue/green), and ordinal (e.g., small/medium/large). Columns of continuous-valued data type are added to the list of candidate attributes.

Algorithm 1 shows pseudocode for the algorithm. For a data set with d candidate attributes, the algorithm first computes the $d \times d$ matrix of correlation coefficients for each pair. The entry in the i-th row and j-th column of the correlation matrix represents the correlation between the i-th candidate attribute and j-th candidate attribute.

The most computationally expensive step of detecting Simpson's paradox comes next. The algorithm partitions the data set by conditioning on each splitby attribute and computes an additional $d \times d$ covariance matrix for each value of each splitby attribute. If the data set has C splitby attributes and attribute c has k_c different values, the algorithm computes $\sum_{c=1}^{C} k_c$ correlation matrices of size $d \times d$. Table 3.1 shows the structure of the two correlation matrices between two candidate attributes when conditioned on each of two values (Blue and Red) of a 'Color' splitby attribute in a synthetic data set. The entries of the correlation

		Attribute 1	*Attribute 2*
Blue	*Attribute 1*	1.000000	-0.619
	Attribute 2	-0.619	1.000000
Red	*Attribute 1*	1.000000	-0.616
	Attribute 2	-0.616	1.000000

Table 3.1: Per-group correlation matrices for a synthetic data set.

matrix are symmetric with respect to the main diagonal. Thus, the algorithm compares the signs of each element in only the upper halves of the $\sum_{c=1}^{C} k_c$ correlation matrices of the subgroups to the correlation matrix for all of the data. Instances of sign reversals are stored with the overall correlation, reversed correlation value, matrix indices of the continuous attributes, index of the splitby attribute, and the subgroup value. The instances constitute detections of trend reversals that may contribute to a case of Simpson's paradox.

Algorithm 1 Simpson's Paradox Detection Algorithm

```
INPUT: Relational Table R
continuous_column ← detectContinuousTypes(R)
splitby_column ← detectNonContinuousTypes(R)
for all (column1,column2) ∈ continuous_column do
  correlationMatrix1 ← computeCorrelation(R, column1,column2)
end for
for column ← splitby_column do
  subgroups ← R.groupby(column)
  for group ← subgroups do
    for all (column1,column2) ∈ continuous_column do
      correlationMatrix2 ← computeCorrelation(group, column1,column2)
    end for
    if isReverse(correlationMatrix1, correlationMatrix2) then
      SP_result ← subgroup_info
    end if
  end for
end for
```

3.3 Example Applications

We conducted experiments to apply the algorithm to a synthetic data set and two real data sets from the University of California, Irvine machine learning repository [90]: the popular iris data set [54] and auto miles per gallon data set [115].

3.3.1 Synthetic Data Set

As a preliminary validation of the algorithm, we generate synthetic data for a simple case of group-wise Simpson's paradox of the regression type, shown in Figure 3.1. The means and

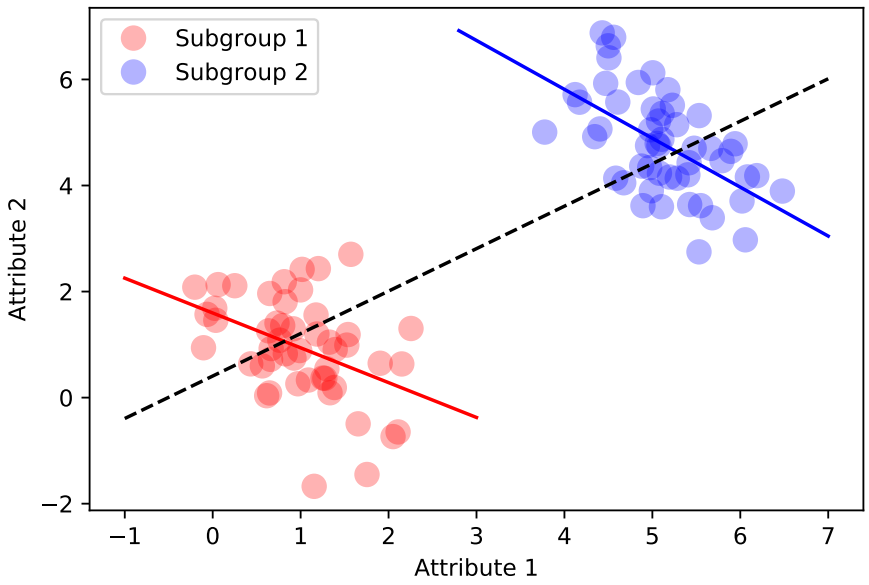


Figure 3.1: Simpsons' paradox occurring in an example data set with two subgroups. Each subgroup has a full weight trendline. The dashed line shows the reversed trend for the entire data set.

covariances are set for randomly generated samples of a multivariate normal distribution. The number of means depends on the number of subgroup clusters. A shared covariance is used for all subgroup clusters. The means and covariances of individual attributes induce instances of Simpson's paradox. We start with a pair of continuous attributes plus a categorical attribute, color. We then add a noisy continuous attribute and two noisy categorical

allCorr	attr1	attr2	revCorr	splitby	subgroup
0.771	attribute 1	attribute 2	-0.619	color	b
0.771	attribute 1	attribute 2	-0.616	color	r

Table 3.2: Result from our algorithm for the synthetic data set.

attributes. The size of the data set is 100 records. The scatter plot in Figure 3.1 visually illustrates the phenomenon. The dashed line indicates the trend for the entire data set, and the two solid lines show the reversed trends in the subgroups.

The detection algorithm finds two occurrences of trend reversals in the synthetic data set. Each row in Table 3.2 corresponds to an occurrence of a reversal. The first column shows the correlation for the entire population. The second column and third column indicate which two attributes exhibit reversals. The fourth column is the reversed correlation found in the subgroup. The categorical attribute used to partition data is in the fifth column, and the last column indicates the value of that categorical attribute. The correlation in the entire data set indicates that an increase in attribute 1 is correlated with an increase in attribute 2. Nevertheless, if one considers the red or blue subgroup of the color attribute, it is apparent that an increase in attribute 1 correlates with a decline in attribute 2, thus exhibiting Simpson's paradox in the color attribute.

3.3.2 Iris Data Set

We use our approach to detect Simpson's paradox in the Iris data set [54]. The Iris data set has 5 attributes: sepal length, sepal width, petal length, petal width, and species. The first four attributes are continuous-valued measurements of flowers. Species are in three categories: Iris Setosa, Iris Versicolour, and Iris Virginica. The number of records is 150 and there are no missing values in the data set.

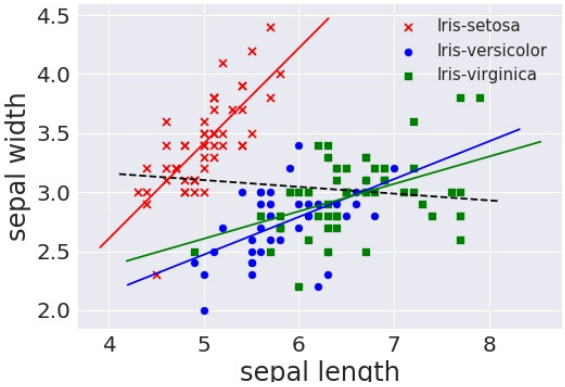
The algorithm detects nine occurrences of trend reversal, shown in Table 3.3. Three of

allCorr	attr1	attr2	revCorr	splitby	subgroup
-0.109	sepal length	sepal width	0.747	class	setosa
-0.109	sepal length	sepal width	0.526	class	versicolor
-0.109	sepal length	sepal width	0.457	class	virginica
-0.421	sepal width	petal length	0.177	class	setosa
-0.421	sepal width	petal length	0.561	class	versicolor
-0.421	sepal width	petal length	0.401	class	virginica
-0.357	sepal width	petal width	0.280	class	setosa
-0.357	sepal width	petal width	0.664	class	versicolor
-0.357	sepal width	petal width	0.538	class	virginica

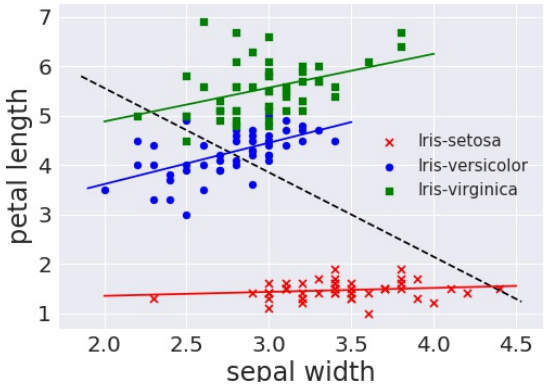
Table 3.3: The output from our algorithm for the Iris data set. Each row corresponds to a detection of a trend reversal, in which allCorr is the population correlation; attr1 and attr2 indicate the pair of continuous attributes; revCorr is the subpopulation correlation for that pair, with sign opposite to that of the overall population; splitby is the categorical attribute; and subgroup is the value for that attribute that exhibited the reversal.

the six pairs of continuous attributes exhibit reversals: sepal length vs. sepal width, sepal width vs. petal length, and sepal width vs. petal width. For example, the correlation for sepal length and sepal width in the subgroup Iris Setosa (first row) is a positive value 0.747, but the correlation for the same pair of attributes in the entire data set is a negative value -0.109. All three records are a full reversal of the trend: each of the three candidate attributes has the opposite value correlation as the overall population for those three pairs, indicating an instance of Simpson's paradox in the class attribute.

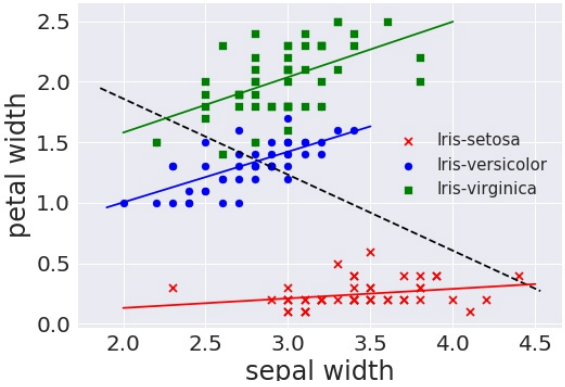
The scatter plots in Figure 3.2 show all three occurrences of Simpson's paradox. The regression lines show the trends in the data. Colors and shapes encode the subgroups. The dashed line indicates the trend for the entire Iris data set. The plots give insight into how the species attribute in Iris data set is a factor in the exhibited Simpson's paradox in the class attribute for each attribute pair.



(a) sepal length vs. sepal width



(b) sepal width vs. petal length



(c) sepal width vs. petal width

Figure 3.2: Visualization of Simpson's paradox in the Iris data set. Black lines show overall trends. Red, green, and blue show trends for individual species. Relationships between the various attributes are reversed in individual species relative to the entire sample.

allCorr	attr1	attr2	revCorr	splitby	subgroup
0.423	MPG	acceleration	-0.819	cylinders	3
0.423	MPG	acceleration	-0.341	cylinders	6
0.423	MPG	acceleration	-0.051	model year	75
0.423	MPG	acceleration	-0.051	model year	79
-0.778	MPG	horsepower	0.621	cylinders	3
-0.778	MPG	horsepower	0.013	cylinders	6

Table 3.4: The output from our algorithm for the Auto MPG data set, in the same form as Table 3.3.

3.3.3 Auto MPG Data Set

The third experimental data set is the auto MPG data set [115]. We select three attributes of continuous (MPG, acceleration, horsepower) and categorical (cylinders, model year, origin) types, and ignore the remaining attributes. We also remove six records missing a value for horsepower, leaving a total of 392 records. Table 3.4 shows the six detected occurrences of trend reversals, four for cylinders and two for model year.

It may be true that higher MPG correlates with higher acceleration and lower horsepower overall. Nonetheless, surprising patterns are found in the 3-cylinder and 6-cylinder groups. For those groups, a higher MPG correlates with lower acceleration and higher horsepower, as seen in Figure 3.3a and 3.3c, respectively. The relationships between MPG and acceleration, and between MPG and horsepower, are examples of trend reversals in particular cylinder groups. Similarly, despite a positive correlation between acceleration and MPG in the overall data, as seen in Figure 3.3b, the data for vehicles with model years 1975 and 1979 do not have a strong linear relationship.

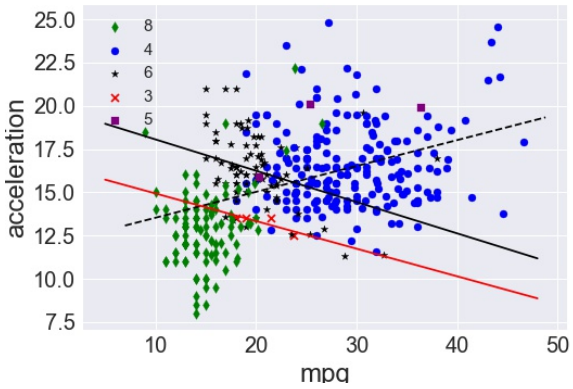
The visualized data in Figure 3.3a shows five types of cylinders. Despite the positive relationship between MPG and acceleration when the data for all types of cylinders are taken together, there exists a negative relationship for the 3-cylinder or 6-cylinder groups taken individually. The reversed trend in those two groups can be easily observed by visualization

and motivates further analysis of subgroup data. Although reverse trends in Iris data set or the synthetic data set occur in all of the subgroups, it is worthy of remark that reverse trends could occur in some of the subgroups. We refer to circumstances in which Simpson's paradox happens in some but not all subgroups as *partial Simpson's paradox*. In Figure 3.3b, a slightly negative relationship between MPG and acceleration exists for years 1975 and 1979, but there exists a positive trend for the entire data set. Similarly, in Figure 3.3c, it is evident that higher MPG tends to accompany lower horsepower overall. However, the regression lines for the 3-cylinder and 6-cylinder groups don't share the same direction with the regression line for the entire data set; the 3-cylinder group becomes more positive than the 6-cylinder group as MPG increases. This is an example of how the strengths of reversals can vary between different subgroups.

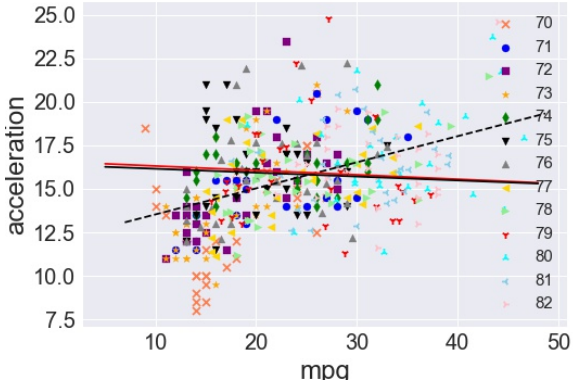
3.4 Performance Evaluation

We evaluated the performance of the algorithm on a 2015 MacBook Pro with a 2.7 GHz Intel Core i5 processor and 8 GB 1867 MHz DDR3 memory. The algorithm is implemented in Python running in Jupyter Notebook.

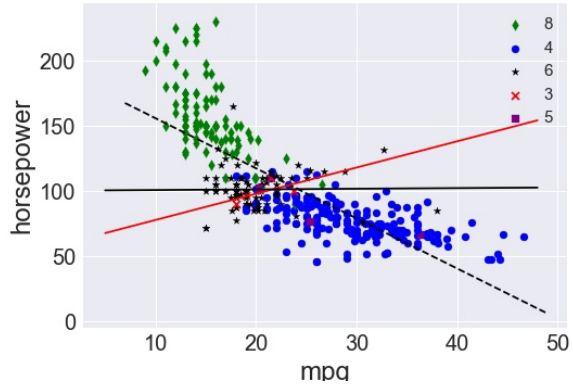
For the performance evaluation, we apply the algorithm to two continuous attributes and a categorical attribute which together exhibit Simpson's paradox. The number of subgroup clusters corresponds to the number of unique values of the categorical attribute. In Table 3.5, 32 clusters mean that there are 32 subgroups partitioned by the categorical attribute. We generate the same number of additional continuous attributes and categorical attributes, for a total of 96 attributes. For example, the 10 attributes shown in Table 3.5 represent a combination of 5 continuous attributes and 5 categorical attributes. The additional continuous attributes' values are generated by drawing random samples from a normal distribution with a predefined mean and standard deviation. Meanwhile, the additional categorical attributes' values are generated by a uniform random sampling of integers within a defined range. We



(a) MPG vs. acceleration in the cylinders group.



(b) MPG vs. acceleration in the model year group.



(c) MPG vs. horsepower in the cylinders group.

Figure 3.3: Simpson's paradox in the Auto MPG data set.

Data Size	#Clusters	10 attr.	20 attr.	30 attr.
100K	32	4.383	11.499	28.723
	256	5.144	14.512	33.954
	1024	10.797	24.270	54.154
500K	32	5.544	16.033	38.259
	256	6.815	18.703	44.084
	1024	12.272	29.723	63.196
1M	32	6.855	22.303	52.011
	256	8.165	23.965	55.423
	1024	13.811	34.985	76.289

Table 3.5: Running time (in seconds) of the detection algorithm.

generate three such synthetic data sets for each test case and calculate the average running time.

Analysis of the results (Table 3.5) reveals three important factors that influence the running time of the algorithm: the total number of continuous and categorical attributes, the total number of records, and the number of unique values of each categorical attribute.

The results for 1024 clusters and 30 attributes (15 continuous and 15 categorical) are shown in Figure 3.4. For data sets of all three sizes, the running time increases as the percentage of the sample data (Section 3.6) increases. The slope is larger for the bigger data sets. The result indicates that time efficiency benefits from our approach to use sampled data to detect Simpson's paradox in large data sets.

3.5 Discussion

Our study of the real Iris and Auto MPG data sets suggests that partial Simpson's paradox could be commonplace. Moreover, the magnitude of the detected occurrences of Simpson's paradox can differ significantly. For example, there can be a strong linear relationship between two continuous attributes in an entire data set, but an inverse relationship at the subgroup level that is not strong. One example is shown in Table 3.4 and Figure 3.3c; for

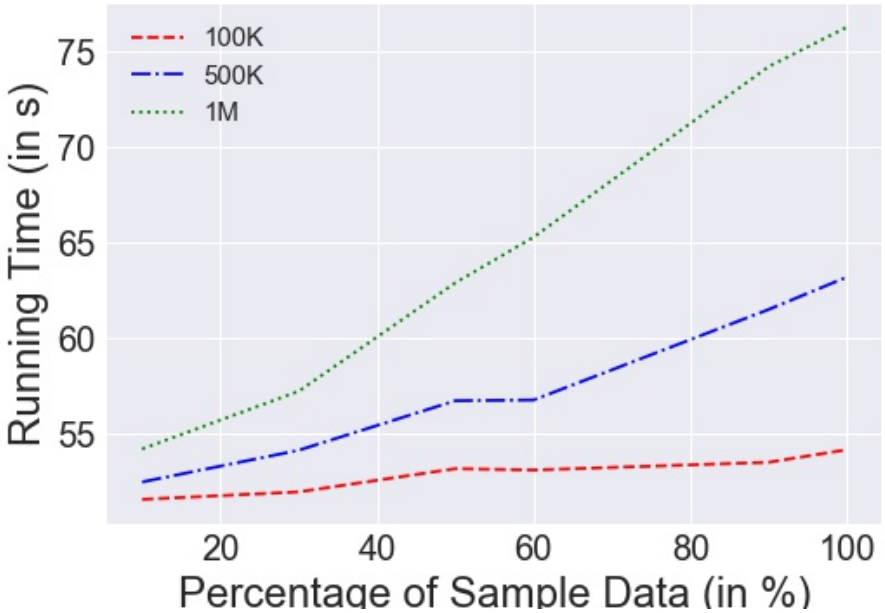


Figure 3.4: Running time of the algorithm for different samplings of the synthetic data set.

6 cylinders, the correlation value (slope of the regression line) is close to 0. The degree of the reversal should be carefully considered when analyzing large data sets. There can be many distractions arising from such minor cases of reversal. Fabris and Freitas discuss ways to rank such occurrences of Simpson's paradox [53]. A comprehensive approach to filter out low significance instances of Simpson's paradox is needed to facilitate the exploration of the meaningful occurrences of Simpson's paradox in large data sets.

Visualization techniques have been developed to assist in the discovery and analysis of Simpson's paradox. Researchers often use single tables to examine and explain Simpson's paradox [20, 18]. Scatter plots and line charts can be used to illustrate the phenomenon as well [118, 80]. More recently, the comet chart was introduced for this purpose [13]. For data that is significantly larger in either the number of records or the number of attributes, new visual techniques are needed to facilitate wider ranging exploration and analysis of Simpson's paradox.

3.6 Simpson's Paradox in Partial Data

The detection algorithm is computationally expensive for large data sets, particularly as the number of attribute combinations increases. To improve its efficiency, subsampling the data may allow less computationally expensive detection than computing on the entire data set. We use subsample sizes of 10%, 30%, 50%, 60%, and 90% of records to assess this approach. For each subsample size, we select five sample sets and run the Simpson's paradox detection algorithm on them. The algorithm's results on the entire data set provide ground truth.

Figure 3.5 shows the average and standard deviation F_1 scores across the sample sets of different subsample sizes. (The F_1 score indicates the correctness of the algorithm on a sampled data subset relative to on the full data set.)

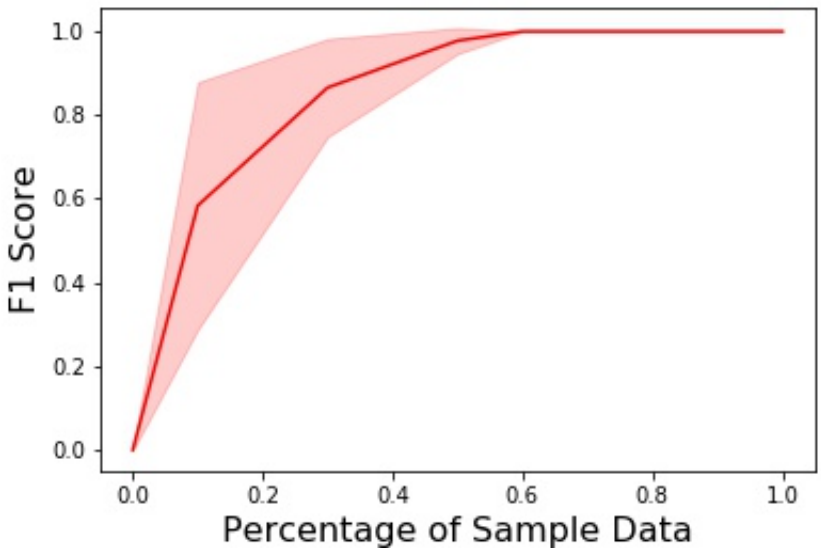


Figure 3.5: The F_1 score when running the Simpson's paradox over random samples of data.

The experiment indicates that the detection algorithm can achieve a high F_1 score even for relatively small subsets of the data. This suggests its potential for application in situations that require high efficiency, such as streaming data scenarios.

3.7 Summary

We present a new approach to detect Simpson's paradox based on the comparison of correlations. A case study on empirical data sets shows the effectiveness of the algorithm. We also explore the feasibility of detecting Simpson's paradox in subsampled data as a preliminary step toward improved scalability. Benchmark results confirm that the total number of continuous attributes and categorical attributes, the total number of records, and the number of unique values for each categorical attribute influence the running time of the algorithm.

This calls for considering the factors needed to sensibly filter real data sets. The currently implemented algorithm partitions the data only once and iterates the partition over the different categorical attributes of the entire data set. Grouping the attributes themselves (two or more) may be necessary to detect significant results of this form. However, grouping on all possible combinations of the categorical attributes would be both intricate and tedious, particularly because the interpretation of occurrences of trend reversals in particular combinations requires the application of in-depth domain knowledge about the data set and the phenomena behind its various data attributes. The following chapters present new visual and interactive techniques that use such occurrences to guide a user's data exploration and analysis.

Chapter 4

Representing Simpson's Paradox with Bivariate Colors

4.1 Overview

Seeking insights from data is a vital activity that increasingly requires human-centric computing approaches for success. Simpson's Paradox is a common phenomenon in data of many forms. Undetected occurrences can lead an unaware analyst to draw incorrect conclusions. In one famous example, researchers erroneously quantified the success rate of a treatment [30]. The error was not discovered until years later [77]. Moreover, as the name suggests, Simpson's paradox is generally a surprising observation and often counterintuitive to non-statisticians. It is increasingly important for individuals without statistical expertise to conduct exploratory analyses of data. As such, interpretable approaches to detect and analyze Simpson's paradox are increasingly important. Similarly, the democratization of data science is empowering both experts and non-experts alike to explore data, making easily interpretable techniques for detecting phenomena like Simpson's paradox essential.

We categorize cases of Simpson's paradox into two types based on the form of detection: rank trend and regression trend. In rank trend Simpson's paradox, the trend is of the

relative rates of a binary outcome in two groups, for example, admissions by gender [18, 65]. In regression trend Simpson's paradox, the trend is based on the sign of a correlation between two variables [31, 80, 164]. While automated methods for the detection of Simpson's paradox [164] can help to facilitate the process of discovery, black-box method detection may not be sufficient to support analysts' sense-making processes.

We take advantage of visualization techniques to improve interpretability and make human-in-the-loop data analytics more accessible to analysts with varied skills and expertise. A novel visual analytics approach facilitates the exploration of data to detect Simpson's paradox. A bivariate color scheme illustrates relationships between trends over a full population and within its subgroups. Detection and analysis activities are integrated via multiple coordinated views that provide users with both an overview of the entire data set and details about subgroups and detected occurrences as desired. Interactive features allow users to effectively and efficiently conduct exploratory analyses. We demonstrate the utility of our visualization design to draw insights from data without being misled by undetected instances of Simpson's paradox.

4.2 Example Data

To illustrate the design, we utilize both real data and example synthetic data structured like real data sets in which Simpson's paradox has been noted. This allows control over the number of occurrences and amount of trend reversal to robustly assess the design.

4.2.1 Rank Trend Simpson's Paradox

To assess the technique with rank trends, we use a synthetic version of the well-known Berkeley admissions case of Simpson's paradox [18]. Rank trends involve three variable types:

dependent variable is a variable that has a rate in which the ranking flips (e.g., from

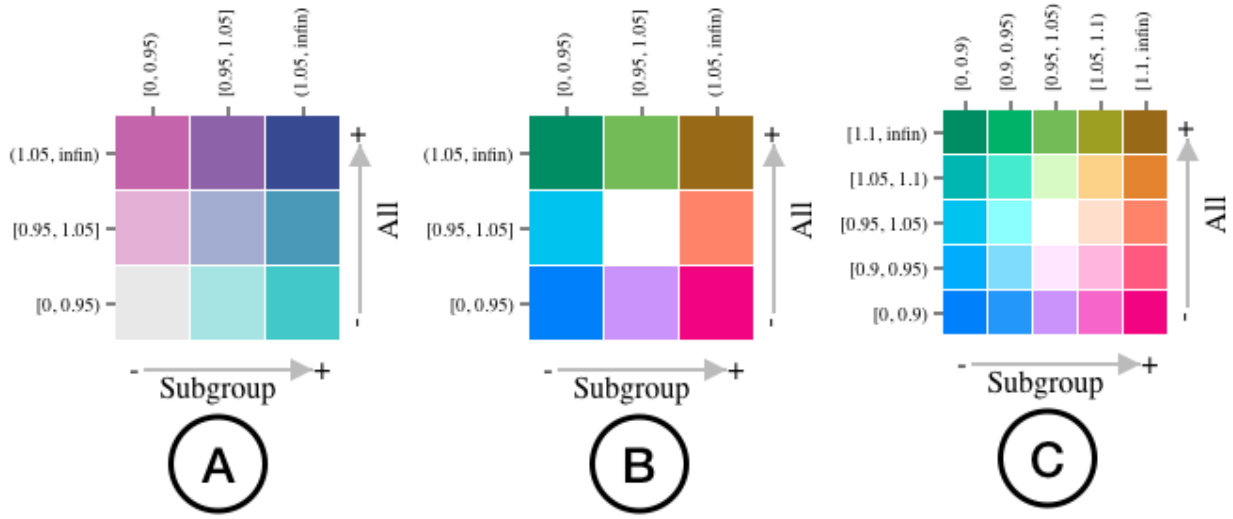


Figure 4.1: Bivariate color legends for rank trend Simpson's paradox: (A) 3×3 sequential-sequential, (B) 3×3 diverging-diverging, and (C) 5×5 diverging-diverging.

Women having a higher admission rate than Men to Men having a higher admission rate than Women), such as admission rate or mean salary;

independent variable is a grouping variable that determines how the rates are compared, such as gender or race; and

splitby variable is a grouping variable that groups rows having the same values into subgroups, such as department or job function.

We set the portion of applications to each department, the rate of each gender applying to each department, and each department's acceptance rate, and generate a random sample with 1000 records that follow the causal explanation of the Berkeley example, namely that the departments with high admissions rates have higher shares of male applicants.

4.2.2 Regression Trend Simpson's Paradox

To assess the technique with regression trends, we generate samples from a Gaussian Mixture Model. The means are drawn from a Gaussian with a high positive correlation coefficient and cluster covariances constructed to have a high negative correlation. We add two continuous

variables (one as dependent variable and the other as independent variable) and a categorical variable as splitby variable. The size of the data set is 100 records.

4.3 Tool Design

The design is implemented as a web-based interactive visualization application using Bostock's D3.js library [22]. The application was designed for use with data from any domain. Labeling throughout the application user interface is generic or drawn from the data, and scales are adaptive to the data. The user interface design is unified to support both forms of Simpson's paradox detection.

4.3.1 Bivariate Color Scheme

A bivariate color scheme [140] is a two-dimensional array of colors that integrates a pair of univariate color schemes. This configuration illustrates the relationship between two quantitative variables and allows comparison within and between them. For example, the color in the right bottom cell of Figure 4.1(A) indicates that a subgroup trend has a negative aggregate trend and a positive subgroup trend. Figure 4.1 shows three examples: a 3×3 sequential/sequential scheme, a 3×3 diverging/diverging scheme, and a 5×5 diverging/diverging scheme. The vertical axis is split into ranges of trend values (rows) for the entire data set. The horizontal axis is similarly split into columns for subgroups. The top-left and bottom-right corners indicate a trend reversal between the overall trend and the subgroup trend, indicating a possible occurrence of Simpson's paradox. The 3×3 sequential/sequential scheme is derived from the logical arrangement of all combinations of the colors in two 3-class sequential color schemes [24]. The option to choose diverging color schemes allows a more perceptually direct examination of the critical midpoint. The 5×5 diverging/diverging scheme offers more levels to indicate the severity of the trend reversal (or lack of one). (Unfortunately, these three bivariate color schemes are less accessible to

users with color deficiency.)

4.3.2 Bivariate-Scaled Heatmaps

We apply the bivariate color scheme to color the cells of feature matrices, creating a heatmap to visually indicate trend reversals and Simpson's paradox. Correlation matrices are used to show regression trends and rate comparison matrices show rank trends.

Correlation Matrices

For a data set with d continuous variables, we compute the $d \times d$ correlation matrix for the whole population. After the data set is partitioned by the C group-by variables, we compute a $d \times d$ correlation matrix for each of the k_c values of group-by variable c. This gives $\sum_{c=1}^{C} k_c$ correlation matrices of size $d \times d$ for each subgroup and a correlation matrix of size $d \times d$ for the entire population. The correlation matrix for the whole population and each of the $\sum_{c=1}^{C} k_c$ subgroup-level correlation matrices are used to generate bivariate heatmaps for visualization $B_{c=s}$, in which each element is defined as $b_{ij} = f_{tran}(\text{corr}(a_i, a_j), \text{corr}(a_i, a_j | c = s))$ where a_i and a_j are the pair of continuous variables, c is the categorical variable, and s is the subgroup value. The transformation function $f_{tran}(\cdot)$ maps the correlations of the entire population and subgroup s into a color in the bivariate color scheme [136]. Since Simpson's paradox is a change of sign, the f_{tran} for regression compares the sign of the correlation coefficients.

Rate Comparison Matrices

Consider a data set with a binary dependent variable Y and variables, $x_1, \ldots x_G$ for grouping G. Assume no prior knowledge about how to distinguish which grouping variables serve as independent variables versus splitby variables; use each grouping variable as both an independent variable and as a splitby variable. This gives $G(G-1)$ total rate comparison matrices to compute. Assume each grouping variable x_g has N_g values: $x_{g,1}, \ldots x_{g,N_g}$. Given a pair of grouping variables (i,j), taking x_i as the independent variable and x_j as the explanatory

variable, compute an $M_i \times N_j$ rate comparison matrix, using $M_i = \binom{N_i}{2}$ comparisons for the N_i values of the independent variable x_i, and the N_j values of the splitby variable x_j, with $i, j \in 1, \ldots, G$.

First, compute the G overall comparison vectors:

$$D_{i,all} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{M_i} \end{bmatrix} = \begin{bmatrix} \frac{E(Y|x_i = x_{i,1})}{E(Y|x_i = x_{i,2})} \\ \frac{E(Y|x_i = x_{i,1})}{E(Y|x_i = x_{i,3})} \\ \vdots \\ \frac{E(Y|x_i = x_{i,N_i-1})}{E(Y|x_i = x_{i,N_i})} \end{bmatrix} \tag{4.1}$$

Each entry in D_{all} is the ratio between the group-wise means of the dependent variable Y conditioned on two values of the i^{th} grouping variable. For example, if the first grouping variable, x_1, is binary coded gender and Y is binary admission decision, then $N_1 = 2$, $M_1 = \binom{2}{2} = 1$, and $D_{1,all}R^{1\times 1}$ is the ratio of the admission rate of men to women.

Next, partition the data set by further conditioning on each of the n values of the splitby variable and compute a $m \times n$ rate comparison matrix, $D_{i,j}$. Each column s_l in $D_{i,j}$ is the rate between the means of the dependent variable Y conditioned under (x_i, x_j). For the example above, if the splitby variable x_2 is department and $N_2 = 4$, then $D_{1,2} \in R^{1\times 2}$ is the per department ratios of the admission rate of men to women.

$$s_{:,l} = \begin{bmatrix} \frac{E(Y|x_i = x_{i,1}, x_j = x_{j,l})}{E(Y|x_i = x_{i,2}, x_j = x_{j,l})} \\ \frac{E(Y|x_i = x_{i,1}, x_j = x_{j,l})}{E(Y|x_i = x_{i,3}, x_j = x_{j,l})} \\ \vdots \\ \frac{E(Y|x_i = x_{i,N_i-1}, x_j = x_{j,l})}{E(Y|x_i = x_{i,N_i}, x_j = x_{j,l})} \end{bmatrix} \tag{4.2}$$

Next, use the G overall matrices, $D_{i,all}$, and the $G!$ rate comparison matrices $D_{i,j}$ to

generate trend comparison matrices of size $M_i \times N_j$.

Finally, each entry b_{ij} in B is computed by the transformation function $f_{tran}(\cdot)$ that maps a_i from D_{all} and s_{ij} from $D_{subgroup}$ into the position in the bivariate color scheme: $b_{ij} = f_{tran}(a_i, s_{ij})$.

4.4 Visual Analysis Workflow

To facilitate the exploration of Simpson's paradox, the web-based application is organized as follows. On the left, a control panel allows users to upload a file, select a Simpson's paradox type, and change the color scheme. In the middle, bivariate-scale heatmaps help users detect Simpson's paradox. On the right, a detail panel (for rank trend Simpson's paradox) or a scatter plot (for regression trend Simpson's paradox) shows the data and trends themselves. The application user interfaces for detecting rank trend and regression trend Simpson's paradox are shown in Figures 4.2 and 4.3, respectively. After a data set is loaded into the application, the user can choose the rank trend or regression trend Simpson's paradox detector by selecting from the type dropdown selector. Currently, the system requires a comma-separated values (CSV) file with headers for analysis. The following sections describe the operation for each type of Simpson's paradox.

4.4.1 Rank Trend Simpson's Paradox

In Figure 4.2, once users select rank trend, they can choose a color scheme from the legend dropdown selector. In the middle view, the rate comparison matrices are colored according to the bivariate color scheme. To facilitate visual exploration, users can highlight cells by clicking a cell in the legend. The user can identify occurrences of Simpson's paradox in the heatmaps by looking at their cells' colors. For example, the light pink of the Sequential 3×3 legend may indicate Simpson's paradox, since the light pink represents a low comparison rate in the range $(0.00, 0.95)$ in the subgroup but a medium comparison rate in the

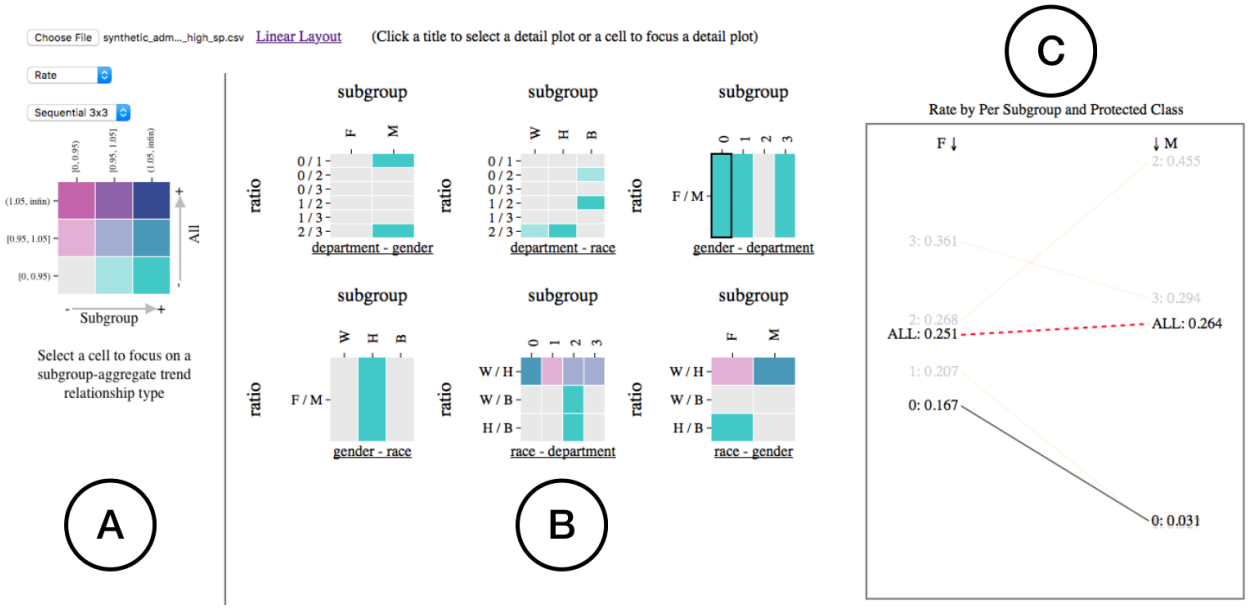


Figure 4.2: The application user interface for rank trend Simpson's paradox detection: (A) the user control panel; (B) bivariate scaled heatmaps for rate comparison matrices; (C) a slope graph showing details for the selected heatmap and cell in it.

range $(0.95, 1.05)$ in the entire data set. This situation could be a Simpson's paradox if the comparison rate is greater than 1.00. In the synthetic data set, the third rate comparison matrix in the first row has four cells in light pink. This means that in each department, females had a higher acceptance rate than males but the acceptance rate for males is higher than for females in the entire data set. After the user clicks on the light pink cell, the slope graph shows the details of acceptance rate for males and females for each department and for the entire data set. The user can compare the trend line of the selected subgroup to that of the whole in the slope graph, with the other subgroups' trend lines faded out. The user can double-click on the clicked cell to bring the trend lines for all subgroups to equal visual prominence.

4.4.2 Regression Trend Simpson's Paradox

In Figure 4.3, after the data file is loaded, the user can select regression type from the type dropdown selector and pick a color scheme for regression trend detection. In the middle, the

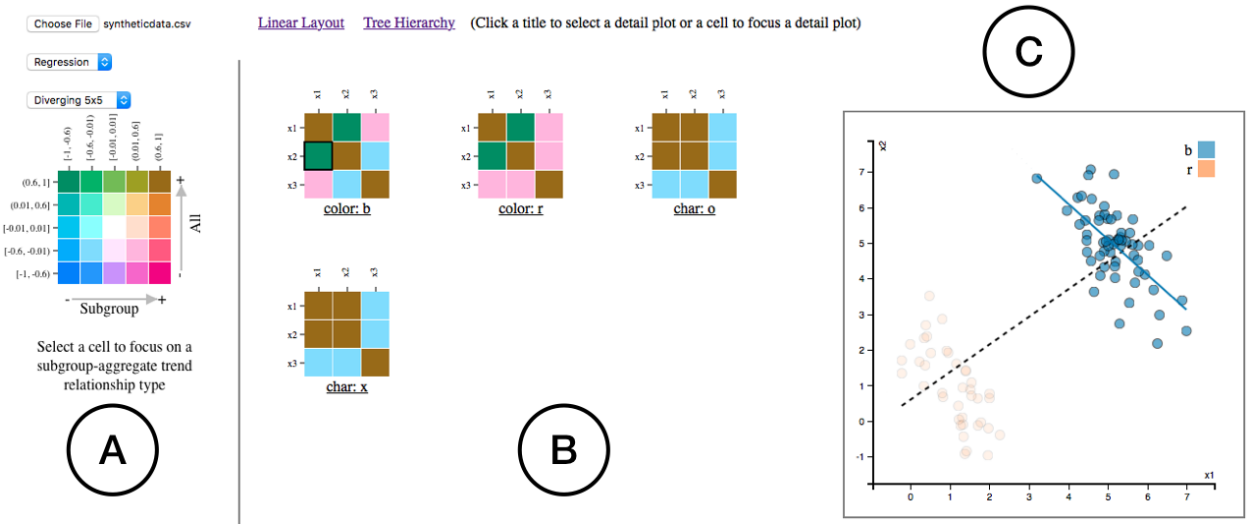


Figure 4.3: The application user interface for regression trend Simpson's paradox detection: (A) the user control panel; (B) bivariate scaled heatmaps for correlation matrices; (C) a scatterplot showing details for the selected heatmap and cell in it.

bivariate heatmaps show subgroup-level matrices for identifying trend cases. The selected dark green cell in Figure 4.3B indicates the trend reversal occurring in the color b group for continuous variables x_1 and x_2. To understand the possible occurrence of Simpson's paradox in the data set, the scatterplot shows both data and trends (Figure 4.3C). The data points related to the chosen subgroup are highlighted. The black-dotted line in the scatterplot shows the regression line for the entire data set. The solid line represents the regression line for the chosen subgroup. The reversal in direction between the two lines provides evidence of the opposing relationship between the two variables.

A three-level tree layout (Figure 4.4) is included to illustrate the hierarchy of groupings. The color in root level and middle level matrices represents the aggregate trend, whereas the color in leaf matrices indicates the relationship between the aggregate trend and the subgroup trend. The tree layout view helps to identify which categorical variables exhibit trend reversals. For example, the leftmost two leaf matrices b and r have dark pink cells indicating trend reversals, and the tree layout shows that they come from the categorical variable *color*. If the user clicks on the matrices of non-leaf nodes (here color, char, and all), all data points are highlighted in the scatter plot. The tree view allows users to interactively

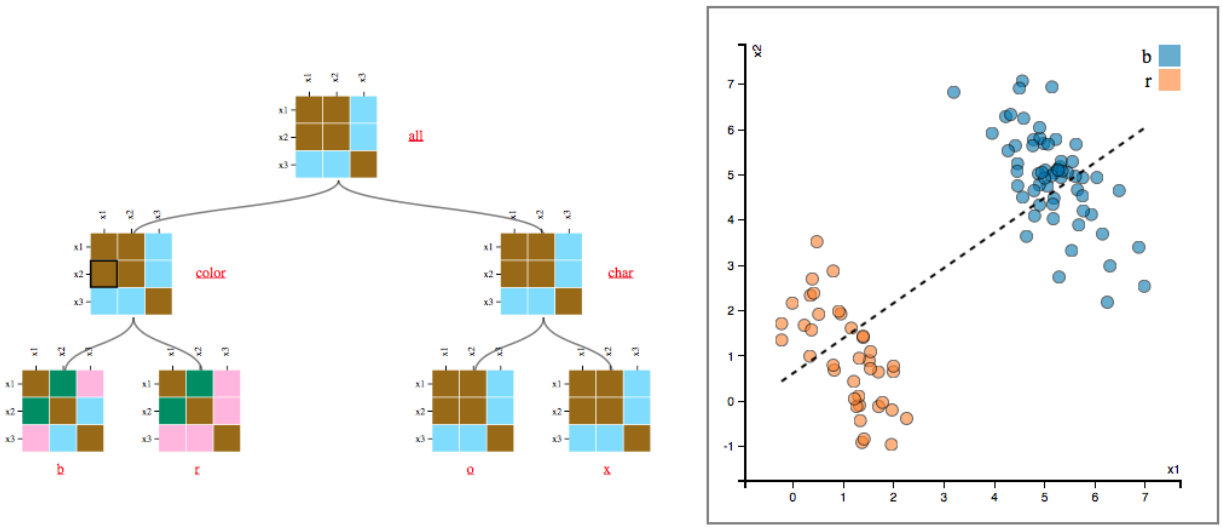


Figure 4.4: The tree layout for regression trend Simpson's paradox. The user clicks on the matrix of the parent node (color), and all data points are highlighted in the scatterplot.

explore the area of the tree that they want to see by dragging the tree or zooming in/out by mouse or touchpad.

4.5 Summary

We present an interactive interface that facilitates visual detection of Simpson's paradox by non-experts during exploration. Bivariate-scale heatmaps of the features indicate both directions of the trend and subgroup-aggregate trend relationship (same or reversed).

The visualization design is an initial step toward an interactive data exploration framework for alerting the user to surprising results. We aimed to develop the visualization further to incorporate ranking of the Simpson's paradox occurrences to better accommodate large data sets. Directions for extension include more flexible and varied trend detection, user control of display thresholds, and joint visualization of multiple trend types. This led to the Wiggum application discussed in the next chapter.

Chapter 5

Wiggum: Interactive Visual Analytics for Examining Mix Effects

5.1 Overview

The importance of data-driven decision-making is rapidly increasing thanks in part to the growing availability and accessibility of data sets and analysis tools. Yet, applicable insight can be difficult due to biases and anomalies in data. An often overlooked phenomenon is *mix effects*, in which subgroups of data exhibit patterns opposite to the data as a whole. This phenomenon is widespread and often leads inexperienced analysts to draw incorrect statistical conclusions. In addition, *Simpson's paradox* is a special case of mix effects, occurring when all subgroups of data partitioned by a certain condition evince the opposite pattern of the aggregate data. In this chapter, we present Wiggum, an interactive visual analysis system for uncovering both mix effects and Simpson's paradox.

Wiggum represents a punctuation point in our work to develop visual analysis tools for a broad community of users concerned with paradoxes and other complex anomalies in multi-dimensional data sets. This work confronts critical challenges to the effective examination of mix effects. In developing Wiggum, we pursue four main research goals (**R1**–**R4**) to address

those challenges.

R1 **Dimensionality**: Checking a multidimensional data set for mix effects can be a challenging task because it requires examination of relationships between subsets of dimensions in terms of the partitions of data in those dimensions on subgrouping criteria. Wiggum provides features to flexibly drill-down into dimensions and specify trend type and criteria.

R2 **Flexibility**: Visual identification and characterization of mix effects can be challenging because the interpretation of trend strengths against the underlying data distribution varies with the trend type and the data types of dimensions considered. Wiggum generalizes the exploration process across different combinations of mix effects and dimension types while also providing visualizations suitable for examining each one.

R3 **Understandability**: Designing usable visualization tools can be challenging because users differ in their understanding of statistics concepts and how those concepts manifest in data, especially when data is transformed along a pipeline for visualization. A participant-based evaluation of Wiggum sheds light on how its exploratory drill-down design helps users examine mix effects in high dimensional data.

R4 **Integrability**: Exploring mix effects in multidimensional data can be difficult because of the need to bring together independent tools for tasks including selection of dimensions, definition of subgroups, and examination of trends. Wiggum integrates visualizations for examining mix effects with a user interface for loading data, browsing dimensions, specifying groupings, and selecting groups to be examined.

Taken together, Wiggum's features support interactive visual identification and examination of mix effects. While automated approaches to find mix effect candidates exist, one must select and examine them to interpret their character, strength, and meaning. Automated methods can thus complement but not supplant manual identification. For the current ver-

sion of Wiggum, we focus on supporting examination of candidates detected by *some* means, and leave addition of automated methods as future work.

To help data analysts better identify and interpret mix effects, Wiggum includes data annotation features to specify variable types and roles. It also offers data augmentation features to generate clusters, quantiles, or intersectional subgroups [26, 28].

We introduce *trend strength* and *trend distance* metrics to support comparison of aggregate and subgroup trends. A *distance heatmap view* provides an overview for efficient exploration of trend measures and lets users inspect patterns of relationships across dimensions for defined subgroups. A second view (specific to trend type) shows the details of the selected trend for examination and interpretation. Interactive filtering and ranking features allow users to efficiently organize trends for exploratory analyses. Our contributions are as follows:

- a **visual analysis system** that supports interactive exploration to examine patterns that reveal mix effects;
- a **processing pipeline** to map data, annotated and augmented by the user, into statistical results for display;
- **mathematical equations** to formalize metrics of mix effects;
- two new **view designs**, adapted from heatmaps and trend plots, to understand and efficiently examine mix effects; and
- a comprehensive **approach** to help users examine *multiple* trend types and explore *different* trend types simultaneously.

We start with an illustrative example of mix effects examination. We then describe how Wiggum prepares and processes data for visual analysis of mix effects. Next, we outline key goals to support visual analysis and the design of interaction visualization features in Wiggum to achieve them.

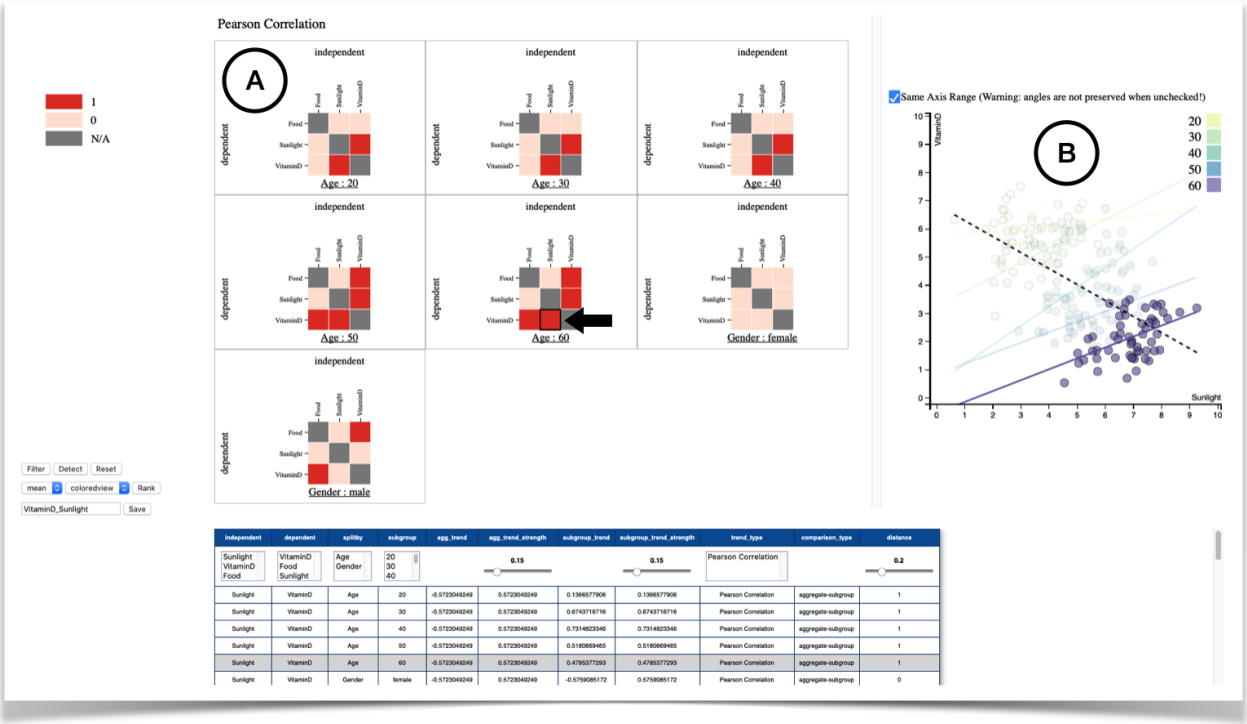


Figure 5.1: Selecting a subgroup and examining trends on the Wiggum visualization page. After selecting the age 60 group and clicking a dark red cell in its distance heatmap (A), the detail view (B) displays the trend reversal between the aggregate data and the age subgroups in a scatter plot.

5.2 Illustrative Example

We illustrate how Wiggum supports visual data exploration of a synthetic data set relating vitamin D level, sunlight exposure level (sunlight), food level in vitamin D (food), age, and gender. (Key concepts are italicized throughout, and will be described more formally in later sections.)

After loading the data set, we annotate the data type and role of each variable on the data configuration page. To study correlations with vitamin D level, we choose the *Pearson correlation* trend type then click the *Visualize Trends* button. On the visualization page, we observe seven 3×3 distance heatmaps (Fig. 5.1A). Each distance heatmap represents one subgroup of either the age group or the gender group. Each cell color encodes the pairwise attribute *trend distance* that measures the discrepancy between the aggregate and subgroup trends. A dark red cell encodes a distance of 1, indicating a trend reversal between the

aggregate and subgroup.

Clicking the *VitaminD x Sunlight* cell in the heatmap for the age 60 subgroup, the detail view (Fig. 5.1B) displays a scatter plot of the age subgroups. Each subgroup and its Pearson correlation trend are color encoded with the age 60 subgroup highlighted in solid purple. Its regression line in the plot indicates a positive relationship between vitamin D level and sunlight exposure level. In contrast, the dashed black regression line indicates a negative relationship between them in the aggregate data set. This reversal pattern matches the meaning of the selected dark red cell in the heatmap. The regression lines for the other age groups are similarly reversed from the aggregate data set, although to different degrees.

Mix effects and Simpson's paradox can be expressed mathematically. A trend distance d is defined in Equation (5.1), using a distance function, dist, and a binary relationship, trend_b, for some summary statistic, stat, of two variables x_1 and x_2. For a single value of $x_3 = y$ (y is a value of the variable x_3, e.g., $age = 60$), the value of d is the *trend distance* between a *subgroup trend y* (i.e., age 60) and the aggregate data. We define a *subgroup trend* as a trend between two variables x_1 and x_2 given $x_3 = y$. In the synthetic data, the stat is the Pearson correlation between vitamin D level (x_1) and sunlight exposure level (x_2) conditioned on age (x_3). The trend_b is positive or negative according to the direction of the correlation. A distance of 0 indicates the trends have the same direction, and 1 indicates a reversal. In this case, the trend between vitamin D level and sunlight exposure level is negative for the aggregate data but positive for the age 60 subgroup. Since the two trends are reversed, the distance between them is 1.

$$d_{x_1,x_2,y} = \text{dist}(\text{trend}_\text{b}(\text{stat}(x_1, x_2)), \text{trend}_\text{b}(\text{stat}(x_1, x_2|x_3 = y))) \tag{5.1}$$

Mix effects occur when reversal ($d = 1$) happens for *some* values of x_3. If reversal occurs for *all* values of x_3, it is a special case known as *Simpson's paradox*. The synthetic data exhibits both Simpson's paradox and mix effects. The correlation between vitamin D level

and food illustrates mix effects, since only two out of five age subgroups (50 and 60) have the reverse trend. The correlation between vitamin D level and sunlight exemplifies Simpson's paradox, since all five age subgroups have the reverse trend. Ignoring these phenomena could lead to incorrect conclusions and decision making.

5.3 Data Preparation and Processing

In this section, we describe the processing pipeline that Wiggum applies to map a data set into the statistical information displayed in its visualizations. The process begins with a data preparation phase, in which the user provides metadata and applies any preprocessing needed for their investigation. Next, Wiggum extracts features and partitions the data as inputs to trend generation. Finally, it calculates trend measures for each subgroup.

5.3.1 Data Preparation

Data preparation happens in two stages, both under interactive user control. In *data annotation*, the user labels the data attributes of the loaded data set with the role they should play in trend calculations. In *data augmentation*, the user can define additional categorical variables for use in partitioning data.

Data Annotation

Wiggum needs type and role information for each data dimension (variable) to determine how to apply rank and regression trend calculations. Upon loading a data set, Wiggum applies an automatic type-mapping function to infer a type for each variable. The user can edit the type to be binary, categorical, continuous, or ordinal. They can also specify each variable's role to be *dependent*, *independent*, or *splitby*.

A *feature* is a variable that has been cast into a type and put into a role for the purpose of calculating trends, as shown in Table 5.1. An instance of mix effects involves three features.

Table 5.1: User-defined types and roles of features in instances of mix effects, for each of the two types of trends.

Feature in Eq. 5.1	Rank		Regression	
	Type	Role	Type	Role
x_1	binary/continuous	dependent	continuous/ordinal	dependent
x_2	categorical	independent	continuous/ordinal	independent
x_3	any	splitby	any	splitby

The first two features, one dependent and one independent, are used to compute a trend. For a rank trend, an aggregation (e.g., average) is applied to values of the *dependent* feature. The *independent* feature partitions a population into ranked groups (e.g., {Women, Men} for *Gender*) based on the aggregation result. For example, in the admissions data set, admission rate is calculated as the mean of a *dependent* feature (*Accept Status* as a binary type). The rank by admission rate of the *independent* feature (*Gender* as a categorical type) determines the rank trend. For a regression trend, Wiggum models the relationship between the features (details in Section 5.3.2). For both types of trend, a third *splitby* feature groups rows by the other features' values.

A drop-down list lets users select multiple roles for a variable in a data set, making it possible to have multiple features in which the same variable plays different roles. Users may want to consider only a subset of variables, especially for high-dimensional data sets. Wiggum provides an *ignore* role option for users to exclude variables that are not of interest. (Note: For the rest of the chapter we use the terms *variable* and *feature* interchangeably.)

Data Augmentation

For many data sets and problem definitions, existing categorical variables can be used as *splitby* attributes for partitioning. Wiggum also lets users define calculated categorical variables. The values of each calculated variable populate an additional column in the data. Wiggum currently supports calculations to discretize the values of a quantitative attribute into quantiles. Users can also create intersectional combinations of other categorical vari-

ables. For example, a user can select features "race" and "gender" to generate a new categorical column called "race_gender" containing values such as "Asian male" from "Asian" and "male". Creation of intersectional variables allows users to discover reverse trends in more complex groupings. Wiggum also lets users define categories via clustering using the Dirichlet Process Gaussian Mixture Model implemented in the Scikit-learn library [112]. The user is not required to specify the number of clusters. We set the maximum number of mixture components to 20, which limits the number of clusters to 20. The user can interact with a range slider to adjust cluster quality and tune the resulting clusters.

5.3.2 Trend Extraction

Wiggum selects subsets of the available features and partitions them into subgroups. It then generates rank and regression trends based on variable types and roles.

Feature Subset and Subgroup Creation

A *subset* is a pair of features along with their respective columns populated with the data values of each feature's variable cast into the feature's type. Wiggum uses the user's entered data annotations to calculate all subsets of the entire data set that are relevant to the selected trend type (see Figure 5.2).

In a typical rank trend analysis, users are interested in ranking different groups on a particular statistic. We define a dependent variable as a data column whose values are used to calculate such a statistic. In Equation (5.1), the dependent and independent variables correspond to features x_1 and x_2, respectively. Wiggum processes the data to find all column pairings that contain one dependent variable and one independent variable. The total number of subsets detected for rank trend mix effects is the number of dependent variables times the number of independent variables. To create *subgroups* for rank trend, Wiggum iterates over the list of splitby variables and partitions each subset on the unique values of each splitby variable. If a splitby variable is also in a subset as an independent variable, it will be skipped

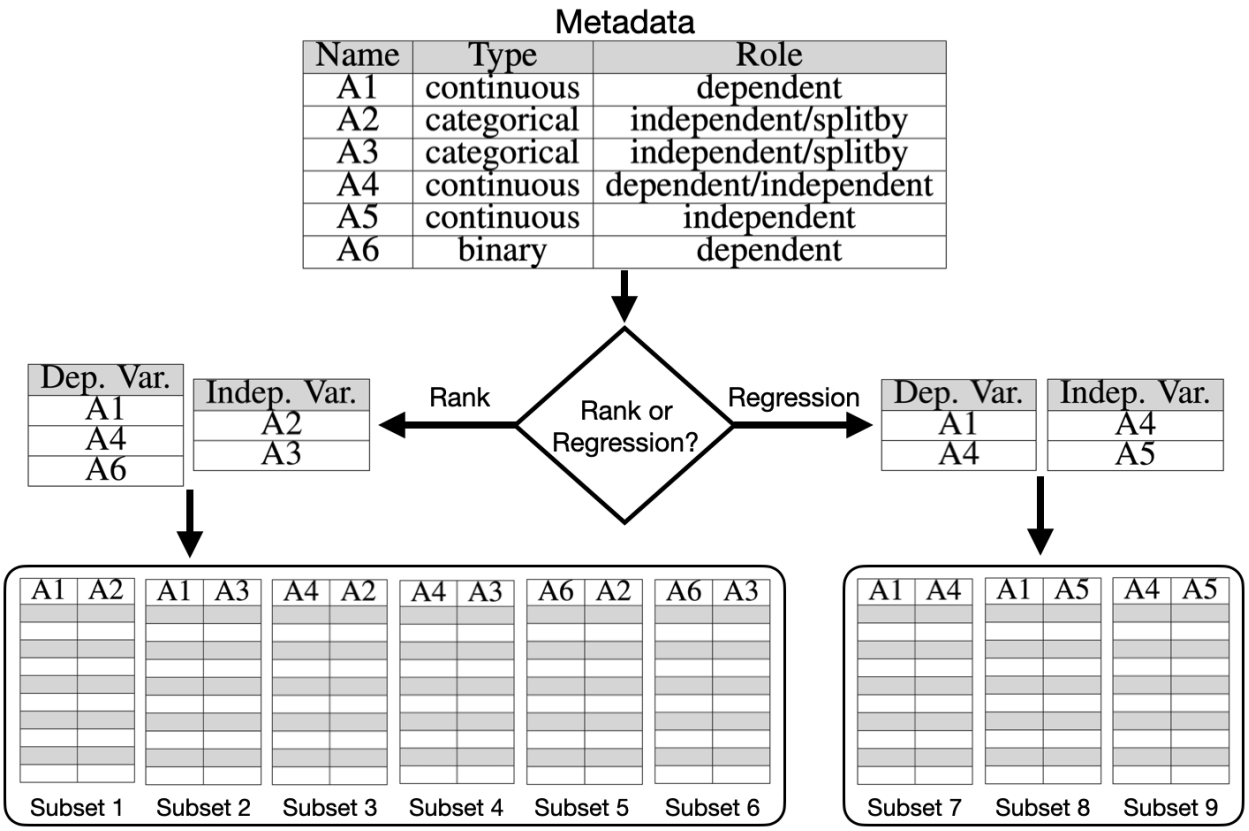


Figure 5.2: Subset generation. For each user-annotated feature (A1–A6), Wiggum generates dependent and independent feature lists appropriate to the user-specified trend type. Generated subsets consist of all possible non-same pairings between the dependent and independent feature lists.

and will not be used to partition the subset.

In a regression trend analysis, the idea is to model the relationship between a pair of continuous or ordinal variables. Variables specified as *continuous* or *ordinal* type in a *dependent* or *independent* role are used for subset generation. The subsets are all possible pairs of dependent and independent variables, excluding pairings of a variable with itself. Variable pairs are treated as x_1 and x_2 in Equation (5.1) to detect instances of regression trend mix effects. For M dependent variables, N independent variables, and P same variable pairs, the total number of subsets for regression trend detection is $M \times N - P$. Unlike rank trend, there is no conflict between *splitby* variables and *independent* variables, so all of the *splitby* variables can be used to partition subsets.

In practice, a data variable may play multiple roles in a data set. For instance, feature

A4 in Figure 5.2 is annotated as playing both *dependent* and *independent* roles. In subset generation, *A4* is paired both with continuous/dependent feature *A1* in Subset 7 and with continuous/independent variable *A5* in Subset 9. Both subsets are used for regression trend detection.

Generating Trends

Given subgroups of different subsets, the next (vital) step is to compute trends for both subgroups and the aggregate data subset. Trend generation applies the same method to both the aggregate data and subgroups. For rank trends, Wiggum computes a summary statistic, such as mean or median, on X_{Dep} (e.g., the admitted variable: 1 admit, 0 do not admit) for every group value $x_i \in X_{Indep} = \{x_1, ..., x_n\}$ (e.g., gender = {Men, Women}). The next step is to rank the n groups by the summary statistic. The trend is an ordered list of ranked groups. In the Berkeley Admissions Example, the trend for aggregate data can be denoted as $t_a = [Women, Men]$ indicating that the admission rate for men is higher than the admission rate for women.

For a regression trend, we consider correlation for a subset. Wiggum applies Pearson correlation on the pair of columns in the subset. We denote the *trend* as $t = corr(X_{Dep}, X_{Indep})$, where X_{Dep} is the variable of the *dependent* role and X_{Indep} is the variable of the *independent* role. A difference in the signs of the correlations for the subgroup and aggregate data indicates a reverse trend.

Despite being a powerful method for detecting mix effects, correlation alone does not suffice to detect the difference between trends without trend reversal. To allow users to track changes in trends even without reversal, the slope of a linear regression line is used as the trend value. Then $t = b$ where b is derived in the equation of a linear regression line $X_{Dep} = b \cdot X_{Indep} + a$.

5.3.3 Trend Measurement

Wiggum introduces a practical set of measures to support discovering mix effects. We define two basic measures: *trend strength* and *trend distance*.

Trend Strength

Trend strength is a metric for assessing how well a trend represents data being examined. For a rank trend, given a subgroup trend t, the strength $s(t)$ indicates how the subgroup trend list and the element-wise sorted list are dissimilar to each other after repeating each element of the subgroup trend by the corresponding proportion to reproduce the full list with the same length. We denote the extended subgroup trend list as L_s. Wiggum generates an element-wise sorted list denoted as L_a by sorting on the X_{Dep} column of the subset, then creates a list of the elements in X_{Indep}. The strength is computed using the absolute value of Kendall's tau similarity between the two lists L_a and L_s. We formulate the strength as:

$$s(t) = |\tau(L_a, L_s)| = \left| \frac{P - Q}{\sqrt{(P + Q + T)(P + Q + U)}} \right|, \tag{5.2}$$

in which P is the number of concordant pairs, Q is the number of discordant pairs, T is the number of ties only in L_a, and U is the number of ties only in L_s. For example, if $L_a = [M, F]$ and $L_s = [F, M]$, then $P = 0$, $Q = 2$, $T = 0$, $U = 0$ and $s(t) = 1$.

For a regression trend, the absolute correlation indicates the strength of the association of the two attributes in the observed subgroup. We denote the trend strength for a regression trend as:

$$s(t) = |\operatorname{corr}(X_{Dep}, X_{Indep})|, \tag{5.3}$$

which quantifies the strength of the relationship between dependent and independent variables to measure how well the regression trend fits the data.

Trend Distance

Distance is a fundamental concept in Wiggum. Given an aggregate trend t_a and a subgroup trend t_s, a pairwise distance $d(t_a, t_s)$ indicates the discrepancy between two trends. To make multiple subgroup trends in different types of trends consistent and comparable, the distances in Wiggum are normalized to $[0, 1]$; 0 is considered "the exact same trend" and 1 "the largest possible difference". For a rank trend we use Kendall's Tau similarity between the aggregate and subgroup trends, since they are represented as lists. We formulate the normalized distance for a rank trend as:

$$d_\tau(t_a, t_s) = 1 - \frac{(\tau(t_a, t_s) + 1)}{2}. \tag{5.4}$$

For regression trends, we use two different strategies of detection to increase the flexibility of the trend comparison. When users only care if the subgroup exhibits the opposite pattern of the aggregate data, Wiggum checks the sign of the correlation. The distance for a subgroup trend is defined as:

$$d_\oplus(t_a, t_s) = \text{sign}(t_a) \oplus \text{sign}(t_s), \tag{5.5}$$

in which *sign* refers to mapping the correlation by its sign to 0 or 1 (1 for positive, 0 for negative) and $\oplus$ is the logic operation for exclusive-OR. In addition to being useful for detection of a reverse trend, Wiggum can help users explore the difference between trends to figure out whether a trend is a recurring phenomenon. We use the slope as the trend value, with distance defined as the normalized angle between two linear regression lines, as:

$$d_\measuredangle(t_a, t_s) = \text{normangle}(t_a, t_s) = \frac{2}{\pi}\left(\left|\tan^{-1}(t_a) - \tan^{-1}(t_s)\right| \% \frac{\pi}{2}\right), \tag{5.6}$$

in which % is the modulo operator, t_a is the slope for the aggregate data, and t_s is the slope for the subgroup data. The normalization step generates a distance $d_\measuredangle \in [0, 1]$ where $d_\measuredangle = 1$

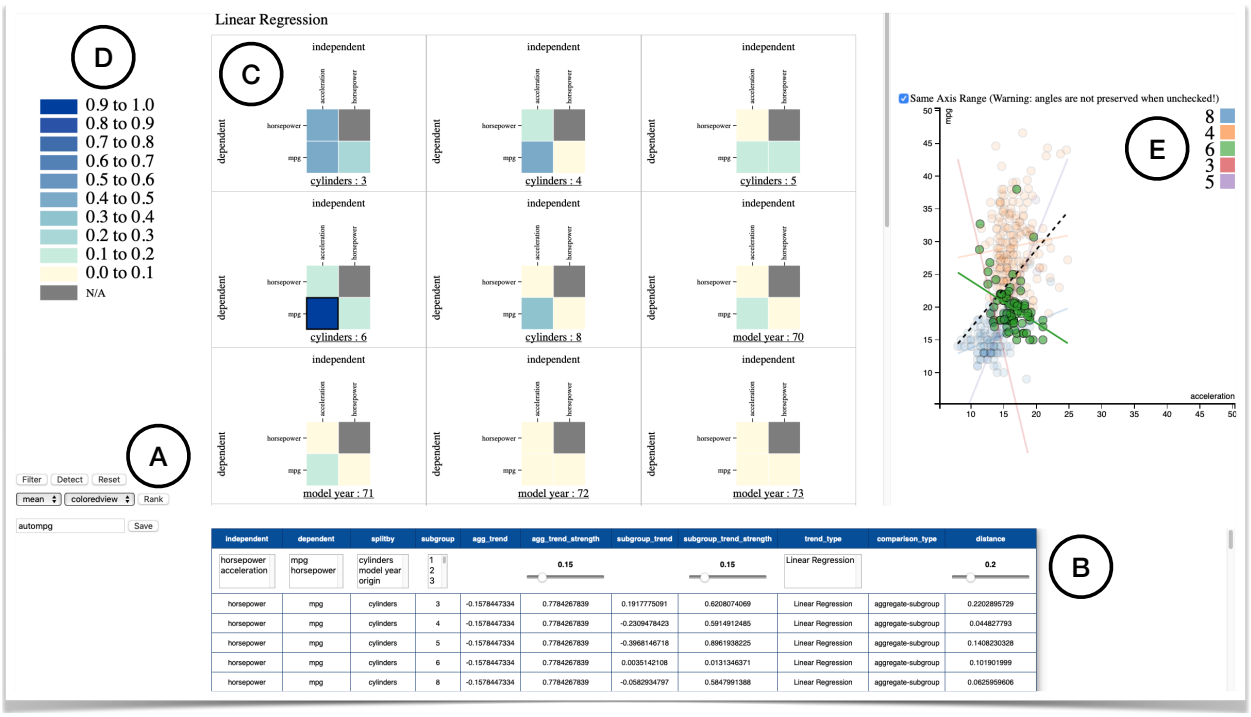


Figure 5.3: The Auto MPG data set [90] in Wiggum. (A) Control panel for filtering, detecting, and ranking potential instances of mix effects. (B) Result table for examining statistical details of each subgroup trend. (C) Scrollable array with a distance heatmap for each result subgroup trend. (D) Legend of heatmap cell colors. (E) Trend plot for exploring details of regression trends.

indicates a right angle and $d_{\measuredangle} = 0$ indicates parallel lines.

5.4 Wiggum: Discovering Mix Effects

Wiggum is a web application consisting of a data preparation page and a visualization page, as shown in Figure 5.3. In the data preparation page, the user can pick trend types, label metadata, and perform data augmentation. Wiggum is motivated by the Visual Information Seeking Mantra: "Overview first, zoom and filter, then details-on-demand" [131]. The visualization page is made up of a distance heatmap collection for overview, a trend plot and result table for detail, and a control panel area to support zoom and filter operations.

5.4.1 Design Goals

Wiggum was designed to be an interactive visual system for exploring mix effects in multi-dimensional data. We established the following set of goals to guide design and evaluation.

G1 Provide easy generation of variables for partitioning data. Although some data sets have categorical or ordinal variables which can be used for partitioning, it is often useful to provide users with a set of approaches to generate subgroups without direct application to existing categorical or ordinal variables. Users should be able to specify the inputs for such data augmentation and quickly generate variables for partitioning.

G2 Support multiple trend types. The system should support detection of multiple, statistically important trend types (e.g., rank trend and regression trend). The visualization design should be general enough to reveal the statistical characteristics for different trend types (**R2**).

G3 Present an overview of the trend distance. The visual design should clearly show the trend distances generated by the detection algorithm in a way that lets users explore them quickly to identify and analyze interesting patterns (**R1**).

G4 Facilitate the interpretation of exploration results. The system should help users understand trend measurements to discern how a subgroup trend reverses compared to the aggregate trend. Users should be able to visualize all subgroups' trends to discover whether mix effects or Simpson's paradox exists in the data set (**R3**).

G5 Allow flexible selection of subgroup trends. Since users may have domain knowledge about important feature attributes or subgroups they want to check, users should be able to select records in the result table using quick, simple interactions.

5.4.2 Preparation Page

On the preparation page, Wiggum lets users load data from a new or previously saved CSV file. Data annotation and augmentation are integrated into the same page. Wiggum allows users to select variables to generate quantiles or make intersectional subgroups. It also supports Dirichlet Process Gaussian Mixture Model clustering over pairs of continuous variables by setting a cluster quality threshold (**G1**). Wiggum equips a list of available trend types such as Pearson correlation, linear regression, and rank trend in a selection box to allow users to select one or multiple options at a time (**G2**). Upon clicking the Visualize Trends button, Wiggum redirects users to the visualization web page that includes the views discussed below.

5.4.3 Result Table View

The output of the data processing pipeline is a result table (Figure 5.3B) for each trend-level comparison. After executing the statistical computation in the Wiggum back-end, the user is provided with a result table view containing information on *dependent*, *independent*, and *splitby* variables, subgroups, and trend types. Trend, trend strength, and trend distance for subgroup and aggregate, respectively, are also included in the result table for each subgroup trend. Each record in the result table represents a subgroup trend from a feature subset. If there are m subsets (see Section 5.3.2), n splitby variables, and the ith *splitby* variable has k_i values, then the total number of rows in the result table will be $m \times \sum_{i=1}^{n} k_i$. In this view, users can find detailed statistics for a trend in an observational subgroup (**G4**). The result table is a basic component containing all information needed to generate the distance heatmap view. To further investigate subgroup trends, the area below the table header provides interactive selection boxes and sliders for use in conjunction with the Filter and Detect buttons.

5.4.4 Distance Heatmap View

The distance heatmap view provides an overview of the subgroup trends from the result table (**G3**). The view consists of multiple distance matrix heatmaps. We designed the distance heatmap view to show the measured distances between different paired features for each subgroup for each trend type. The different trend types generate heatmaps separately. For example, if there are n *splitby* variables used for partitioning in a data set and the ith *splitby* variable has k_i values, then there are $\sum_{i=1}^{n} k_i$ separate heatmaps for each trend type. If there are T trend types that are selected by users, the total number of heatmaps will be $T \times \sum_{i=1}^{n} k_i$. To use the central space more efficiently, Wiggum lays out three heatmaps per row with vertical scrolling.

The first step in generating a distance heatmap is to select rows from a subset of the result table which contains *dependent*, *independent*, *splitby*, subgroup, trend type, and distance information. For example, in the table in Figure 5.4, each row represents a subgroup trend in the Auto MPG data set. The selection iterates over trend type, *splitby*, and subgroup; each subset has the same values of these three things. The second step is to reshape the subset data into a two-dimensional matrix, as illustrated in Figure 5.4B. The matrix's values are the distance values, and missing values will be set to not-a-number (NaN). Rows show *Dependent* values and columns show *independent* values. Each distance matrix heatmap's size is the product of the number of unique values in the *dependent* variable and the number of unique values in the *independent* variable for the current choice of subgroup and trend type. For example, as the subset table after selection in Figure 5.4 shows, there are two unique values of *dependent* variable (MPG and horsepower) and two unique values of *independent* variable (horsepower and acceleration), giving the distance matrix a size of 2×2. The corresponding distance matrix heatmap (Figure 5.4C) is a visual representation of the cells of that distance matrix with each cell representing a subgroup trend. The cells of the matrix are color-encoded to show distance values. Subgroup trends with higher distance are more saturated. In addition, a dark/light red color encodes a binary distance (i.e., 1 for a reverse

dependent	independent	splitby	subgroup	trend_type	distance
mpg	horsepower	cylinders	3	Linear Regression	0.2202895729
mpg	horsepower	cylinders	4	Linear Regression	0.044827793
mpg	horsepower	cylinders	5	Linear Regression	0.1408230328
mpg	horsepower	cylinders	6	Linear Regression	0.101901999
mpg	horsepower	cylinders	8	Linear Regression	0.0625959606
mpg	horsepower	model year	70	Linear Regression	0.0440369053
...	...	...	...	...	...

select A

dependent	independent	splitby	subgroup	trend_type	distance
mpg	horsepower	cylinders	6	Linear Regression	0.101901999
mpg	acceleration	cylinders	6	Linear Regression	0.9208861583
horsepower	acceleration	cylinders	6	Linear Regression	0.100538338

B reshape

	independent	
dependent	acceleration	horsepower
horsepower	0.100538338	NaN
mpg	0.9208861583	0.101901999

cylinders : 6

C visual encoding

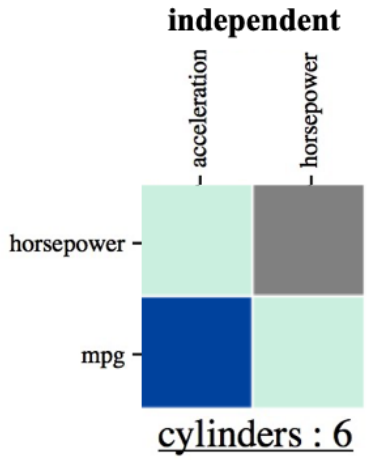


Figure 5.4: Internal pipeline to generate a distance matrix heatmap from a result table. (A) Records with the same trend type, splitby, and subgroup are selected to build a subset table. (B) Distance values are mapped into dependent variable rows and independent variable columns to form a distance matrix. (C) The matrix is visually encoded as a distance heatmap.

trend and 0 for a same trend) for a Pearson Correlation trend type. Grey indicates a cell with an absent (NaN) distance value.

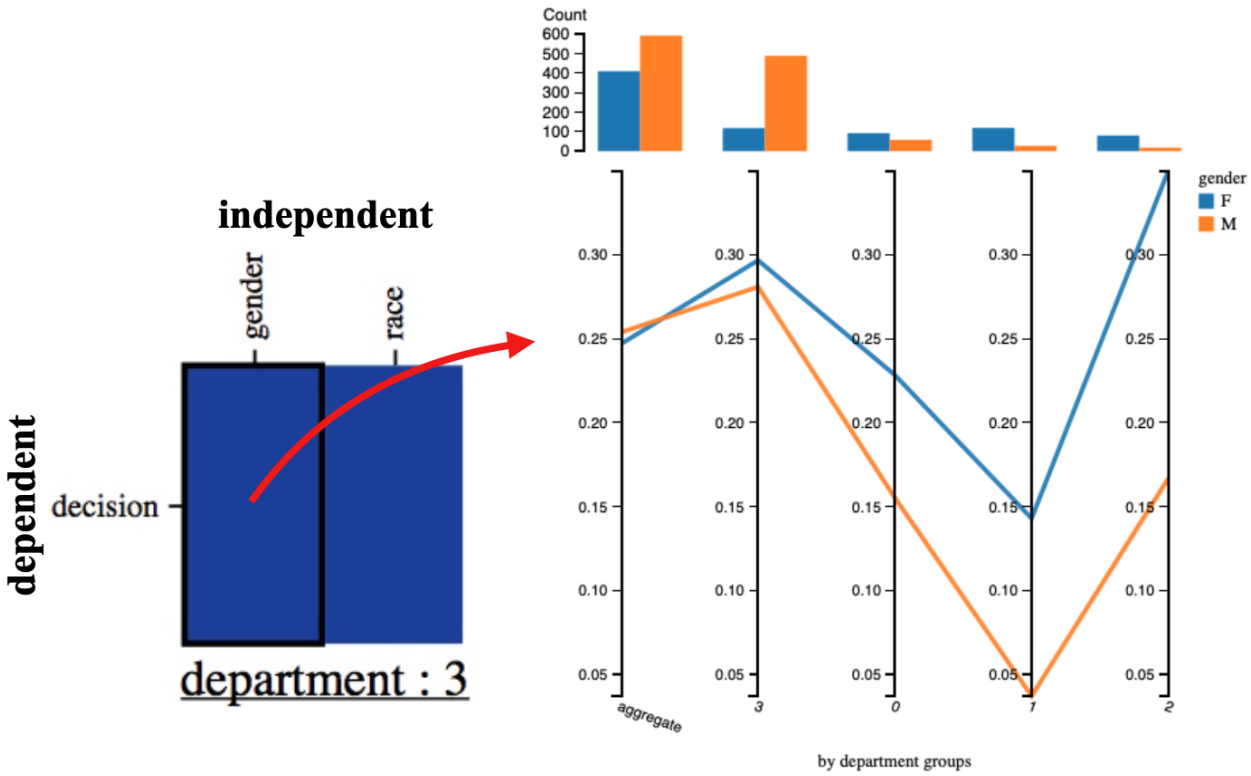


Figure 5.5: A screenshot of a distance heatmap and its detail view. The detail view shows a rank trend. When the user clicks a decision by gender cell in the distance heatmap, the parallel coordinates view shows a line for each gender through decision rates on each axis to represent a rank subgroup trend in the department 3 subgroup. A grouped bar chart provides details about the counts of each subgroup.

5.4.5 Detail Views

Wiggum includes detail views to help users investigate and understand relationships between aggregate and subgroup trends (**G4**). For each trend type, we choose different visual techniques to help explain various situations in the data. We describe the detail views corresponding to the two basic trend types, **rank trend** and **regression trend**, as follows.

The detail view for **rank trend** contains two sub-views: a grouped bar chart and a parallel coordinate plot. As shown in Figure 5.5, when users click a cell in the distance matrix heatmap, the cell is highlighted with a solid black border, and the detail view provides details on rates and counts for aggregate data and subgroups. A grouped bar chart at the top displays the counts of records for all combinations of the values in the independent variable

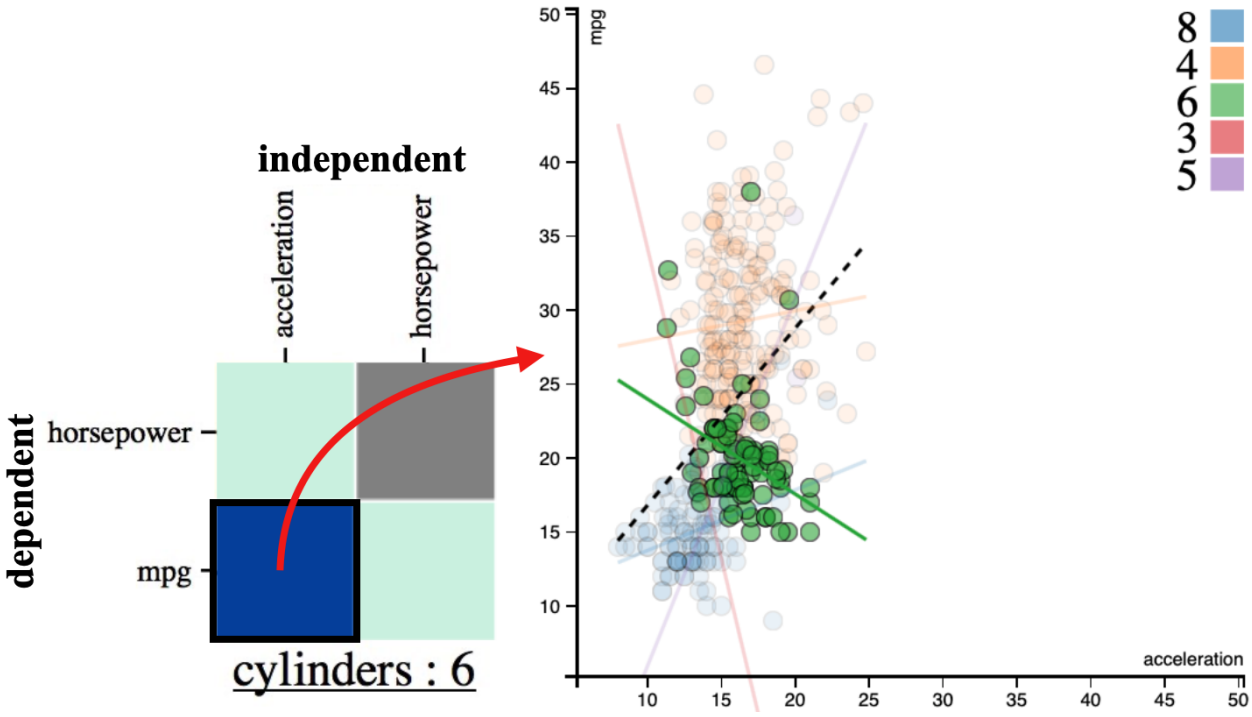


Figure 5.6: A screenshot of a distance heatmap and its detail view. The detail view shows a regression trend. When the user clicks the MPG by acceleration cell in the distance heatmap, the trend plot view shows the relationship between MPG and acceleration to represent a regression subgroup trend in the cylinders 6 subgroup. The legend in the trend plot is a list of the cylinder numbers.

and the splitby variable. Bars are grouped by position above the axis for the aggregate and each subgroup. Bars are color-encoded to represent each trend group in the trend group variable. The parallel coordinate plot allows comparison of the aggregate and subgroups to identify trend differences among them. The detailed information for the aggregate is always plotted on the leftmost axis, with the user-selected subgroup on the second axis. After users click a cell for another subgroup, the order of the vertical axis adjusts correspondingly. To better satisfy the need to observe the change of rates, each axis has the same scale. The colors of the connecting lines represent the trend group.

A trend plot—a scatterplot with trend enhancements—is a useful tool for observing relationships in bivariate data, and supports interpretation of the correlation coefficient or a linear regression model of a **regression trend**. As shown in Figure 5.6, the user selects a subgroup trend by clicking a cell in the distance matrix. Wiggum gets two columns of the

data from the column names, which are indicated by the cell's *dependent* and *independent* values, then plots all the data points in the trend plot. The vertical axis represents the *dependent* variable, and the horizontal axis represents the *independent* variable. Wiggum uses the *splitby* variable to color points in the trend plot. Regression lines for the selected subgroup and aggregate are plotted for users to observe their trends. A dotted black line represents the aggregate trend, and each colored line indicates the corresponding subgroup trend. The slope of the regression line indicates the direction of the relationship between dependent and independent variables. The positive slope of the aggregate trend and a negative slope of the subgroup trend in Figure 5.6 indicates a trend reversal. The degree of the slope allows users to track changes in trends even without a trend reversal.

At the top of the trend plot, Wiggum provides a checkbox for users to switch between two different axis range settings: same axis range and different axis ranges. The same axis range setting helps users more accurately observe slope and the angle between two regression lines. In contrast, angle and slope are not well-preserved when different axis ranges are used. It is sometimes preferable to use different axis ranges to have a finer view of the data when two variables have different ranges. In addition, the trend plot provides an interactive legend that indicates the subgroups in the *splitby* variable. Clicking legend cells selects subgroups and highlights their data.

5.4.6 The Control Panel Area

Wiggum provides a control panel to support efficient exploration of mix effects. The control panel allows **filtering**, **detecting**, **ranking**, **saving**, and **resetting** to support this process.

Users can **filter** the result table via the combo box at the top of the result table and the Filter button in the control panel (**G5**). Each combo box displays all unique values from the corresponding column. The result table view only keeps the rows with the same values based on the combo boxes' selections. Users can select multiple values from the combo boxes in multiple columns (i.e., *dependent*, *independent*, *splitby*, subgroup, and trend type) at the

same time. The distance heatmap view is updated based on the filtered result table. A Reset button restores the original rows of the table.

To start **detecting**, users can adjust the thresholds for subgroup trend strength, aggregate trend strength, and distance using the sliders in the result table. The table and heatmaps update correspondingly. Wiggum suggests default values for those thresholds. When users click the Detect button, the thresholds are sent to a filtering method in the data flow pipeline. By adding filtering by strength and distance, users are able to discover interesting patterns. For example, a distance threshold setting of 1 for a regression trend detects all reverse subgroup trends, since the distance method returns 1 for a reverse trend and 0 for a same trend.

Furthermore, Wiggum supports a comprehensive **ranking** function for users to examine the detected results. Wiggum provides three levels of ranking, as follows.

Viewpoint-level ranking. We define a viewpoint as a set of subgroup trends having the same *dependent* and *independent* variables. We refer to a pair of variables (x_1, x_2) that are used for computing trends as a viewpoint of the data, considering that two variables allow us to plot the data. To go deeper into the data, the next level of ranking is based on summary statistics by grouping the subgroup trends in the same viewpoint. Each viewpoint is ranked by the viewpoint aggregate score.

Colored Viewpoint-level ranking. In this level, we define a colored viewpoint as a set of subgroup trends having the same *dependent*, *independent*, and *splitby* variables. We refer to a 3-tuple of variables (x_1, x_2, x_3) that are used for computing subgroup trends as a colored viewpoint of the data, with encoding of the *splitby* variable (x_3) as a point color in the plot. The colored viewpoint-level measure of ranking can be obtained by aggregating over all subgroup trends within the same colored viewpoint. Users can choose which aggregation method (sum, mean, max, min) on the distance column to use to compute a colored viewpoint score. The options for colored viewpoint and viewpoint ranking is shown by clicking the select button. The colored viewpoint aggregate scores are added to the result table and used to

rank each colored viewpoint.

Subgroup-trend-level ranking. In the analyzing stage, users are shown a result table comprising potential reverse subgroup trends. Each row represents a subgroup trend. Each subgroup trend has four elements (x_1, x_2, x_3, y), in which x_1 represents the *dependent* variable, x_2 is the *independent* variable, x_3 is the *splitby* variable, and y represents subgroup value. A user can rank the subgroup trends by clicking the header of the distance column in the result table, in which case the ranks of all of the subgroup trends are based on the results of a descending sort on distance in the result table.

Moreover, Wiggum provides users with a button for saving the original data, the metadata, and the result table. In the preparation page, users can load prior saved files into Wiggum from a chosen folder. This approach can avoid repeating the process of data labeling, and save time by reusing a result table instead of computing it again in the data flow pipeline.

Wiggum also allows users to retrieve the initial result by clicking a Reset button. After resetting is triggered, the result table is set to the original result table without any filtering, detecting, or ranking. The distance heatmap view is redrawn from the original result table.

5.4.7 Implementation

The Wiggum system is a web app. It uses D3 [22] for visualization and a Flask [3] server to connect the Wiggum Python library to JavaScript-powered visual analytics in the browser. The Python Wiggum library includes all of the computational features and runs as a backend server to the Wiggum app. Wiggum has been designed as a modular framework, allowing for each individual component to be modified or extended.

5.5 Summary

In this chapter, we present Wiggum, an integrated system for detecting, ranking, and visualizing mix effects and Simpson's paradox. A distance heatmap view provides an overview of subgroup trends and offers clues to find potentially important ones. Trend plots and parallel coordinate views support detailed examination of mix effects and Simpson's paradox. Coordination of multiple views through a variety of supporting interactions helps users discover and explore patterns related to trend reversal. Users can filter, rank, and compare subgroup trends on trend strength and distance to eliminate spurious subgroup trends.

Chapter 6

Evaluation of Wiggum

6.1 Overview

Through use cases, we describe how Wiggum supports the discovery of mix effects in three real data sets and demonstrate how a combination of visualization techniques—heatmaps, trend plots, small multiples, coordinated multiple views, and dynamic queries for multi-attribute drill-down—are effective for analyzing mix effects. We conducted a user study to evaluate Wiggum, focusing on users' ability to comprehend the statistical concepts, identify the corresponding visual patterns, and perform common analysis tasks correctly and efficiently. We discuss usability issues, utility limitations, and outline future directions to improve Wiggum.

6.2 Use Cases

In this section, we describe how Wiggum can be applied in practice to two real-world data sets from the UCI Machine Learning repository [90] and a famous example of Simpson's paradox [18]. The first use case shows how Wiggum can be applied to detect trend reversal in partitioned data for linear regression analysis of multidimensional data. The second use case explores a binary ranking of rates which ranks two groups through a known example of

Simpson's paradox from the UC Berkeley admissions data set. The third scenario highlights how Wiggum can be generalized to a more advanced usage that detects mix effects relative to trends based on both binary and multiple ranking cases. Multiple ranking analysis considers rank trend with respect to the rank of three or more groups. Table 6.1 summarizes the domain, trend type, the number of the records, and the number of the attributes for each use case.

Table 6.1: Summary of the three use cases.

Case	Domain	Type	#Records	#Attributes
1	Auto MPG	Regression	392	9
2	Graduate Admissions	Binary Rank	12	4
3	Adult Income	Multiple Rank	32,561	8

6.2.1 Regression Analysis

In the first use case, we demonstrate how Wiggum can be used to discover mix effects in a regression trend type data set. We apply Wiggum to the Auto MPG data set. After removing rows with missing data, 392 records each represent a car model with 9 attributes: *MPG, cylinders, displacement, horsepower, weight, acceleration, model year, origin, and car name.* Data preparation begins after loading the data. The *MPG* attribute is set to *continuous* type and *dependent* role. The *horsepower* attribute is set to *continuous* type and role as both *dependent* and *independent*. The *acceleration* attribute is set to *continuous* type and *independent* role. The other (discrete, multi-valued) attributes are set to *categorical* type and *splitby* role. The *displacement*, *weight*, and *car name* attributes' roles are set to *ignore* to exclude them from trend computation in regression calculations. To study the relationship between two continuous variables in the data, we also set the trend type to *Linear Regression*.

Upon clicking the *Visualize Trends* button, Wiggum switches to the visualization page. The result table view shows 63 rows, one for each combination of 21 subgroups and 3 *splitby* variables. The distance heatmap view displays a heatmap for each of the 21 subgroups and

chosen *splitby* variable. Each heatmap shows a 2x2 matrix corresponding to the chosen two *dependent* variables and two *independent* variables. Inspecting the distance heatmap view immediately reveals some surprising patterns. Two of the 13 heatmaps for model year subgroups contain dark blue cells located in the same cell position, indicating a relationship between *MPG* and *acceleration*. Clicking the dark blue cell in the heatmap of model year 75, we see a positive relationship between *MPG* and *acceleration* for all data points. One might conclude that higher *MPG* corresponds to higher *acceleration*, yet this inference is not supported in the dis-aggregated data. A negative relationship is obvious in model years 75 and 79. The inconsistency of patterns between aggregate data and subgroups may lead a user to hesitate to draw conclusions. Furthermore, we observe an interesting pattern in the heatmap for the 6 cylinders subgroup, as shown in Figure 5.3, in which the relationship between *MPG* and *acceleration* shows a trend reversal. This finding could influence the decision making process for a car consumer who considers a high MPG and high acceleration car. Once they recognize that MPG will decrease as acceleration increases for the cars with 6 cylinders, other cylinder models tend to be more attractive to them.

There are more surprising trend reversals in the Auto data set that we can discover and study through Wiggum. Although space precludes describing more of them here, the example above is evidence of Wiggum's capability for both detecting and explaining mix effects in linear regression analysis.

6.2.2 Binary Ranking Analysis

We illustrate how Wiggum can support visual data exploration through the gender bias case in the UC Berkeley graduate admissions data set. A study of graduate admissions to the University of California, Berkeley [18] was conducted to investigate gender bias. The binary ranking analysis is based on the ranking of two subgroups, men and women. The admission rates for Fall 1973 showed that 44% of men and 35% of women were accepted overall, yet in the data for the six largest departments, four departments show a lower acceptance rate for

men than women.

The graduation admissions data contains 12 records with 4 attributes: *department, gender, number of applicants, and rate of admission*. After loading the data set, we annotated data types and roles for each variable in the data configuration page. Since we wanted to study the ranking of gender in admission rate, we chose the *rank trend* type, then clicked the *Visualize Trends* button. In the visualization page, each of the six 1×1 distance heatmaps (Fig. 6.1A) represents one department. The dark blue cell indicates a high trend distance between the aggregate data and the subgroup data. After clicking a cell in the distance heatmap for Department A, the detail view (Fig. 6.1B) displays the corresponding grouped bar chart and parallel coordinate plot for examining rank trend type. In the parallel coordinate plot, the first axis shows that men have a higher admission rate than women in the aggregate data (F:35% vs. M:44%). Nevertheless, the second axis reveals a reverse trend in which the admission rate for women exceeds that of men in Department A (M:62% vs. F:82%). Since the relative rank of the genders by admission rate is flipped (from [*Women, Men*] to [*Men, Women*]), the clicked cell in the distance heatmap is color-encoded dark blue. The parallel coordinate plot represents the trend distance by visualizing the rankings for gender on each axis. The grouped bar chart shows the gender distribution in aggregate and in each department. The Berkeley admission case clearly illustrates mix effects, since four out of six departments have the reverse trend.

6.2.3 Multiple Ranking Analysis

The adult data set [90] contains 32,561 records with 8 attributes: *workclass, education, marital-status, occupation, relationship, race, sex, and income*. A multiple ranking analysis can be applied to the ranking based on any attribute (e.g., race) with more than two values (e.g., White, Black, Asian-Pac-Islander). In this use case, we aim to find surprising trends in the rate of people making more than 50K per year. We again load the data file and select variable types and roles. All attributes except *income* are set to *categorical* type and both



Figure 6.1: The visualization page for the UC Berkeley Graduate Admissions data set. After clicking a dark blue cell in the distance heatmap (A) for department A, the detail view (B) displays the corresponding grouped bar chart and parallel coordinate plot for examining the trend reversal between the aggregate data and subgroups.

independent and *splitby* roles. Thus, those 7 attributes are either the *independent* or *splitby* variable of a subgroup trend. The *income* attribute is a binary attribute containing two values: $>50K$ and $<=50K$. We set its type to *binary* and role to *dependent*, making it the *dependent* variable of a subgroup trend. Since we use this data set to study the trend based on the rate of adults' income exceeding 50K per year, we set the trend type to *rank trend*.

On the visualization page (Figure 6.2) Wiggum displays 9 out of 60 distance heatmaps, each representing a subgroup. Each heatmap has one row (for the *dependent* variable) and six columns (for the *independent* variables), with the *splitby* variables used to partition the data. In the result table view, there are 360 records (one *dependent* $\times$ six *independent* $\times$ 60 subgroups). Each one represents a subgroup trend corresponding to a cell in the heatmaps.

The blue cells in the distance heatmap view are mostly related to race and sex. Focusing on the darkest ones, we adjust the distance slider to 0.9 and both strength sliders to 0. Clicking the *detect* button leaves only three subgroup trends in the result table. Meanwhile,

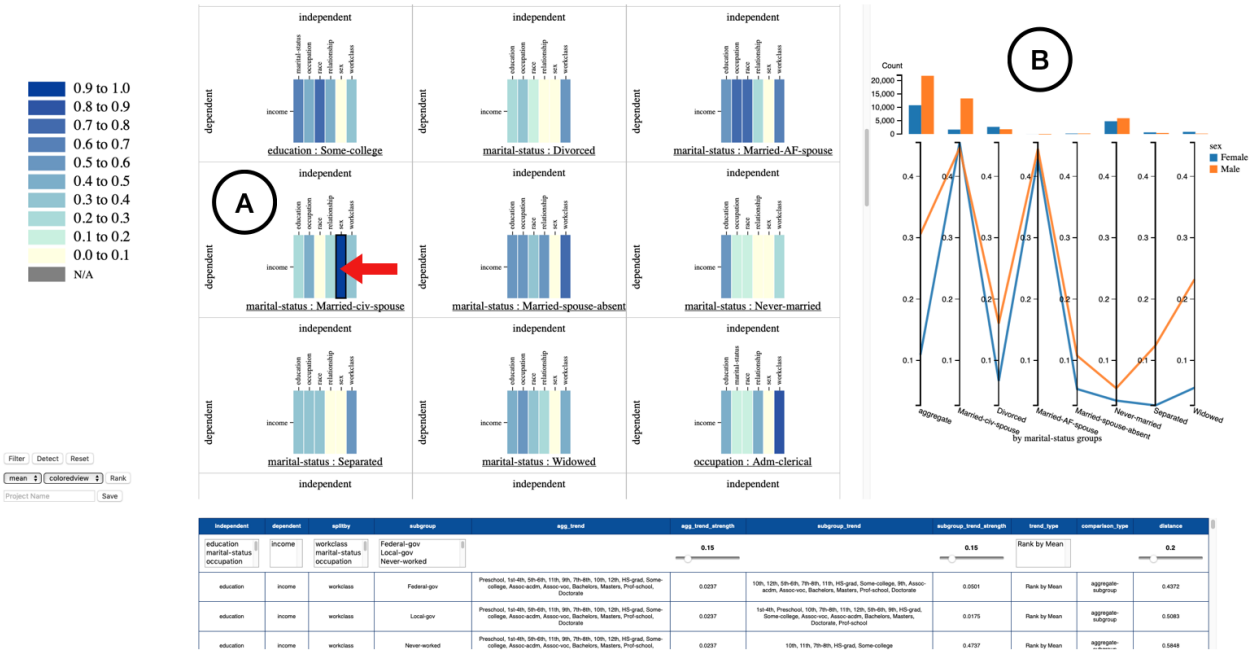


Figure 6.2: The visualization page for the adult data set. The distance heatmap view displays 9 out of 60 1×6 distance heatmaps; vertical scrolling reveals the others. After clicking a cell in the distance heatmap for a subgroup (A), the detail view displays the corresponding grouped bar chart and parallel coordinate plot for examining rank trend type (B).

we see three 1×1 distance matrix heatmaps in the overview area. Several interesting observations arise upon checking the detail view of the three subgroup trends. For instance, in the *Married-civ-spouse* subgroup, we can observe a reverse trend in terms of *sex*. The trend in the aggregate data is that the rate of over 50K annually is higher for males (30.6%) than females (10.9%). After dividing the data set based on *marital-status*, we see there is a reverse trend with females (45.5%) at a higher rate than males (44.6%) in the *Married-civ-spouse* subgroup.

In addition to a binary ranking, we observe another interesting case in the detail view, in which the trends are seen among different racial groups based on the rate of over 50K income. We select *race* in the *independent* menu selection and pick *occupation* in the *splitby* menu selection, then click the *Filter* button. Looking within racial groups, the rate of income over 50K among *Asian-Pac-Islander* is 26.6%, followed by *White* (25.6%), *Black* (12.4%), *Amer-Indian-Eskimo* (11.6%) and *Other* (9.2%). However, *Asian-Pac-Islander*

is not always the highest income earning race. Clicking the cell belonging to the *Exec-managerial* subgroup partitioned by the *occupation* variable reveals that *White* experiences the highest rate (49.9%) of income over 50K, followed by *Asian-Pac-Islander* (45.2%), *Black* (34.4%), *Other* (18.2%) and *Amer-Indian-Eskimo* (10%). Similarly interesting patterns exist and can be readily explored for each occupation's subgroups.

This example demonstrates the capability to explore trends in both binary and multiple ranking cases. A more comprehensive understanding of the data emerges from observing how surprising ranking trends of subgroups are inconsistent with aggregate ranking trends.

6.3 Evaluation

We conducted a formal user study to evaluate how well Wiggum helps target users detect and understand mix effects, and also to solicit ideas for enhancement motivated by their individual data analysis wants and needs. In this study, we focused specifically on detection of reverse regression trends using Pearson correlation. The study was approved by the Internal Review Board (IRB) at the University of Oklahoma.

6.3.1 Participants

We did a pilot study with 2 participants—one library staff member with a biology background, one undergraduate in computer engineering—in order to test feasibility, duration, and improve upon the study design. For the main study, we recruited 37 other participants: 14 Ph.D. students, 8 faculty members, 8 M.S. students, 4 staff members (3 library, 1 senior research associate), 2 postdocs, and 1 undergraduate student. Participants' majors or current primary academic disciplines included physics, economics, geography, meteorology, computer science, psychology, data science, library information science, geology, biology, mathematics, and management information systems. All participants had at least basic knowledge of statistics, such as linear regression or Pearson correlation. Most participants

Table 6.2: The predefined roles—dependent, independent and splitby—of variables, the number of distance heatmaps, and their internal dimensions generated in the user study.

Data Set	Size	Dependent	Independent	Splitby	Distance Heatmap	
					Num.	Dimensions
Iris	150	sepal length, sepal width	sepal length, petal length, petal width	class	3	2×3
Auto	392	MPG, acceleration	displacement, horsepower, weight, acceleration	cylinders, model year, origin	21	2×4
Facebook	495	page total likes, reach, consumers, num_comments, num_shares	page total likes, consumers, num_comments, num_likes, num_shares	type, category, month, weekday, hour, paid	44	5×5

took at least one full semester statistics course. Participants were not compensated.

6.3.2 Procedure

We conducted the study with participants one-on-one via Zoom video conferencing. With the consent of all participants, we recorded audio and video with screen sharing. Participants were encouraged to think aloud and to ask questions. The procedure consisted of the following steps: an *introduction*, to give a quick overview and explain the purpose of the study; a *questionnaire*, to gather demographic and educational background details; a *knowledge assessment*, to evaluate participants' knowledge of mix effects and associated concepts; a six minute *video tutorial*, to give an overview of Wiggum and demonstrate its features; performance of a set of data exploration *tasks* (within-subjects) to assess utility and identify usability issues with Wiggum; and a *post-survey*, to gather subjective feedback about its design and features. We set up a local web server running Wiggum. Participant access was remote over public internet through a secure tunnel (*ngrok* [4]).

6.3.3 Data and Tasks

Our study employed three open-source data sets available from the UCI machine learning repository [90]: Iris, Auto, and Facebook (short for *Facebook metrics*). We chose these data sets because they give the exploration tasks three levels of complexity: easy, medium,

Table 6.3: Tasks and questions in the user study.

Task	**Question**	**Task Score Rubrics**
Identify reverse trends in heatmaps	T1_Q1: How many reverse trends do you find? T1_Q2: Describe a reverse trend.	Score = average correctness per question (0.5 partial credit for simple miscount)
Identify subgroup or aggregate trends (for two different subgroups and pairs of variables: Q1–Q4 and Q5–Q8)	T2_Q1/Q5: What is the trend between variables for the aggregate data? T2_Q2/Q6: What is the trend between variables for the subgroup? T2_Q3/Q7: Are the two trends the reverse or the same? T2_Q4/Q8: What are the other subgroups' trends between the variables?	Score = average correctness per question
Confirm Simpson's paradox or mix effects	T3_Q1: How many instances of Simpson's paradox? T3_Q2: Describe one instance of Simpson's paradox. T3_Q3: How many instances of mix effects? T3_Q4: Describe one instance of mix effects.	Score = average correctness per question (0.5 partial credit for simple miscount)

and hard. The levels of complexity are driven by the number and dimensions of distance heatmaps. For example, the easy data set (Iris) generates three 2×3 distance heatmaps, whereas the hard data set (Facebook) generates 44 5×5 distance heatmaps. We preset the roles for all variables (see Table 6.2), allowing participants to get started with minimal data preparation. After loading the data, the participants chose *Pearson Correlation* for the trend type, then clicked the *Visualize Trends* button. The number and dimensionality of distance heatmaps vary from data set to data set. (Some dimensions have too few records or attribute values to include. Heatmaps are not generated for such dimensions.)

Amar, et al. describe ten kinds of low-level analysis tasks used to evaluate the design of visualizations for data understanding [9]. We focus on tasks that involve retrieving values, finding anomalies, and assessing correlations. As our objective was to observe how Wiggum helps target users detect and understand mix effects, we designed tasks to capture an interaction process that includes input, output, and analysis steps [85]. We designed three tasks to afford participants an opportunity to gradually learn key concepts needed to understand mix effects: subgroup trend vs. aggregate trend, same trend vs. reverse trend, and Simpson's paradox vs. mix effects. Table 6.3 summarizes the goals, questions asked, and scoring rubrics applied for each task.

6.3.4 Correctness

Figure 6.3 shows the number of participants who answered correctly for each task-question-dataset combination. For correctness, scores were binary. Of the 37 participants, the majority (Iris: 26/37; Auto: 28/37; Facebook: 24/37) were able to identify all reverse trends (T1_Q1). For the Iris data set, most incorrect answers (8/11) arose from misconception about how reverse trends are represented in heatmaps; in those cases participants interpreted cells in the same position in heatmaps as a single reverse trend. For the other two data sets, most of the participants who gave incorrect answers (Auto: 6/9; Facebook: 8/13) miscounted reverse trends due to difficulty counting over the increased number of heatmaps and the need to scroll over them. Participants performed well on most questions when asked to identify a trend by giving specified information (T2_Q1–Q8). For the Facebook data set, participants performed poorly on T2_Q4 and T2_Q8 due to the large number of subgroups and their trends. In T3, the Iris data set exhibits only Simpson's paradox, and the Auto and Facebook data sets exhibit only mix effects. Participant responses offer insight even without a data set that exhibits both Simpson' paradox and mix effects. When asked to

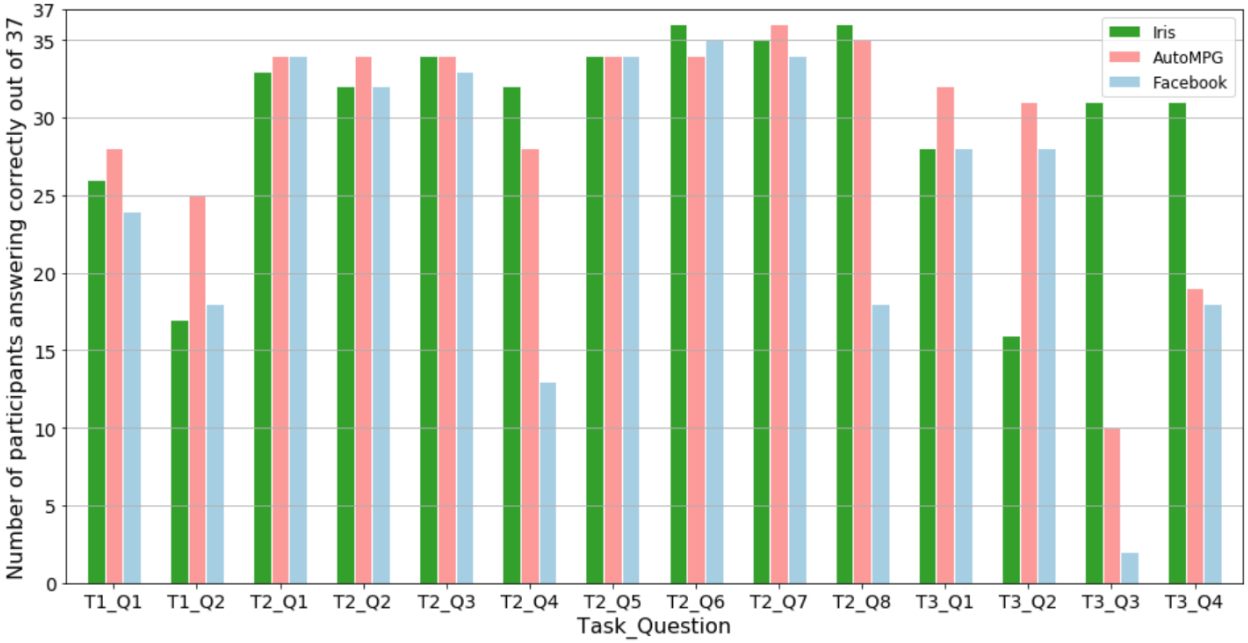


Figure 6.3: Correct answers for each task-question-dataset combination.

confirm mix effects (T3_Q3), the number of correct answers declined sharply between data sets (Auto: 10/37; Facebook: 2/37). This suggests that data set size substantially degrades (and ultimately limits) one's ability to visually detect mix effects in the Wiggum design. Evaluation on a wider range of data set sizes is needed to assess this hypothesis.

6.3.5 Time vs. Task Score

The average session duration was 109.05 minutes (s.d.=29.64). Mean completion times for T1 in the Iris, Auto, and Facebook data sets were 7.81 (s.d.=4.38), 6.92 (s.d.=3.62), and 5.95 (s.d.=2.40) minutes, respectively. For T2 these were 6.30 (s.d.=3.53), 5.70 (s.d.=2.36), and 5.92 (s.d.=3.24) minutes. For T3 these were 7.19 (s.d.=4.32), 7.19 (s.d.=3.37), and 9.73 (s.d.=6.73) minutes. For timing, T1 and T3 scores could earn half credit, to account for the prevalence of mental addition errors even when participants counted items correctly (see Table 6.3). Half credit produced the distribution of fractional average task scores shown in Figure 6.4. Task scores (color) are widely distributed over task durations (vertical position). They are only weakly correlated in a few of the nine task-dataset combinations, such as higher scores on T1 taking longer particularly on the Iris data set, and much longer times to correctly count all mix effects in T3 on the Facebook data set. The former suggests that some participants were still learning Wiggum, with more or different training possibly needed. The latter suggests that task duration is closely coupled with counting effort, and thus that adding counting aids to the Wiggum design, such as some kind of selection highlighting to remember counted items, could significantly increase both efficiency and correctness of task performance.

6.3.6 Qualitative Results

We administered a post-survey, asking participants to rate Wiggum on four aspects. Average Likert ratings were high all around on a 5-point scale: easy to learn (3.9/5.0, s.d.= 0.66), easy to use (3.9/5.0, s.d.= 0.85), fast to use (4.1/5, s.d.= 0.95), and comfortable to perform tasks

(3.9/5.0, s.d.= 0.97). While this feedback is encouraging, it indicates no obvious directions for design or feature improvement.

Some participants clearly found Wiggum helpful for learning mix effects. P10 wrote: *"I feel like after using the tool I have a more practical sense of what the paradox looks like, and the formal definition makes more sense to me now that I've seen examples and can put it in my own words."* Similarly, P9 commented, *"seeing it visually helped my understanding of the concept."* Unfortunately, learning to interpret a new type of visualization is not an easy task, and participants often struggled to identify instances of mix effects. Some participants were not able to understand the transition between heatmaps and trend plot, and didn't realize that clicking on the same cell position in any heatmap of a given *splitby* variable highlights different subgroups in the same trend plot. In the distance heatmap view, we also observed that participants were hindered by how Wiggum shows all distance heatmaps together in one scrolling layout rather than grouping them by their *splitby* variables. P15 wrote: *"Looking where a particular category ends and another begins was difficult too."*

We found the distance heatmap view to help participants' awareness of different patterns. In general, participants enjoyed using it. P33 commented, *"I really liked the heat maps, and*

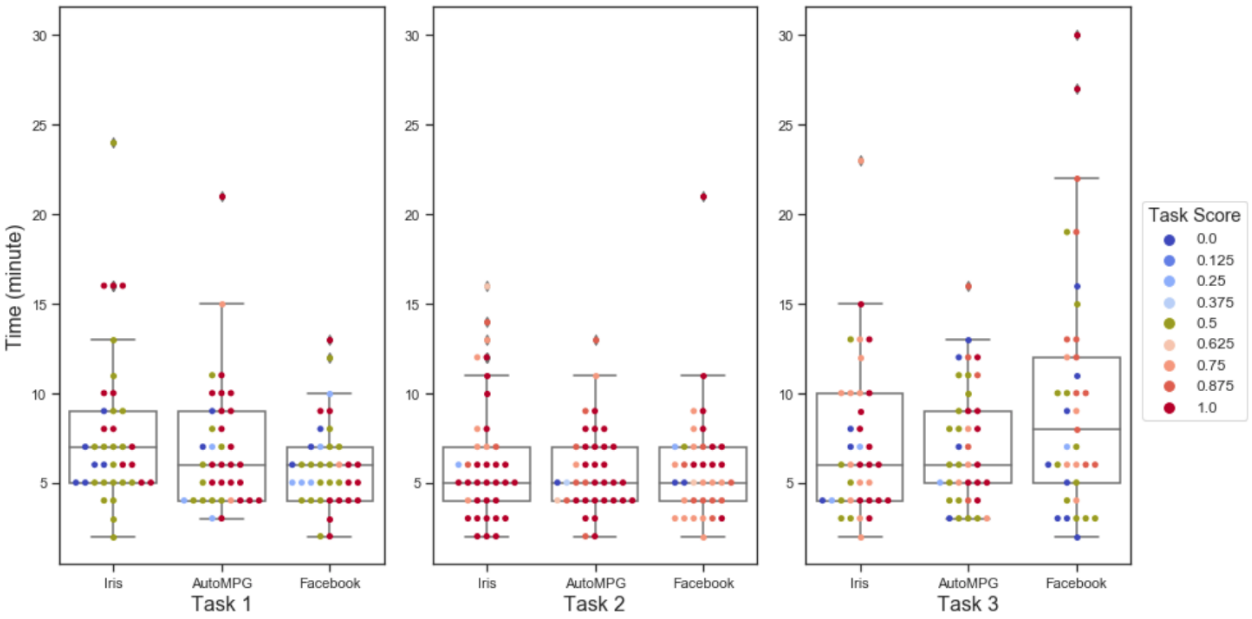


Figure 6.4: Time vs. task score in each task level for the different data sets.

the fact that all subgroup trends for a particular relationship were shown at once." P22 said, *"I preferred to use the heat maps for detecting Simpson's paradox, especially for the larger data sets, because the trendlines [in the trend plot] were difficult to see for larger sets."* However, some participants noted how working with larger data was a struggle. P29 wrote, *"The volume of data increased the difficulty to look and see if there was a Simpson's paradox."* P20 said, *"When there was a great number of variables to address in the Facebook data set, that feeling of comfort with the interface started to decline."* Participants sometimes overlooked the scrollbar, which hides itself in some browsers, thus missing some heatmaps, or accidentally zoomed in on individual heatmaps when trying to scroll.

Participants especially enjoyed the overall ease and efficiency of using Wiggum. P3 said, *"I liked it. It was easy to use and understand what was going on/what I was doing."* P24 pointed out that *"Visual feedback showing selected data sets and trendlines was very quick, essentially instantaneous."* Similarly, P16 commented: *"It was simple to use. It definitely helps you look at data in an easy manner."*

6.4 Limitations and Future Work

In this section, we discuss limitations of the current version of Wiggum and identify opportunities for future development.

6.4.1 Trends Types and Comparisons

Wiggum is currently designed for rank and regression trends (**R2**), but can be extended to other statistical measures, such as for performance of a binary classification task. For example, the difference between precision for an entire population and conditioned on a specific subgroup ($\frac{TP}{TP+FP}$) is a promising measure to support fairness forensics [40, 26, 72, 48]. Beyond the brute force discovery of clusters currently implemented in Wiggum, future work on more flexible proxy techniques to recover the omitted *splitby* variable could enable

Wiggum to further support fairness forensics. In addition, Wiggum's current support for comparison between aggregate and subgroup trends could be extended to interesting and practical comparison between subgroups themselves.

6.4.2 Visualization Design

Participants performed poorly at identifying and counting mix effects in larger data sets. Techniques to increase visual scalability for larger and more complex data sets (**R1**) is a clear direction for improvement. In practice, we find it helps to report summary information such as the number of mix effects. In future work we will explore more guided (less ad hoc) approaches to selecting dimensional subsets. It can also be difficult to differentiate the subgroups of different *splitby* variables in the current heatmap layout. This could be addressed by more clearly organizing the heatmaps by their subgroups. The heatmaps also become denser as rows and columns increase in number. An overview heatmap over all subgroups might help with checking trends in individual heatmaps. The same issue arises in the parallel coordinate plot as the number of distinct values of the *splitby* variable increases. Although parallel coordinates can help users observe changes of ranking between aggregate and subgroup data, the distance and strength of a subgroup trend are relatively hard to read visually. Alternative visualization techniques could support more precise reading and hence easier interpretation of those statistical measurements. (The bump chart [142] is a candidate, but it poorly shows relative rate difference, which is crucial for rank comparison.) In the trend plot, the angle between two regression lines is hard to interpret and compare across axis scale changes, which happens when users select a different cell in a heatmap. We plan to explore visual techniques to help users read angles; simple radial plots may be a promising option for this task.

6.4.3 Causality

While Wiggum provides a means to detect and explain mix effects and Simpson's paradox statistically, what to infer from them visually is not always clear (**R3**). Causal inference may provide a practical perspective from which to explain them. Incorporating ways to visualize the network of causal relationships among variables might prove highly beneficial for deeper sense-making.

6.4.4 Scalability

Moving forward, we will investigate the scalability of computation and visualization in Wiggum in terms of data size and dimensionality (**R1**), particularly in terms of the number of independent, dependent, and splitby variables. We conducted a preliminary study that produced three findings.

First, varying data sizes doesn't slow down the computation significantly. Second, as the number of dependent/independent variables increases, we see a trend showing a quadratic increase in the running time. Lastly, as the number of total subgroups increases, there exists a trend showing a linear increase in the running time. The current visualization design (see Section 5.4) anticipated and addressed some scalability issues. Moreover, filtering low correlation pairs of dependent and independent variables after the computation of aggregate data can alleviate visual complexity by reducing the number of rows and columns in the heatmap view. As the number of subgroups increases, Wiggum might also display only the top-k heatmaps that have the highest average distance scores. Follow-up studies could be conducted to investigate multiple optimization methods with Wiggum with the goal of supporting data analysis at larger scales.

6.5 Summary

This chapter describes three use cases for evaluating the utility of Wiggum. Wiggum appears effective for supporting deep visual analysis of multiple trend types in real-world data sets. We conducted a user study to learn users' ability to comprehend the statistical concepts, identify the corresponding visual patterns, and perform common analysis tasks correctly and efficiently. The user study revealed usability issues that motivate design improvements as well as analysis features for ongoing development and study.

Chapter 7

A Model of Visual Correspondence

7.1 Overview

Well-designed visualizations can lead to effective and efficient exploratory data analysis. However, the meaning of data can be ambiguous as its interpretation depends on what kind of information in it is being conveyed. There are different pieces of information we may take from data. For instance, if we have a quantitative data value, we might look at it in terms of whether it is positive or negative, or whether it is high or low relative to some reference value, or the amount of precision in the data value. Additionally, data items with the same interpretation could have multiple visual representations, and such representations might be used for looking at the values in different ways simultaneously. As the ambiguity of data and the complexity of visualizations continue to grow, identifying corresponding graphical items becomes challenging. We define *visual correspondence* as the extent to which visual encodings of the same or equivalent information are interpreted as being the same or equivalent. Current visualization theories [17, 37, 94] focus on determining a visual channel for a single encoding of a graphical item. In contrast, our work explores the pairwise effect of visual channels when visual correspondence is considered.

It would be useful to develop a conceptual model for visual correspondence that visual

designers can apply in the visualization design process. Visualizations often consist of multiple views. Visual correspondence provides an implicit and passive way to bridge from view to view with a smooth transition. Visual linking, which provides explicit continuity and connectivity through connecting lines, is a workaround for weak correspondence. Visual links should minimize visual interference and the occlusion of important information [134]. When it is hard to achieve the above goals, visual correspondence can be an alternative way to associate related graphical items. Also, strong visual correspondence could strengthen the effectiveness of visual encodings for identification and comparison tasks. Our theory makes predictions about design, experience, and effectiveness, when the identity of items throughout a visualization is a key factor in performing tasks.

Visual correspondence in general is about figuring out how to make a set of same or equivalent things be seen as related to each other, but unrelated to all the others. Gestalt theory helps us to figure out which combinations of visual channels may or may not work for creating visual correspondence. We derive our conceptual model from some of the more relevant principles of Gestalt theory, including alignment, proximity, and similarity.

We develop a complete theory that tells the visualization designer what to do when they have two different visual representations for the same data item with any possible set of data cardinalities (number of unique possible values of the item). Our conceptual model measures the degree of correspondence in terms of four factors: aspects of information, **a pair** of visual channels, and cardinality. A general method applies all possible combinations of these four factors to measure visual correspondence. A specific method for populating the model applies three cases: a simple case, an intermediate case, and a complex case. Our contributions are as follows:

- a **model** of visual correspondence,
- an assessment of the model for **visual encoding pairs** in common configurations,
- a **comparative analysis** of the model in application to a variety of example visual-

izations,

- a set of **guidelines** for the design and evaluation of visual correspondence in practice, and
- a sketch of a **research program** to develop **foundational mathematics** for analyzing tasks in terms of the effects of visual correspondence on item identification and comparison.

Together, these contributions provide a basis for designing visualizations that account for effective visual correspondence.

7.2 A Model of Visual Correspondence

Visual correspondence is the extent to which different graphical items can be associated when they encode the same data value. Visual correspondence builds on existing knowledge and practice of encoding channels for visualizing data. In Figure 7.1a, a number and a dot, which encode the same data as vertical position in a single view, can correspond to each other. Similarly, the elements in juxtaposed views or nested views could activate visual correspondence for revealing the related data items. In Figure 7.1b, horizontal position in the bar chart and shape in the scatterplot represent the same categorical data items. In Figure 7.1c, leaf nodes encode data as vertical position, and bars encode the same data as horizontal position. This section presents a conceptual model of *visual correspondence*. We first describe the factors contributing to the structure of the space for our model. We then describe the model itself.

7.2.1 Information Aspects

Representations of information always involve interpretation. A single data value can exhibit different types of information depending on how it is interpreted. Numbers represent quan-

tities; meanwhile, they have an order and can also be thought of as nominal. For example, a position along a path in a national park, in which the actual position is a quantity of walking miles, can also have ordinal characteristics (one landmark comes before another landmark) as well as nominal characteristics (specific landmarks). The quantitative characteristic can be less important than an ordinal or nominal characteristic when considering how to design a visualization.

We focus on multidimensional data which are structured into a table. A value in each cell of the table represents the information about an attribute of a record. A broader categorization interprets the value as one of three types of information: nominal, ordinal, or quantitative [94]. However, the information of an attribute is not always confined to a single type. When an attribute can be interpreted as multiple types, we refer each such interpretation as an aspect of the information. We call an aspect "type-like", for instance, nominal-like or quantitative-like. Our model considers visual encodings of data in terms of such interpretable aspects of the information rather than their overt information types.

We define a *visual object* as a minimal graphical object which applies a visual channel to encode a data value in a graphical item. For instance, a visual object can be a color-filled circle, or the edge of a circle that uses its width to encode data. Our goal is to learn how visual objects correspond to each other. To achieve this, we first need to understand

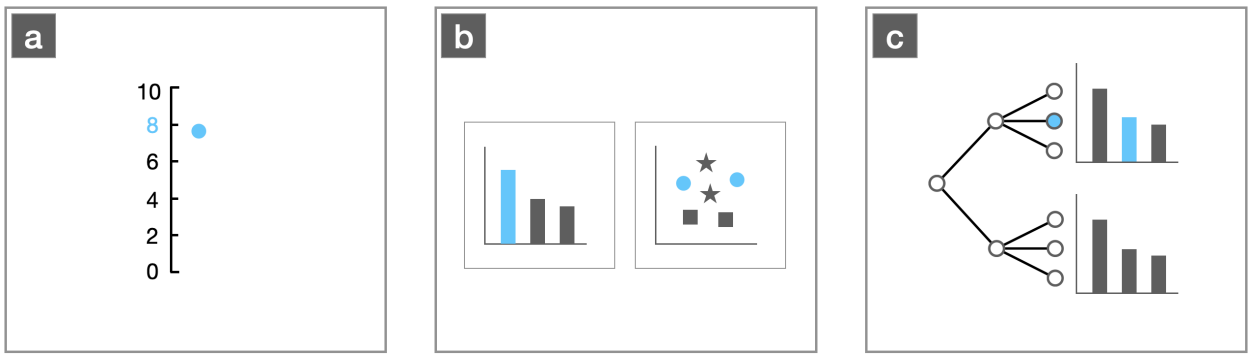


Figure 7.1: Three basic visual designs for considering visual correspondence: (a) a single view, (b) juxtaposed views, and (c) nested views. Color is used here for illustration only; without the strong visual correspondence provided by associated colors, the overall visual correspondence between each item's pair of visual encodings would be weaker.

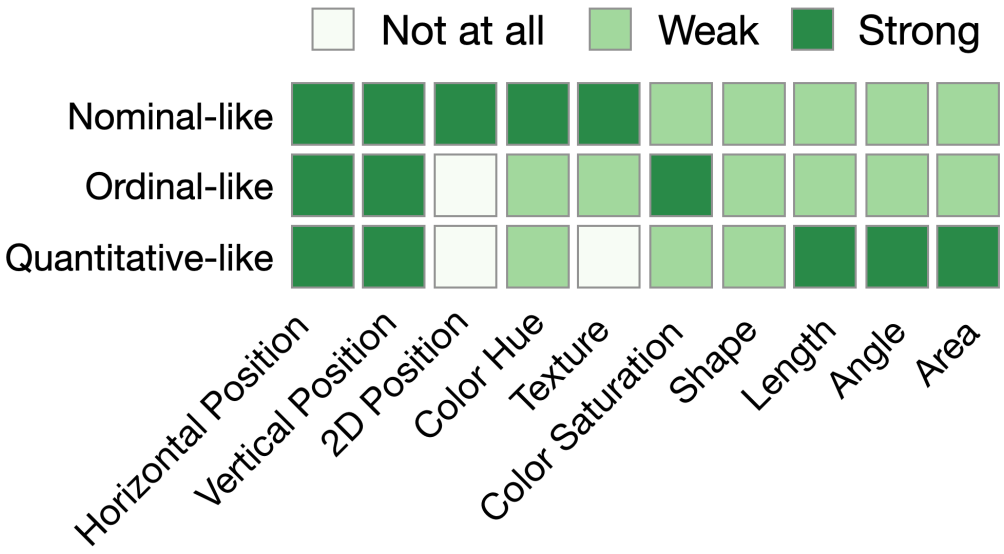


Figure 7.2: Three-level degree of likeness for three different information aspects over common visual encoding channels.

the correspondence between a data item and its visual object. Mackinlay's [94] Automatic Presentation Tool (APT) uses the ranking of perceptual tasks by information type to express the effectiveness of visual channels. The effectiveness implies the correspondence between a data item and a visual object that encodes it. We use three levels (i.e., Not at all, Weak, Strong) to rank pairs of visual channels between visual objects for the three different aspects of data considered in Mackinlay's rankings. The coarse level of ranking simplifies the methods for applying our model, and helps to verify our model and methods in the initial effort presented here.

7.2.2 Visual Channels

For a given data value, which visual channel should we choose to represent it? Part of the difficulty is that any given representation, visual or otherwise, is not purely quantitative or ordinal, or nominal. For instance, horizontal position seems like it inherently involves a quantity, but this is actually false. Quantities are only associated with seen positions in space because we frame them within an imposed interpretation of a coordinate system in which we define the origin and orientations of axes, effectively constraining interpretation of positions

as measures of quantity. We are habituated in constraining information interpretations as well as following shared common interpretations. Even ordinality has an ambiguity problem in need of interpretation when it uses a position channel, since we have to specify which direction the ordering is in. Not only can visual channels be utilized to show different types of information, but they also can be interpreted in diverse ways.

This work challenges the notion that there is just one way to interpret a representation of information, whether it is a data representation of bits or a graphical representation of visual channels. Bridging data to visual channels involves a great deal of careful consideration of the aspects of information carried by data and visual channels.

We take all visual channels from Mackinlay's ranking of visual channels [94], and exclude four visual channels: connection, containment, density, and slope. We do not consider connection because it is an explicit visual binary relation between two things. We exclude containment and density since there are relatively few examples to draw from to inform the model. We do not use slope because it is similar to angle and can be subsumed in it. Conversely, we split Mackinlay's position channel to distinguish 1D and 2D positions as distinct channels including horizontal position, vertical position, and 2D position. In Figure 7.2, we use a heat map as in Mackinlay's rankings but recast in terms of degree of likeness. The horizontal represents visual channels, and the vertical indicates the three different aspects. We observe that horizontal and vertical positions work well for all three aspects of an attribute, but 2D positions aren't perceptually ordered or metric unless one imposes a 1D ordering or metric on them. Furthermore, color hue is strong for nominal-like data but weak for ordinal-like and quantitative-like aspects because perceptual non-linearity and distinguishability weaken ordinal and quantitative interpretations. In addition, shape itself is not relevant to quantitative and ordinal information, but we mark it as weak for ordinal-like and quantitative-like data since the interpretation of a shape could convey an ordinal or quantitative meaning through its properties, for example, the number of its sides or corners.

Our theory challenges the accepted "atomicity" of visual channels and data types. The heat map might be extended by elaborating visual channels or expressing degree of likeness in finer or alternative ways. Although it is impractical to derive a comprehensive and accurate ranking of visual channels with respect to information aspects, our conceptual model for visual correspondence still can be built upon the heuristics from existing visual encoding theories that are still the common foundation for visualization research and practice.

7.2.3 Visual Cardinality

Data cardinality is the number of unique values of a data attribute. We define *visual cardinality* as the number of unique values of a data attribute that can be mapped into a visual channel with perceptual differentiability. As data cardinality increases, identifying corresponding visual objects becomes harder. (We refer to *visual cardinality* simply as *cardinality* from now on.) The visual identification of data values through a visual channel can be influenced by cardinality. For instance, color hue encodes at most 7-11 distinct "identities". Color hue can perform well for identifying a value from an attribute with data cardinality 5, but cardinality 10 starts to impede performance. Distinguishing color hues for an attribute with cardinality 20 or higher can be unfeasible. For visual correspondence, tasks involve identification and comparison. For instance, one sees an item and identifies it over here, then sees there is another item somewhere else that needs to be assessed by comparing the two to determine whether they are the same or not. Since visual channels are limited in terms of visually differentiable items, the interaction between visual channels and cardinality has an effect on visual effectiveness across different tasks. If the cardinality of a data attribute is too high, it affects the ability to identify a given value represented by a given channel. As a result, it also affects the ability to determine matches between visual encoded values in two places. Therefore, our model includes cardinality for visual correspondence analysis.

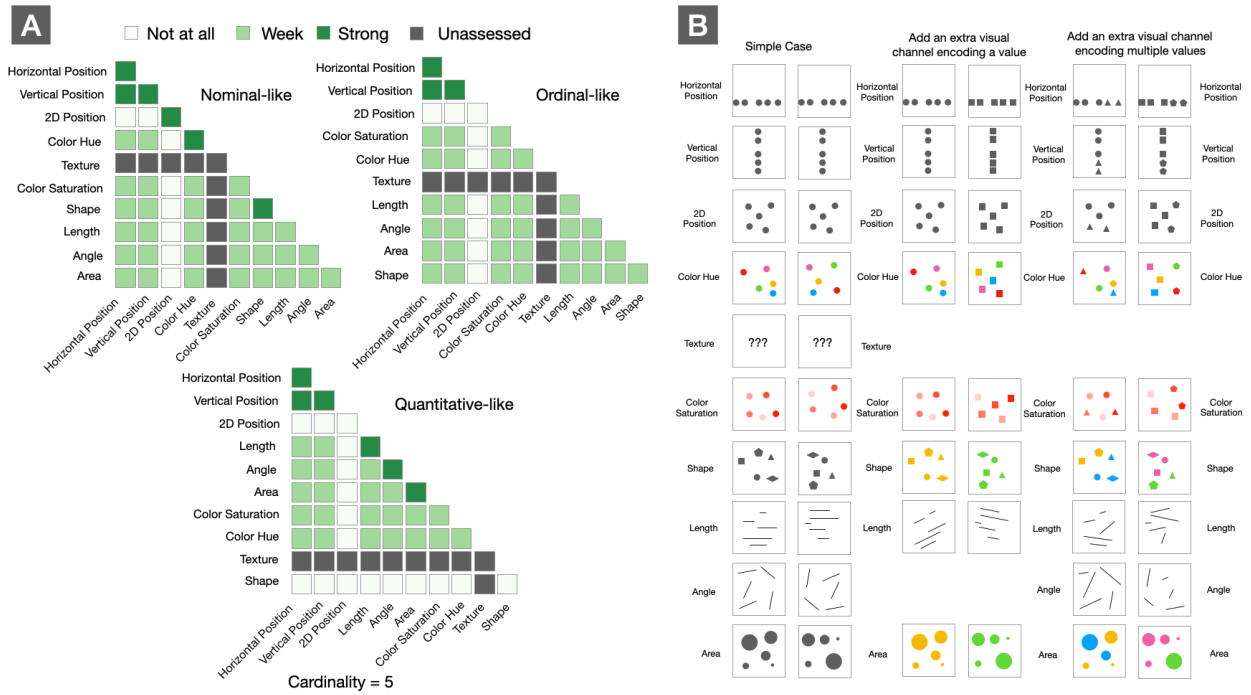


Figure 7.3: (A) Populated matrices for three information aspects with cardinality five. (B) Three cases of populating the model by assessing visual correspondence through visual encoding pairs in the diagonal cells of the heat map: a simple case, a case with an extra visual channel encoding a single value, and a case with an extra visual channel encoding multiple values.

7.2.4 Degree of Correspondence

Visual correspondence is the extent to which we can identify identity or equivalence between items shown in two different places through their visual encodings. The degree of correspondence is determined by four factors: the aspect of information, **a pair** of visual channels, and cardinality. The measure of the degree of correspondence includes steps about assessing against the theoretical criteria, for example, whether a visual channel can differentiate items at a certain cardinality. This is the same as how the visual encodings in Mackinlay's ranking were judged originally and as still judged in practice. The degree of correspondence is a relatively ordered scale from best-case to worst-case to not applying at all. The model can be thought of as being a collection of combination matrices for each information aspect and the level of cardinality, as shown in Figure 7.3.

Determining the degree of correspondence includes consideration of three criteria: *iden-*

tifiability, *consistency*, and *indeterminacy*. ***Identifiability*** is the ability to identify an item visually. Visual correspondence builds on identification tasks. If visual identifiability is weak, we also consider that visual correspondence is probably weak. ***Consistency*** is the quality of producing the same results after repeated testing by adding some distractors. A graphical item could have multiple visual encodings, and visual correspondence may be established by only one of the visual encodings. Varying conditions based on additional visual encodings and their related cardinality could help to find cases producing different results for degree of correspondence. ***Indeterminacy*** is the quality of being uncertain. The more possible things that need to match across the two visually encoded sets of possible values, the more difficult correspondence.

7.3 Applying the Model

In this section, we describe a general method for applying the model and a specific method to populate the model for common channels and cardinalities. Next, we discuss the results and apply them to practical examples.

7.3.1 General Method

We apply the model of visual correspondence by covering all three information aspects, most of the visual channels, and a representative subset of cardinality values.

In Figure 7.3A, each lower triangle heat map shows the degree of correspondence for each pair of visual channels under a specific information aspect and a specific cardinality. The diagonal cells are pairings of the same visual channel, while off-diagonal cells represent pairings of two different visual channels. We assess most of the visual channels from our model discussed in Section 3.2, except for texture. We initially included texture in our visual channel list. As we applied the method to populate the model, we found it too complicated to make recommendations for texture especially because it can combine the

characteristics from other visual channels (e.g., color, angle, shape, etc.) in complex ways. We leave treatment of texture to future work.

The general approach to varying cardinality is based on the typical regimes of the number of different values that visual channels can differentiate. As such vary the cardinality focusing specifically on cardinalities 2, 5, 10, and 20. Cardinality 2 is the base case. Cardinality 5 is a typical case for which most visual channels should work. At cardinality 10, some visual channels drop off but some still work well for encoding. At cardinality 20, only position typically works well for encoding. When position stops working for encoding, visual correspondence stops working too. In general, visual correspondence is bounded by the visual identifiability of the two visual encodings.

Measuring visual correspondence is complicated, especially for cases with two different visual channels. Even though we consider the degree of correspondence as a relatively ordered scale from best to worst, we simplify our method to have three levels for the degree of the correspondence (i.e., Not at all, Weak, Strong). Three-level ranking makes visual correspondence relatively easy to assess, thus facilitating the process of populating the model and helping to verify our methods, at least in the rough strokes.

7.3.2 Populating the Model

We create two views to encode the same data for each included visual channel. In Figure 7.3B, the simple case shows the views in side-by-side pairs that correspond to the diagonal cells in the matrices. This results in a representative set of prototype views with a few items each. To assess the degree of correspondence, we add some distractors by increasing complexity in terms of additional visual encodings and their related cardinality. We add an extra visual channel that encodes either a single value or multiple values in the other two cases. We omit the second case for the angle channel, because the first and second cases are the same if we consider length as the additional channel.

The cells for the degree of visual correspondence are filled based on the overall assess-

ment against the criteria for the three cases. The assessment considers whether the visually encoded items are identifiable, whether the results from the three cases are consistent, and the degree of uncertainty in matching items. We start by filling in the diagonal cells representing the same visual channel by looking at the side-by-side pair of views in each column of Figure 7.3B. After filling the diagonal cells, off-diagonal cells representing pairings of two different visual channels are filled by looking at the corresponding views in different row positions. We fill the lower triangle matrices for cardinalities 2, 5, and 10 based on the steps described above. For cardinality 20, our judgment of visual correspondence is based on the results from cardinality 10 and our practical experience with each visual channel's encoding ability. For example, color hues are harder to tell apart when the cardinality is 20 than when it is 10, making for a correspondingly low prediction of visual correspondence.

7.3.3 Observations and Patterns

Figure 7.4 shows an overview of the results after populating the model. In general, the diagonal cells representing same visual channel pairings work better than the off-diagonal cells representing different visual channel pairings. The top three cells indicate that pairings between horizontal and vertical positions can handle all three information aspects under all four cardinality conditions. In contrast, 2D position only corresponds well to another 2D position encoding, and only for nominal information. The pairing of area and length is strong for visual correspondence when the cardinality is 2 for a nominal-like attribute; for circles, this may be due to the functional relationship between area and a radius length. This suggests that functional relationships between channels tend to promote higher correspondence.

Cardinality is an important factor that affects the degree of correspondence. Most visual channel pairings have at least some visual correspondence when the cardinality is 2 or 5. As cardinality increases, the degree of correspondence drops. Some visual channels start to be challenging to use when the cardinality is 10. For instance, the pairing of color saturation to color saturation is not fit for perceiving corresponding items since visual identifiability is

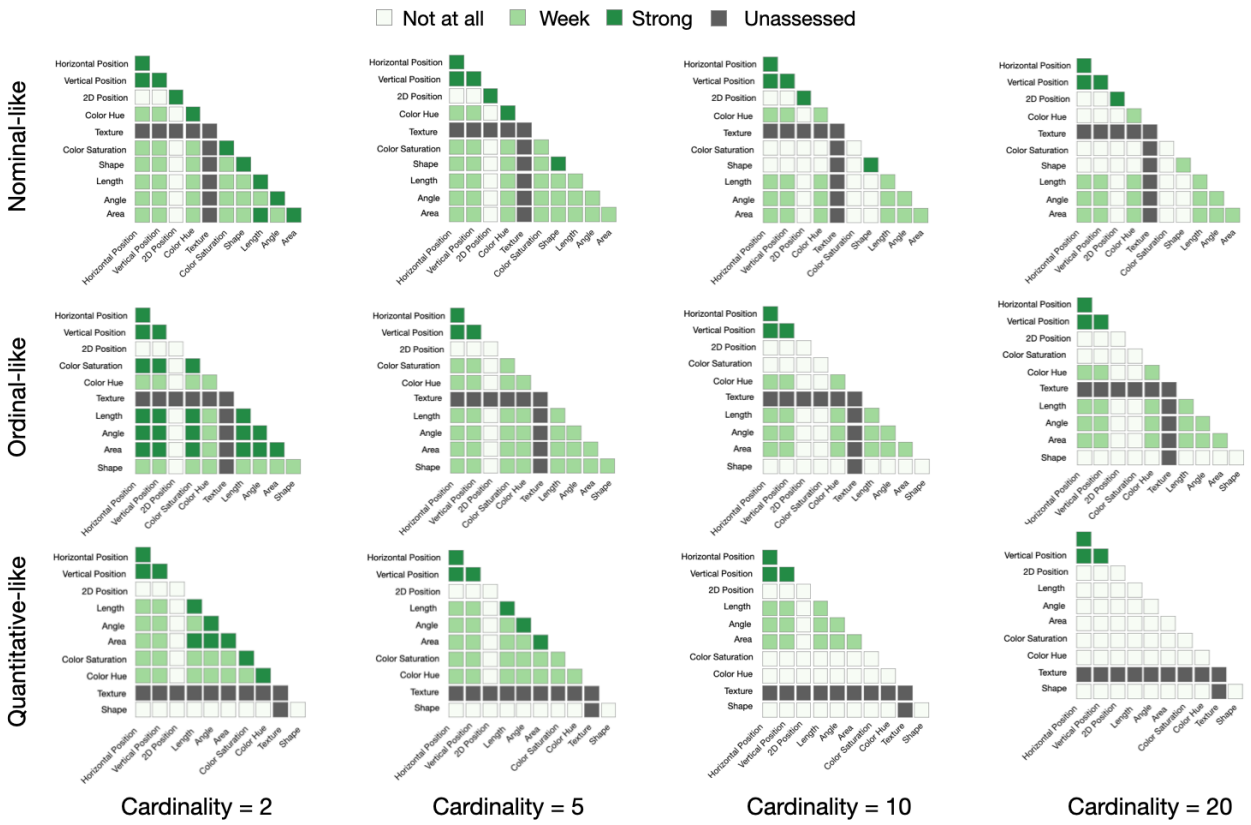


Figure 7.4: The combination matrices populated by our method. Each cell in the matrix shows the degree of correspondence for a pair of visual channels by an information aspect and a level of cardinality.

too poor if not absent. When the cardinality reaches 20, only position pairings are left. We assume that visual correspondence breaks down badly once an attribute is at a high enough cardinality, with 20 a sufficient maximum to include.

According to visual encoding theory, position ranks highly for nominal information, and hue is nearly as strong as position. When comparing position and hue, we observe that hue is slightly worse than position at low cardinalities, but a lot worse at high cardinalities. We should thus keep cardinality in mind, and switch to position from hue above a certain cardinality. By comparing the two pairings above, we gain insight into not only the effects of cardinality on perceiving identity through a visual encoding, but also its relative effect (with potentially high impact) on perceiving correspondence.

7.3.4 Applying the Results

We apply the results of the populated matrices to 12 example visualizations to assess the results, the general method, and our model. The 12 example visualizations shown in Figure 7.5 include: (a) Minard's map [144], (b) Literature fingerprinting [79], (c) XGobi [25], (d) timeline visualization [165], (e) MapTrix [167], (f) MotionExplorer [16], (g) Cluster viewer [147], (h) filled line chart and horizon graphs [69], (i) visualization course figures [99], (j) network visualizations [59], (k) network visualization for cliques [100], and (l) GrouseFlocks [12]. For each example visualization, there is a set of visual object pairings denoted as a letter and a set of key visual correspondences, including visual linking (if present), denoted as a number. For example, in the MapTrix visualization, there are three views, hence three pairings of views. The bar chart and a matrix view (Figure 7.5e-B) have a correspondence between vertical position and vertical position (Figure 7.5e-B1) and also use visual linking (Figure 7.5e-B4). We use our populated matrices as predictions, and compare the predicted correspondence for the general case to the expected correspondence for the particular case that we discern (from practical experience) in the example visualizations. There are two versions for the degree of correspondence. The predicted degree of correspondence from our populated matrices is shown in the upper-left triangle. The expected correspondence is depicted in the lower-right corner of the degree box next to the channel pairing views. The degree boxes next to a case of visual linking (e.g., Figure 7.5a-2) do not have a lower triangle, since our model does not offer predictions about visual linking. We use the information aspect, visual channels, and cardinality of the visualized data to look up the predicted degree of correspondence in the populated matrices in Figure 7.4. We use the closest cardinality in the populated matrices. (It is usually easy to round one way or the other without changing the prediction.) Next, we present examples of analysis of correspondence in the 12 example visualizations.

A hue to hue pairing can generate strong correspondence at low cardinality. In Figure 7.5g, the Cluster Viewer has a correspondence case (A1) between a calendar view and a line chart view. The strong green in the upper-left triangle indicates a strong correspon-

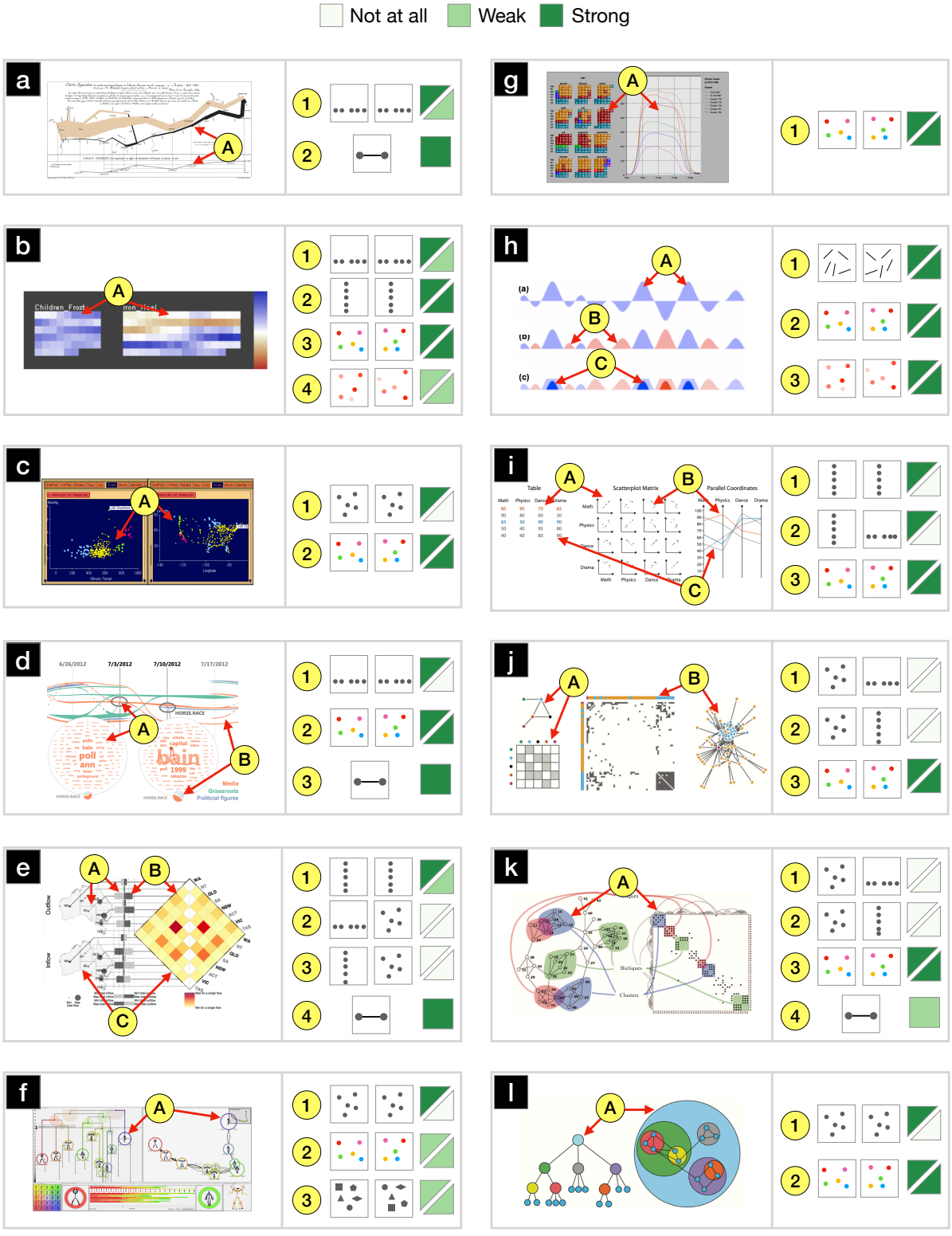


Figure 7.5: The 12 example visualizations used to assess natural instances of correspondence.

dence from our model prediction. The strong green in the lower-right that indicates strong correspondence is expected, too. Other cases involving a hue to hue correspondence (e.g., Figure 7.5d-B2, Figure 7.5j-A3, Figure 7.5l-A2) similarly show a match between predicted and expected correspondence. This case verifies the correct predictive ability of our model.

Figure 7.5k uses a node-link view and a matrix view to show the different types of cliques in a network. There are four correspondence cases. The first two pairings, between a 2D position and horizontal and vertical positions, are not corresponding at all in both prediction and expectation. The third correspondence case is a hue to hue pair, with strong correspondence both predicted and expected. We observe that the visual linking is actually weak for perceiving correspondence since the lines in the node-link view and the matrix view could interfere with the perception of the visual linking, and the visual link lines overlap each other, too. The visual linking contribution is minor at best to the group level correspondence, since the hue to hue correspondence already generates strong correspondence between the same cliques. Moreover, there is a visual correspondence between the color of the link and the color of the linked items. This provides an additional level of reinforcement of equivalence transitively through the link. Overall, the crossing lines make it hard to follow the path of visual linking, consequently; the correspondence between individual cliques via the links is not strong. Introducing another visual correspondence case into the visualization design could alleviate the issue of crossing lines. For example, by adding color saturation to the cliques in both views, the correspondence for individual cliques might be stronger, allowing the visual linking lines to be removed. This is an example of a potential design improvement resulting from model prediction.

In Figure 7.5a, one case of correspondence (A1) is horizontal position to horizontal position between the map view and the line chart. We see a strong green in the upper-left triangle from the prediction of our populated matrices and a weak green in the lower-right for the expected correspondence. Our model predicts a strong correspondence for this case, but the expected correspondence is moderately weak. The nuance with respect to Minard's

map is that the map weakly shows discrete positions as line intersections because the paths of the armies and the temperature plots are in fact line plots. However, the position that we use to fill out the model is based on discrete positions as opposed to bend points in the line chart. Consequently, being able to see the correspondence from position to position is weaker than predicted. Visual linking helps to enhance the corresponding positions explicitly connecting corresponding bend points.

Similarly, the correspondence case in Figure 7.5e-B1 is vertical position to vertical position between the bar chart and the matrix. Our model predicts strong correspondence, but the expected correspondence is moderately weak. The alignment between the bars and the rows or columns of the matrix is weakened due to the rotation of the matrix. This is a case of visual linking (B4) strongly enhancing an otherwise weak correspondence. The perception of correspondence for a position to position channel is strongly dependent on how clear the alignment is. Similar cases include Figure 7.5d-A1 and Figure 7.5i-A1, B1, C1. The alignment in those cases is very bad; consequently, the expected correspondence is none. This suggests possibilities to extend the visual channels applied in the model to a more nuanced organization.

The example visualizations (Figure 7.5c, f, l) have correspondence cases with 2D position to 2D position (A1). Our model predicts strong correspondence, but the expected correspondence is none. In example visualization c, the two views have different coordinate systems. The left view uses a normal 1D$\times$1D coordinate system of a scatterplot, but the right view uses a 2D geographic coordinate system. In example visualizations f and j, the organization principles are different. Motionexplorer (f) uses a dendrogram view and a node-link view. For the illustration in GrouseFlocks (l), the left view is a top-down node-link tree view and the right view is a circular treemap. Since the two views in each visualization have different spatial organization principles, the 2D position to 2D position mappings are ambiguous if not determinate, offering little if any correspondence. This indicates how the model implicitly assumes 2D positions are in the same, equivalent, or at least readily translatable coordinate

systems. This limitation of the model may carry over to other geometric channels as well. All three example visualizations use a hue to hue pairing to establish strong correspondence to compensate.

7.4 Discussion

In this section, we discuss additional limitations of and opportunities for developing and applying the model.

Non-same Channel Pairings. Based on the 12 example visualizations, we observe that all of the visual correspondence cases use pairings with the same visual channels to generate correspondence. This observation suggests that exploiting visual correspondence between two different visual channels is rare. It illustrates the need for more examples of visualization designs to be created that show off different channel pairings for visual correspondence. The pattern of populated matrices shown in Figure 7.4, which suggests that the same visual channel pairings work better than different visual channel pairings, can help explain this finding. Similarly, the 2D position to horizontal/vertical position cases in the example visualizations (Figure 7.5e, j, k) match the pattern in the populated matrices in which 2D position only corresponds strongly to another 2D position. These observations and the patterns in the populated matrices indicate a predictive power that can benefit visualization design by supporting consideration of effective visual correspondence.

Means to Correspond. Visual linking uses explicit graphical connections to represent correspondence. Brushing establishes or reinforces correspondence by temporarily turning on additional visual encoding features such that corresponding items more clearly match relative to other items, i.e., they are highlighted in their respective views. Visual correspondence, visual linking, and brushing thus all serve the same purpose to support seeing equivalent items in multiple places. Strategic combination of correspondence methods including visual linking, brushing, and "passive" visual correspondence could lead to interesting and powerful

new techniques for visualization designs in a variety of static and interactive circumstances. The trade-offs among correspondence methods could be considered for integrating diverse approaches across multiple views. Moreover, the potential for exploiting weaker correspondence in pairings of different visual channels remains almost entirely unexplored, and might lead to deeper investigation of how people perceive information differently across visual channels and view contexts.

Cardinality Regimes. The 12 examples involve a wide range of visual cardinality. The strategy of picking a close cardinality from the populated matrices could be problematic due to the nuances of visual identifiability among different visual channels. Sampling more levels of cardinalities could alleviate the problem and help with more accurate predictions. In addition, most correspondence cases in the example visualizations are low cardinality cases. Evidence at high cardinality is sparse. Visualizations to study correspondence at higher cardinalities are needed. The model may itself encourage the development of such visualizations.

Same Versus Equivalent. Based on the 12 example visualizations, we observe that the relationship between corresponding visual objects can be one-to-one (e.g., Figure 7.5a), one-to-many (e.g., Figure 7.5g), or many-to-many (e.g., Figure 7.5c). The arity relationship of visual objects could be an important additional dimension to extend the model. The current model presented here measures the effectiveness of individual item correspondence cases between pairings of visual channels. One could also assess overall collective correspondence from multiple pairwise correspondence cases among a set of related visual objects. The individual correspondence cases might interact with each other, for example, two weak correspondence individual cases contributing to a strong overall correspondence.

Correspondence Strength Levels. While perceptual rankings [94, 101] generally treat visual channels as separate from each other, interactions between different visual channels can play an important role in perceiving correspondence visually. For example, a small size of a shape could reduce or eliminate the visual correspondence established by a pairing of

shape channels. Furthermore, the relationship between the encoding capability of visual channels and the degree of visual correspondence needs development. Evaluation of this information could help to identify new and subtle factors for assessing the effectiveness of visual encoding choices within and between views. The degree of likeness and the degree of correspondence is estimated in a coarse manner, calling for systematic dissection of both through empirical studies.

Ratio of Unique to All Data Values. In our assessment of the visual encoding pairs, the number of data values is equal to the cardinality of the data values, however, it is more often that the number of data values is greater than the cardinality. The assessment could consider different cases in terms of cardinality relative to the number of data values. Specifically, the distribution of data values for a quantitative attribute could be included in assessment of visual correspondence. Including data from a variety of domains could also help to gain in-depth insight into how people perceive visual correspondences. Mechanical Turk and similarly crowdsourced perception experiments are viable and seem likely to contribute to understanding the nuances of different visual channel pairings for visual correspondence.

Other Factors to Consider. There are additional factors that seem likely to contribute to correspondence, such as alignment between two 1D position channels and the layouts of items in views with 2D channels. Ignoring those factors could result in wrong predictions. Moreover, the specific design of a visual object can have an impact on perception of visual correspondence; for example, 1D position as discrete points versus as line bend points. The designs of visual objects can also influence visual perception of items collectively, such as the ability to perceive alignment of a set of items, calling for extensive consideration of a wide variety of item encodings ranging from simple marks to complex glyphs.

7.5 Guidelines

Based on the results of applying and analyzing the model, we propose a set of guidelines for designing and evaluating visualizations that account for visual correspondence. Certainly, approaches exist to establish visual correspondence overtly, such as visual linking. These guidelines aim to also encourage less overt establishment of visual correspondence through the pairing of channels. Creating a visual correspondence between a pair of visual objects could entail multiple steps. In general, the first step is to see to what extent the base visual channels that the visual representations already use can create visual correspondence. Next, if we cannot use visual channels due to externally imposed or prior constraints on how the visual channels are used, can we mix in an additional visual channel? The detailed guidelines follow. An overview is shown in Figure 7.6.

G1: Consider more effective pairwise channels to visually correspond objects.

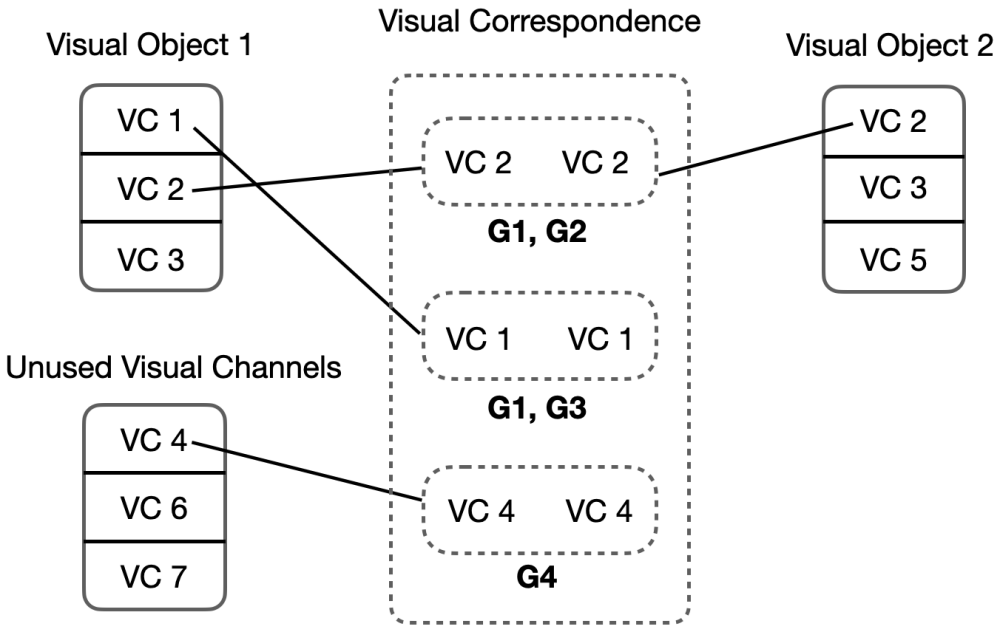


Figure 7.6: Guidelines for considering visual correspondence. G1: Consider more effective pairwise visual channels (VC) to visually correspond objects. G2: If possible, choose the same or highly related visual channel for corresponding visual objects. G3: If G2 fails, reuse an already used visual channel of one visual object. G4: If G2 and G3 fail, pick the most highly visually correspondent visual channel from unused visual channels for any aspect of the information.

The appropriateness of using a visual channel to encode a data item depends on several factors, and those factors can interact with each other, inadvertently or not. It is not clear whether there exist any situations in which a strong visual correspondence can coincide with very poor visual encoding of information. A methodical approach to visual correspondence will list candidate visual channels, and sort them by the degree of likeness which indicates their individual visual effectiveness for the information to be encoded. If there are multiple viable pairings of visual channels for that information, the option with the highest degree of likeness for each visual channel is suggested.

G2: If possible, choose the same or highly related visual channel for corresponding visual objects. Unsurprisingly, we found that using the same visual channel between two sets of visual objects almost always affords very good visual perception of correspondence. This is essentially what common forms of highlighting do in brushing, after all. In Figure 7.4, the dark green diagonal cells indicate the same or better correspondence between the same visual channel than between different ones. Highly related can be nearly as strong, such as vertical versus horizontal position (that are basically subcategories of the position channel) or in the case of color, color hue versus color saturation. However, it may not always be possible to establish visual correspondence using seemingly related visual channels. For example, the U.S. states might be encoded as rows (1D position) in a list and as regions (2D position) in a map. Vertical position and 2D position are predicted to be strong encoding channels for nominal-like data. Items are clearly distinguishable as rows and regions in their respective views. Nevertheless, the visual correspondence between a vertical position in the list and a 2D position in the map is poor. It is hard to tell which vertical item corresponds to which region of the map, even though they represent the same thing. We would have to use some other visual channel to actually create the visual correspondence, and may even have to resort to symbols (state names or abbreviations as labels) to establish the correspondence explicitly.

G3: If G2 fails, reuse an already used visual channel of one visual object. If

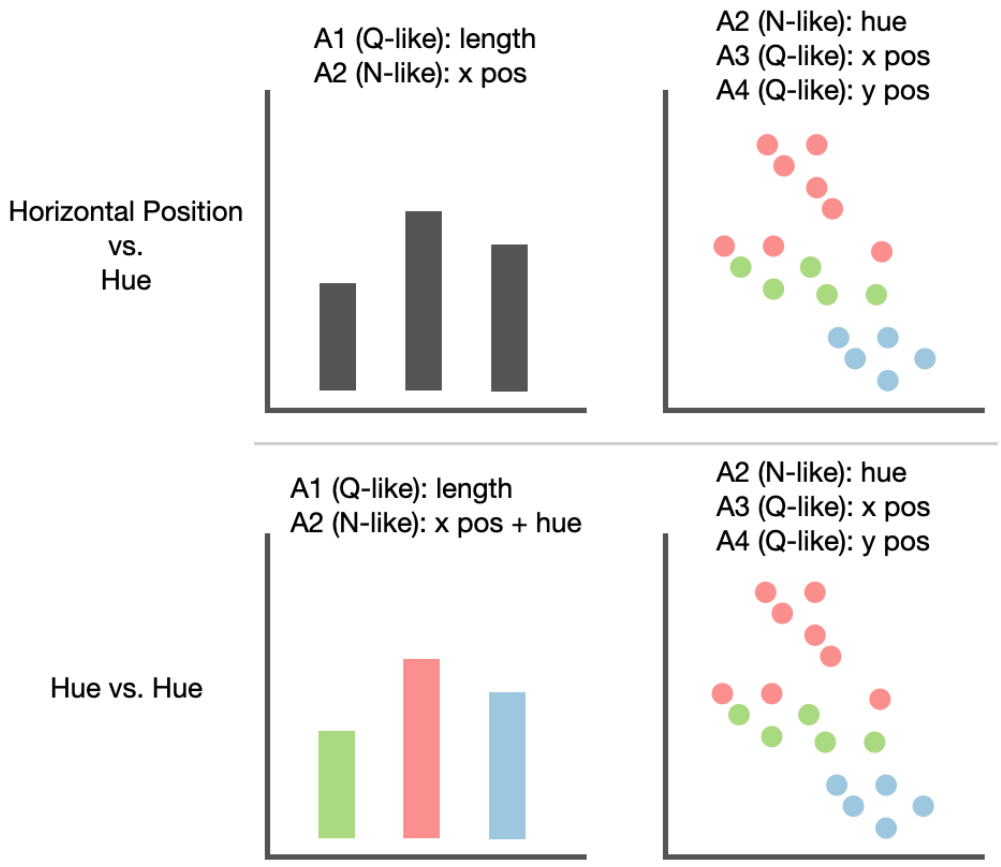


Figure 7.7: An example of Guideline 3. At top, the two views have weak visual correspondence for attribute A2. At bottom, the two views show a strong visual correspondence through addition of color hue for A2 to the bar chart.

the base visual channels in the visual representations are not sufficient for perceiving correspondence, one possible way forward is to consider visual channels already used for visual encoding. As shown in Figure 7.7, the top bar chart encodes a quantitative-like attribute (A1) using length and encodes a nominal-like attribute (A2) by horizontal position. For the scatterplot, two different quantitative-like attributes (A3, A4) are encoded in the horizontal and vertical position respectively, and the same nominal-like attribute (A2) is encoded by color hue. Although horizontal position and color hue are strong encoding channels for the nominal-like data, the visual correspondence between these two visual channels is weak. It is hard to tell which bar in the bar chart is related to which colored point in the scatterplot, even though the underlying data is exactly the same in both. However, we can reuse the color hue visual channel from the scatterplot view to encode the nominal-like attribute (A2)

in the bar chart. The hue to hue channel pairing establishes a strong visual correspondence between the two views.

G4: If G2 and G3 fail, pick the most highly visually correspondent visual channel from unused visual channels for any aspect of the information. Visual channels are not always available for visual correspondence due to existing use. They may be reserved for layout or for encoding of particular data attributes. For example, vertical position is

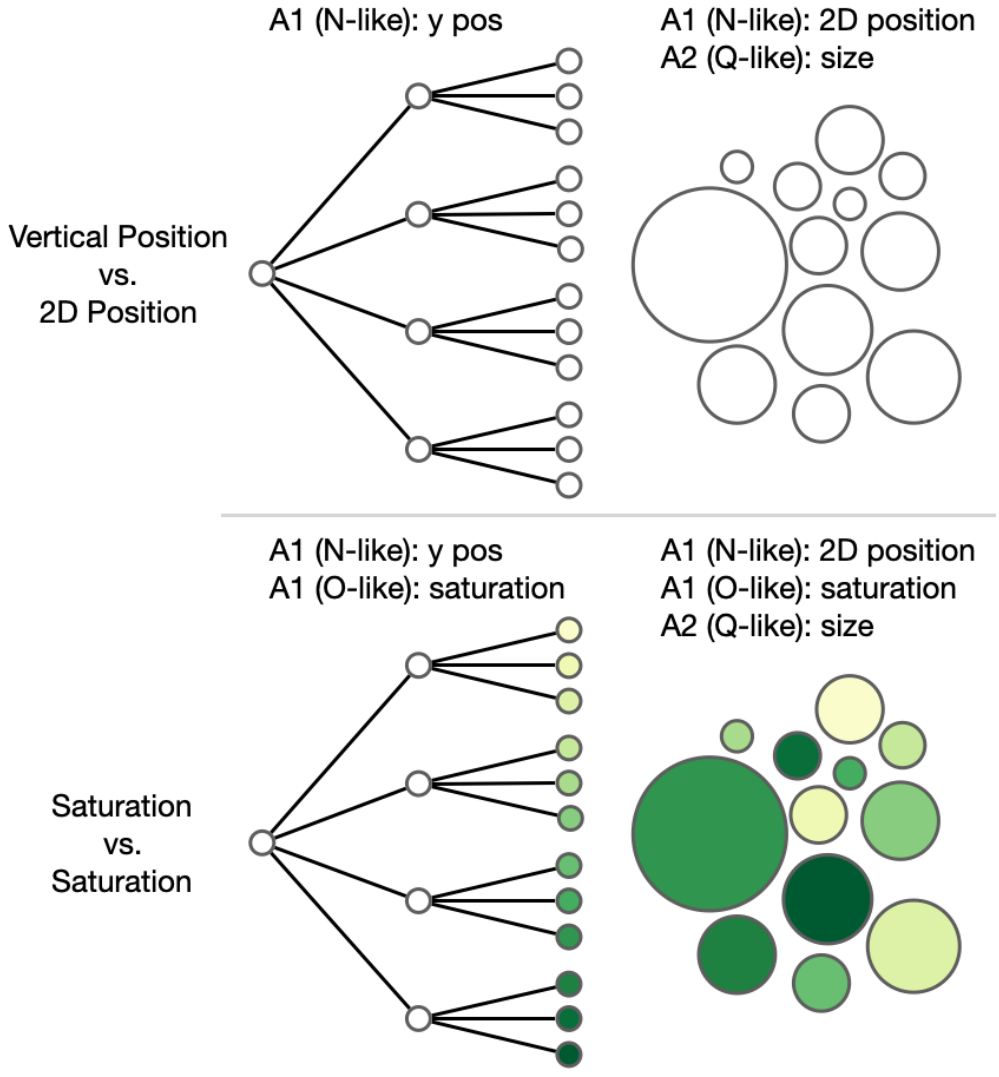


Figure 7.8: An example of Guideline 4. The top pair of views do not show correspondence at all, despite the attribute A1 being encoded in both views. The bottom panel depicts a relatively stronger visual correspondence after adding a saturation channel encoding for the ordinal-like aspect of A1 to both views.

used to layout each level of nodes in a left-right node-link tree. In such cases, visualization designers should consider other any remaining available visual channels and how they might be utilized in contribution to establish correspondence for the visual objects. As shown at the top of Figure 7.8, the same data attribute (A1) can be encoded differently by position. It is encoded as vertical position of leaf nodes in the tree visualization at left, while the bubble chart at right encodes it as 2D position. Both position channels encode the nominal-like aspect of A1. While either vertical position or 2D position is generally one of the best choices for a nominal-like attribute, the visual objects representing the same attribute values in the two views do not correspond at all. As shown at the bottom of Figure 7.8, A1 also has an ordinal-like aspect; for example, a month can be either nominal-like or ordinal-like data. We check the unused visual channels and find that color saturation is one of the best choices for an ordinal-like aspect (up to a certain cardinality). After applying saturation to A1 for its ordinal-like aspect in both views, the visual correspondence is enhanced. The pairing of saturation channels represents the ordinal-like aspect of A1 (e.g., month) in the fill color of leaf nodes in the tree visualization and bubbles in the bubble chart.

7.6 Toward Systematic Evaluation of Correspondence

Despite the substantial theoretical foundation of visual encoding and evolving understanding of its application, **in practice selecting visual encodings remains highly contextual and experience-driven. Applying the model of visual correspondence is similar in these regards.** For both visual encoding and visual correspondence, continuing to populate a space of diverse examples is essential for the ongoing development of theory. In particular, exploring the relationships between the two is necessary to better understand their roles in identification, comparison, and other perceptual steps in visualization tasks. In this section, we sketch a future program of theoretical and empirical work to determine functional relationships between the two in terms of their combined effectiveness for key

perceptual steps in visualization tasks.

Much like visual encodings provide a means to perceptually identify data objects and attributes within a visual representation, visual correspondences provide a means to perceptually compare data objects and attributes between visual representations, and thus serve as a critical bridge to identify the same (or equivalent) data objects and attributes as seen from different perspectives. We envision breaking down visualization tasks into non-perceptual and perceptual sub-tasks like these to identify components of a general perceptually-aware mechanics underlying those tasks, akin to how Fitt's law and Hick's law predict functional relationships to optimize efficiency in selecting and laying out user interface components. In our sketch, we speculate under a working assumption that the individual and combined influences of visual encodings and correspondences on the effectiveness of perceptual steps are functionally linear; the equations are merely suggestive placeholders for functional relationships to be determined. We consider first visual encodings on their own, then visual correspondences between pairs of visual encodings.

7.6.1 Visual Presence within a Visual Encoding

Visual encoding is the correspondence between a data item and how it is visually encoded. Visual encoding helps in understanding how well the visual representation of a data item captures not only the data value but also the desired aspect of information in the data value. We treat the correspondence between a data item and its visual object as a *visual presence* of the item within a visual encoding. The effectiveness of visual encoding involves an identification task to perceive the data item through the visual object. The *visual effectiveness for identification* ($E(\cdot)$) is defined as follows, and is determined by the degree of likeness of the visual channel A (L_a) to the desired apsect:

$$E(a) = k_a L_a + k \tag{7.1}$$

in which k_a is the constant for visual channel A and k is a baseline constant. Since the degree of likeness is an ungrounded value, we expect it to be derived from some function of the quality of the visual encoding. The constants may be something like baseline visual effectiveness, and we can not tell them apart for different visual channels. Although we expect the general constant to be zero, we could get a non-zero value when we actually measure under the assumption of a linear model. Developing empirical methods to determine functional forms and coefficients is a matter for future work.

7.6.2 Visual Correspondence between Visual Encodings

For visual correspondence, we have two visual objects for a given data item. In this case, the effectiveness is related to the comparison task. The comparison depends not only on whether or not the two visual encodings are effective or not, but also on whether the two visual channels are visually correspondent. This means that strong visual correspondence alone may not guarantee effectiveness of comparison; poorly chosen visual encoding channels may undermine it. Therefore, to assess effectiveness for comparison we factor in how well a visual encoding captures the aspects of the data value that we want to show, as well the visual correspondence between the two visual channels. We define the *visual effectiveness for comparison* ($E(a \rightarrow b)$) of a visual object encoded by visual channel A to a visual object encoded by visual channel B as:

$$E(a \rightarrow b) = k_a L_a + k_b L_b + k_{ab} C_{ab} + k \tag{7.2}$$

in which L_a and L_b are the degrees of likeness for visual channel A and visual channel B, respectively, C_{ab} is the degree of correspondence from visual channel A to visual channel B, k_{ab} is the constant for the visual correspondence between visual channel A and B, and k is again a baseline constant. The constant k_{ab} could be low if the overall effectiveness for comparison is primarily determined by the effectiveness of the individual visual encodings.

7.6.3 Discussion

While we do not want to suggest specifics of a formalism, we offer mathematical forms in a figurative way to describe how components might combine to make overall predictions of task effectiveness. In this manner, the formulas serve two purposes. First, we use them as a way to summarize how key factors interact in combination to influence task effectiveness. Second, the formulas can lead us to move forward and design experiments to understand how these factors combine with each other to determine the visual correspondence between items. Using the formulas one could model overall visual effectiveness in terms of degrees of likeness and visual correspondences. Systematic empirical study over a space of suitably varying visualization designs would ask users to perform identification and comparison tasks to determine the formulas and constants, and thereby assess the model. If the formulas and constants were found to be consistent over useful portions of the visualization design space, they could be integrated into an automated visualization design process. Effective visualization designs could be predicted from task descriptions.

7.7 Summary

In this chapter, We offer a new perspective to think about visual channels in terms of how they relate to each other, and suggest considering the effect of pairwise visual channels in the design of visualizations. We present *visual correspondence*, a conceptual model of how well associated visual objects are correctly seen as encoding the same data. The effect of pairwise visual channels is assessed for basic views and information aspects in order to gain initial insights into the utility and predictive capability of the model. Guidelines offer strategies to visualization designers to incorporate stronger visual correspondence in their designs. We lay out a program of future research to systematically develop measures of visual correspondence and their potential contribution to a mechanics of visualization tasks.

The motivation to develop the model of visual correspondence arose during development

of the initial version of the *PatternTree* visualization technique described in next chapter. The hybrid tree visualization design revealed issues of being able to establish correspondence while browsing. We developed the general theory of *Visual Correspondence* to help associate visual objects which encode the same data through pairwise visual channels for connecting different views to each other. We then applied the theory specifically to refine the PatternTree technique, and in turn assessed the theory through design cases of PatternTree applications. These efforts are described in next chapter.

Chapter 8

PatternTree: A Hybrid Tree Visualization for Hierarchical Patterns

8.1 Overview

Visualization helps users to explore phenomena, gather insights, communicate ideas, and memorize information from data. Exploratory data analysis endeavours to understand data from different perspectives through manipulation, such as filtering, grouping, or aggregating. Visualization of hierarchical data is important in a wide range of domains, including genome research [106], political redistricting studies [66], event detection [152], and sports analytics [27]. One study considered the role that visualization design plays in human cognition by measuring the memorability of visualizations [21].

We are interested in studying user comprehension of visualizations that convey hierarchical information. Trees are important data structures for representing hierarchical information. There exist numerous ways to visually represent tree structures [125]. When data is multifaceted [78], such as when it consists of different attributes or data computed from dif-

ferent approaches, a visual representation must convey the relationships between attributes as well as in the data structure overall.

In this chapter, we formalize a structure of hierarchical information for designing and implementing virtual layering in multifaceted visualizations. Virtual layering is a form of composite visualization [75] that embeds multiple views [153, 155, 145] as nodes inside a host node-link diagram. Virtual layering provides smooth visual transitions between the data structures in the embedded visualizations and those in the overall host visualization. These transitions can enhance the comprehension of both the composite visualization and the underlying data. Although we concentrate here only on virtual layering in node-link tree visualizations, we predict that virtual layering can be applied to graphs and many other kinds of multifaceted data structures.

Specifically, we propose a *PatternTree* technique that uses virtual layering to support smooth navigation and drill-down into data structures embedded inside hierarchical information. We apply *PatternTree* to study *gerrymandering*, the political manipulation of boundaries of electoral districts with the intent of creating partisan advantage [52]. We demonstrate the utility of *PatternTree* and its virtual layering for visual analysis of electoral data sets.

The main contributions of this chapter are:

- *virtual layering*, a new visualization design pattern that facilitates smooth visual transitions in composite visualizations (Section 8.2);
- a *design space* for integrating *virtual layering* into the nodes in a node-link tree visualization (Section 8.2);
- a *hierarchical pattern structure* for statistical analysis of multidimensional data and its visualization using the *PatternTree* technique for exploratory data analysis (Section 8.3–Section 8.5);
- two use cases involving a presidential election data set to demonstrate the utility of *PatternTree* (Section 8.8), and

- an assessment of visual correspondence (Chapter 7) in the *PatternTree* use cases (Section 8.9) and the applicability of the *PatternTree* technique in general (Section 8.10).

8.2 Virtual Layering

We can create a composite visualization that represents data with nested relational semantics using virtual layering as a means to transition between containing and contained visualizations. The virtual layering aims to smooth out the disconnect between the two-dimensional visual space of a containing visualization and the two-dimensional visual spaces of the visualizations contained in it. We refer to these two "layers" as the identity portion and the view portion, respectively. The identity portion visually represents the relationships between nodes in the tree (Section 8.2.1). The view portion visually represents the relationship within nodes in the tree in the form of embedded visualizations that show different facets of data for their respective purposes (Section 8.2.2), which may be the same or different from node to node.

8.2.1 Identity Portion

The identity portion conveys information about a node in its role as part of a hierarchy. As shown in Figure 8.1, there can be two identity portions in the virtual layering for a node located in a middle layer of a tree visualization. Since in this case we embedded the virtual layer into a left-right oriented node-linked diagram, the left identity portion indicates the node's relationship to its parent node or ancestry, and the right identity portion indicates the node's relationship to its children nodes or descendants. Since we embed a visualization into a tree at a specific layer, the embedded visualization will break the connectivity of the original tree structure. Consequently, extra mental effort is required to identify the association between the tree node and the embedded visualization. The identity portion allows users to transition from the tree visualization to the embedded visualization smoothly and efficiently.

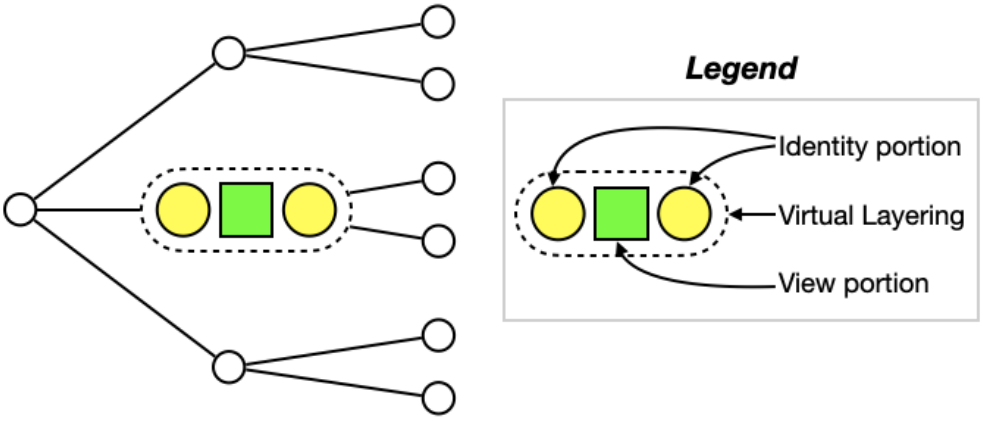


Figure 8.1: Basic virtual layering concept illustrated in a node-link diagram. The virtual layering consists of the identity portion and the view portion.

A well-designed identity portion can reduce the competition for space between tree links and the embedded view; as a result, it mitigates visual clutter for the visualizations.

8.2.2 View Portion

The view portion conveys information about a node in its role as a container of embedded data. It serves as a display space of an embedded visualization in which multiple aspects of embedded data may be rendered, such as multivariate attributes, statistical analysis, or geospatial context. A view portion can be a single visualization or a composite visualization such as a juxtaposition of multiple visualizations. A view portion can support visual alternatives which are activated by user interaction and vary in the approach to displaying information. The integration of view portion and identity portion strives to reduce the mental effort of transitioning from the tree visualization into the embedded visualization and back out again. The view portion is limited by the space constraints of the tree visualization layout.

8.2.3 Visual Compositions in the Virtual Layer

Design patterns for visual composition in the virtual layer can apply the existing approaches presented in the literature [75, 67]. Putting off exhaustive investigation of such patterns as

future work, here we illustrate the virtual layering technique through a few prominent, practical design patterns. Two common visual composition patterns for virtual layering follow the *juxtaposition* and *superimposition* patterns common to multiple view compositions. *Juxtaposition* places the visual representations of the identity and visual portions side-by-side in the virtual layer. As shown in Figure 8.2a, the identity portion, consisting of nodes in the tree, is placed next to the view portion represented by a dot strip plot. *Superimposition* overlays the visual representation of the identity portion to the side of the view portion. Figure 8.2b shows virtual layering in the middle level of a node-lined diagram, with both parent and child identity portions superimposed on left and right vertical axes of a scatterplot, respectively.

A third pattern unifies the identity portion with the view portion. The idea of *unification* is not to show the visual representation of the identity portion explicitly, but rather to utilize a visual association approach, such as association by visual ordering, to indicate the association between a parent node and its child nodes. Figure 8.2c shows a hybrid that combines two different methods. The parent identity portion uses *unification* to abstract

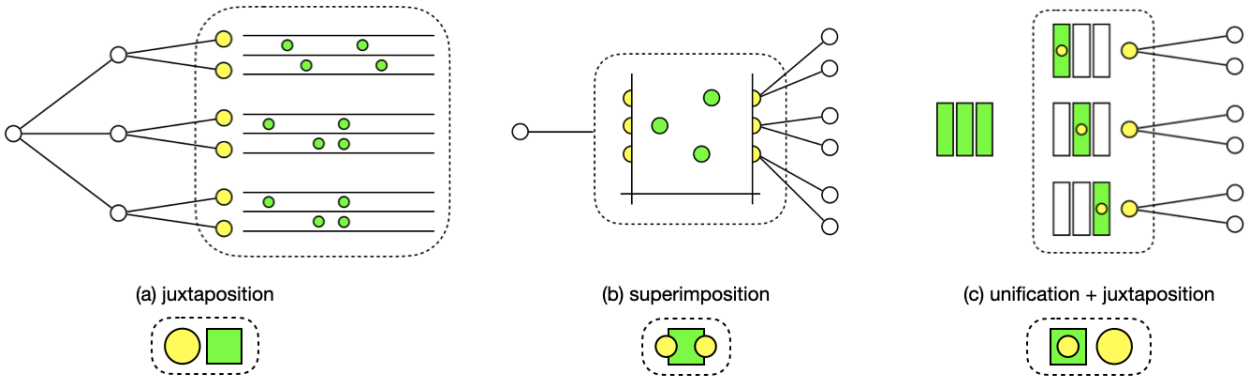


Figure 8.2: Composite visual designs for the *identity* and *view portion* of the virtual layer. The dashed rectangles indicate the current virtual layer. Yellow nodes represent the visual items for the identity portion while green items represent the visual items in the view portion. (a) Identity nodes juxtaposed with dot strip plots. (b) By superimposition, identity nodes are overlaid on the axes of a scatterplot. (c) A hybrid approach uses unification for the parent identity and juxtaposition for the child identity. The parent identity portions are implicitly unified within the replicated heatmap views from its parent, whereas the child identity nodes are juxtaposed to the right of the heatmap view.

the relation between two virtual layers by associating visual ordering with highlighting of visual items in views replicated from the parent. The child identity portion is *juxtaposed* to the right of the replicated views. The nodes directly drawn inside the replicated views represent the parent identity portion. In practice, those nodes can be omitted thanks to the visual correspondence between horizontal position (in each node's view portion) and vertical position (between each child node's view portion).

8.2.4 Positioning View Portion

In a left-right oriented node-linked diagram, the location and size of view portions are most constrained by vertical space. Horizontal space is easier to adjust as needed. Strategies to distribute embedded views over a vertical space vary. We investigated strategies for three situations, as illustrated in Figure 8.3.

Figure 8.3a shows a situation in which the view's height is the same as the interval between the first and last sibling nodes of a parent. Because the view aligns well with the vertical space taken by the nodes, we can use the first node's y position for the view's top y position, and the last node's y position as the view's bottom y position.

Figure 8.3b shows how a view with a shorter height can be centered in the middle vertically. (We could still draw the view at the first node's y position.) To center the view, the y position for the top of the view is simply computed as:

$$y = y' + (\frac{1}{2}interval - \frac{1}{2}height), \tag{8.1}$$

in which y' is the y position of the first node in the branch, $interval$ is the vertical distance between the first node and the last node in the branch, and $height$ is the height of the view.

In Figure 8.3c, the height of each view exceeds the vertical interval of its branch in the tree. Since consistency of view size may make comparisons easier, in this scenario we assume views have the same height. To place multiple views, we can calculate the y position for the

first view as follows:

$$y_1 = y' - \frac{1}{2}(n * height - total_interval), \tag{8.2}$$

where y' is the y position of the first node in an entire level, n is the number of nodes/views in the level, *height* is the height of each view (plus any desired padding), and *total_interval* is the distance between the first node and the last node in the entire level. After we get the position for the first view, we can compute the y position for the rest of views by adding the view height to the y position of each previous view in turn, as follows:

$$y_i = y_{i-1} + height. \tag{8.3}$$

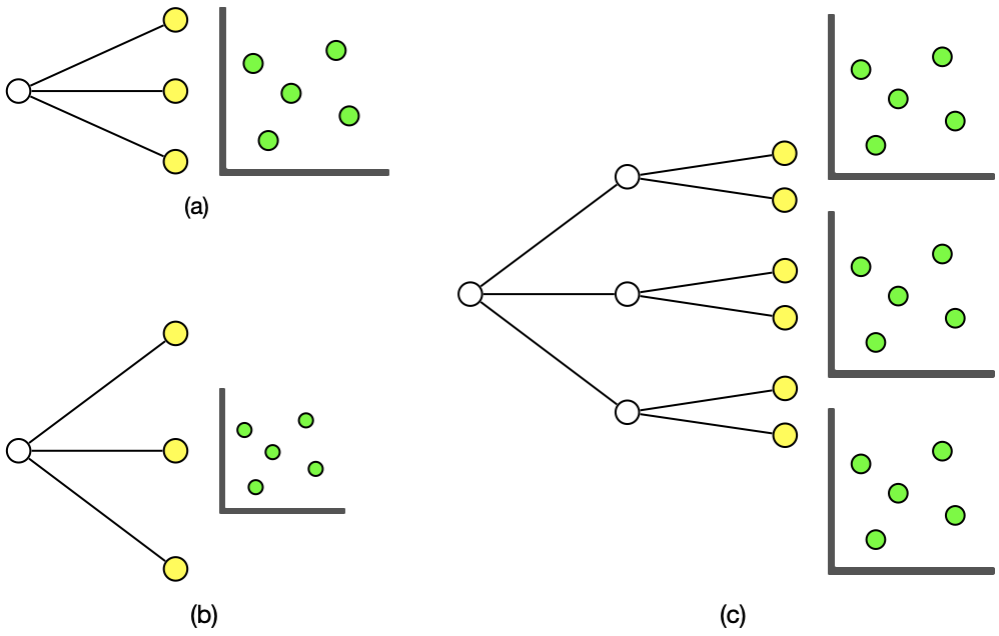


Figure 8.3: Positioning the view portion for three different situations involving different view heights. (a) Positioning a view when its height is the same as the branch height. (b) Centering a view that is shorter. (c) Placing multiple views with equal heights each larger than its branch height.

8.2.5 Association Between Identity and View Portions

An identity portion may couple with a single element in a view portion (1:1), such as a bar in a bar chart or a dot in a scatterplot. Multiform visualization [116] presents the same data in alternative ways by applying different types of visualization or different visual encodings. When the identity portion visually encodes some data, multiform visualization of the same data can be applied to the view portion to create a one-to-one visual relationship between the components. Users can choose the multiform visualization to suit the data, domain, and analysis needs.

An identity portion may also be coupled with multiple items in a view portion (1:M). The view portion presents additional dimensions of the data that need to couple with the identity portion. For example, a set of bars in a grouped bar chart or a group of dots in a scatterplot may associate with an identity portion. For example, in the Gerrymandering case, the leaf nodes represent the districts under a districting plan. If we draw a grouped bar chart to show the voting share percentages of the two parties for each district, the identity portion (which represents the district) will associate with two grouped bars that each represents a party's voting share percentage in the view portion. The view portion aims to provide a visualization of complementary information from different aspects of the data.

Many types of perceptual cues can be applied to make the association between identity and view portions stronger and more salient to users. The goal is to reduce the cognitive and perceptual load of view transitions by taking advantage of the Gestalt principles [83], particularly *proximity*, *similarity*, and *continuity*. As shown in Figure 8.4, we assessed 14 visualization types for potential use as view portions in virtual layering. We categorize the visualization types as either *overlapping* or *space-filling* [50], or in an *other* category, which includes additional kinds of visualizations, such as bar charts and tables. Below, we discuss in detail ways to strengthen the perceptual mapping between an identity portion and a view portion.

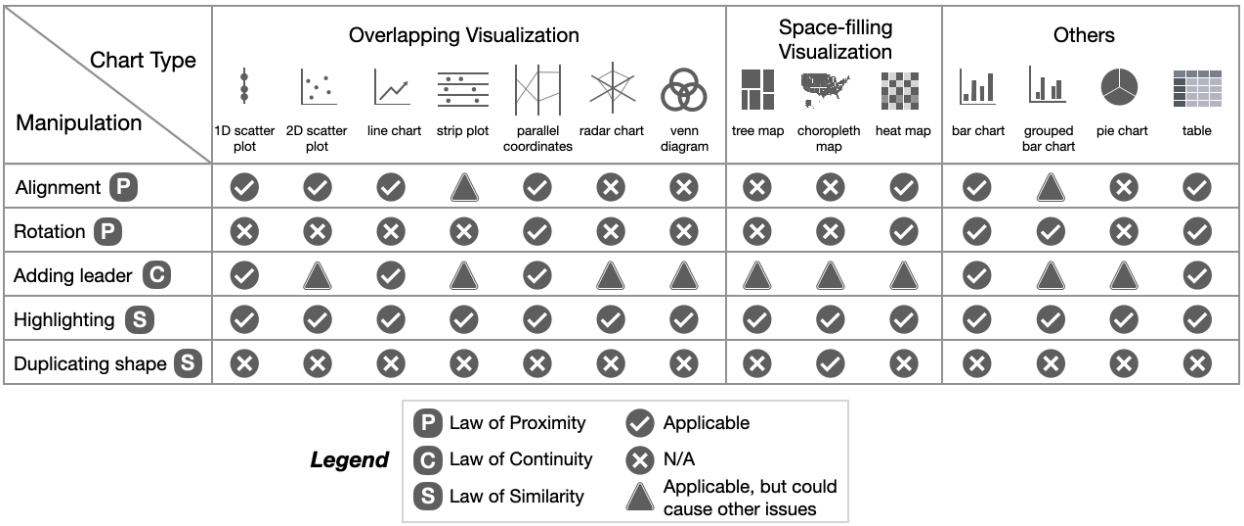


Figure 8.4: Visual manipulations and their applicability to common visualization types.

Law of Proximity. Objects that are close to one another are perceived as belonging to the same group. An association can be created by spatial proximity via positional alignment or view rotation.

Alignment can be characterized in terms of the data relation between the identity portion and the view portion. Aligning items in the identity portion to items in the view portion (item-item alignment) can be used to create a 1:1 relation, whereas aligning items in the identity portion to the view of the view portion (item-view alignment) can be used to create a 1:M relation.

An item-view alignment can vertically move the identity items so that they are close to the view. For example, in Figure 8.3c, we can move the top two leaf nodes close to the middle of the y-axis of the uppermost scatterplot. Some visualization types can be aligned without moving the identity items, such as the strip plot, as shown in Figure 8.2a.

An item-item alignment is more complex than an item-view alignment. Figure 8.4 summarizes how most visualizations that contain two-dimensional geometric shapes, such as radar charts, venn diagrams, tree maps, choropleth maps, and pie charts, are not amenable to item-item alignment. Items in those visualizations are positioned in a two-dimensional space, but identity items are positioned vertically. The mismatch between 1D space for

identity items and 2D space for view items makes item-item alignment hard in most cases. Heatmap and table visualization are special cases in that the identity items can be aligned with rows that represent view items. Similarly, we can place identity items on the y-axis of a 2D scatter plot to align them horizontally with plot items, as shown in Figure 8.2b. This alignment approach for a 2D scatter plot allows users to perceive item associations unless there are too many overlapping items on the y-axis. The same approach can be applied to line charts and parallel coordinates. Strip plots and grouped bar charts may be amenable to alignment, but issues can arise from grouping. Each identity item can correspond to an entire group of items rather than the individual items in these groups. One or more levels of such 1:M correspondence might require more cognitive effort to associate each identity item with particular view items through their groupings.

An embedded view can also be rotated or transposed to make the relationships between identity items and view items more apparent. Rotation exploits the internal spatial arrangement of the view to reposition relevant items in the view portion close to the corresponding identity items. Visualizations like scatter plots, heatmaps, and tables have both horizontal/row and vertical/column dimensions that one can rotate by 90° or transpose. When identity items are associated with columns in a view, rotating the view to turn columns into rows can help the user perceive item associations based on the law of proximity.

Law of continuity. The law of continuity is also known as the law of good continuation. It states that visual objects are more likely to be followed by the human eye and perceived as more related if they are in a continuous flow. Using a leader line to connect an identity item and its associated item in the view portion offers a strong way to perceive a 1:1 relation between items. Although adding leader lines is a powerful approach to explicitly bridge visualizations and is easy to implement, they have shortcomings that designers should keep in mind. Leader lines may penetrate a view and potentially occlude items. When leader line crossings occur, it can generate visual clutter and diminish the perception of continuity,

making it harder to follow to see group elements along the path. Additionally, when a visualization can contain grouping or hierarchical structure, such as strip plots and grouped bar charts, adding leader lines may require extra cognitive effort to identify which items in the corresponding level of grouping or hierarchy that the lines mean to connect. This issue is similar to that of alignment for those visualization types.

Law of similarity. People tend to perceive as grouped objects that are visually similar, through features, such as shape, size, and color. We illustrate two manipulations of virtual layer components that exploit similarity to improve their correspondence.

First, highlighting can provide users with immediate visual feedback to perceive the association between identity items and view items. Although highlighting can be applied to all visualization types listed in Figure 8.4, they often entail different design choices. The most common highlighting design changes the color of selected items or their edges. When a highlighted item is small, increasing the item's size can improve visual salience to enhance the perception of its association.

Second, items can have the same or similar shapes to associate them. Specifically, identity items can duplicate the shape of their corresponding view items. For example, when identity items represent districts in a state and the view is a districting map of the state, we can use the district's shape in the map to draw the identity items. The shapes need to be distinctive for recognition. In their generic forms, of the visualization types in Figure 8.4, only the choropleth map may be suitable using shape as a similarity cue.

In practice, multiple manipulations may strengthen item association, and one can weigh which to use under the circumstances. It is also possible to combine two manipulations, such as adding leader lines and highlighting with color. The designer should consider trade-offs among the various visualization types and which kinds of manipulation they support.

8.3 PatternTree Design

We develop an improved version of how users can see the collection of the dimensional combinations of heatmaps in Wiggum using a hierarchical approach. *PatternTree* is designed for visualizing patterns in hierarchical data structures that embed complex node information. We first introduce a hierarchical pattern structure. To explore patterns, a node-link diagram is used to represent the portions in the hierarchical pattern structure.

8.3.1 Hierarchical Pattern Structure

We structure an information hierarchy as a tree in which we explore patterns in a data set and its partitions. Given a data set, D, we define a *pattern* as the relationship between a pair of attributes $\{x_m, x_n\}$. For those attributes, we denote a *pattern* $P(\{x_m, x_n\}, D)$ for the overall data set, and call it the *overall pattern*. D can be omitted for simplicity. A hierarchy can be established by dividing the data set into multiple *subgroups* $D_1, D_2, ..., D_n$ which contains a set of records grouping by a third attribute x_s. We call the third attribute as *splitby* variable. A pattern P of $\{x_m, x_n\}$ created from a subgroup of the data D_i is denoted $P(\{x_m, x_n\}, D_i)$, and we call it a *subgroup pattern*.

8.3.2 Visual Design

We describe the hierarchical design of the *PatternTree*, which uses a node-link diagram as the base representation.

Hierarchical View

The hierarchical pattern structure can be organized into a three-level tree structure as shown in Figure 8.5. The overall pattern $P(\{x_m, x_n\})$ maps to the node in the first level of the tree structure, and the subgroup patterns $P(\{x_m, x_n\}, D_i)$ map to the nodes in the third level. The different *splitby* variables x_s are the nodes in the middle level of the tree, and they are

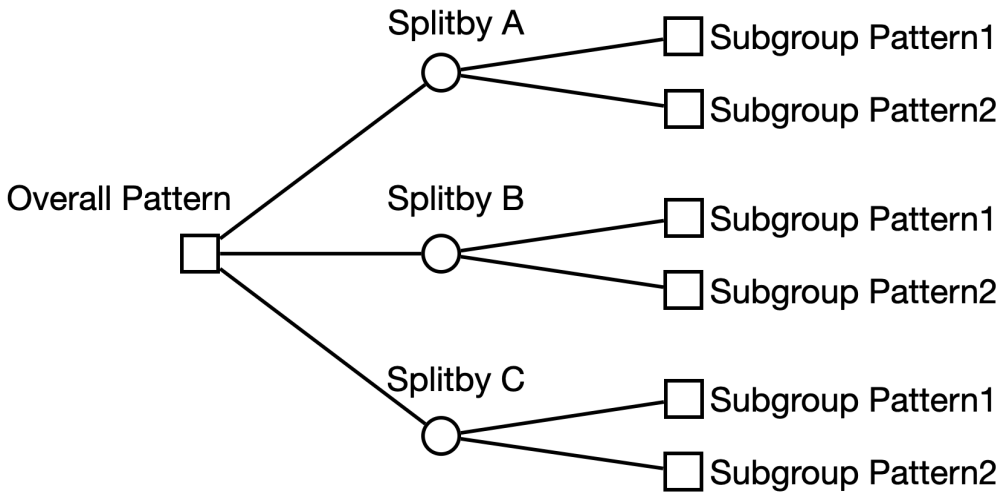


Figure 8.5: The hierarchical view design for the PatternTree.

connected to the overall pattern and their own subgroup patterns. The hierarchical structure can be extended by further dividing the data set, but here we focus on a one-level partition to illustrate the utilization of virtual layering in example domains. To differentiate the pattern levels from the splitby level in the node-link diagram, we can use shape to encode their nodes. For example, a circle node represents a splitby variable, and a square node represents patterns in Figure 8.5. An alternative method is to apply background color to alternate levels. There are two reasons why we choose a node-link diagram over other hierarchical visualization techniques [126] like Treemaps [129], Sunburst [133], or icicle plots [160] as the base representation, First, the non-leaf nodes for splitby components are more expressive in a node-link diagram than in a Treemap because Treemaps strongly emphasize leaf nodes over all other nodes. Second, node-link diagrams have more flexible layouts than space-filling methods [124]. For example, it is easy to adjust the widths of each level in a node-link diagram to make room for embedded views.

8.4 Use Case: Gerrymandering

Gerrymandering refers to the manipulation of voting district boundaries to advantage a political party or group. Negative effects of gerrymandering can include letting politicians choose

their voters, packing partisans, splitting communities, diluting minority votes, etc. [89]. When states redraw their districts, they may consider and balance diverse criteria for a wide variety of factors. The most common criteria and factors, such as equal population, minority representation, contiguity, compactness, communities of interest, etc., are often reviewed to ensure a fair redistricting plan.

There are a variety of ways to measure continuity and compactness when comparing the redistricting maps drawn by independent redistricting commissions to those drawn by state legislatures [49]. In addition to measures based on redistricting criteria, the seats-votes relationship can be useful for assessing the swing ratio and partisan bias of redistricting plans [143]. Measures of electoral competition can also contribute to comparison of the redistricting plans; a recent study suggests that, despite expectations, independent redistrictors may not increase electoral competition to achieve a more neutral political balance [70]. Most recently, DeFord, Eubank and Rodden [46] introduced a measure, *partisan dislocation*, to indicate cracking and packing by considering a voter's geographic nearest neighbors.

Rather than developing measures, researchers use automated redistricting algorithms to generate redistricting plans and perform comparative analysis to reveal gerrymandering. Two types of redistricting algorithms that are commonly used in the redistricting literature are partitioning algorithms and swapping algorithms, which can be integrated into a divide and conquer redistricting algorithm [88]. Automated districting simulations that are blind to partisanship and race are used to measure unintentional gerrymandering that can emerge from patterns of the geographic distribution of voters with respect to partisanship [32]. The difficulty of statistical analysis with quantitative metrics can discourage plaintiffs from raising legal challenges to gerrymandering [135]. Making statistical analysis more visually interpretable and less confusing is a primary motivation for this work.

Table 8.1: Example: Precinct-Level Data for the 2016 and 2020 Presidential Elections in Oklahoma with Congressional Districts for the 2010 and 2020 Cycles.

District 2010 Cycle	District 2020 Cycle	GEOID20	Party	Votes 2016	Votes 2020
2	2	40079000102	DEM	75	82
2	2	40079000102	REP	385	412
3	3	40113000113	DEM	17	14
3	3	40113000113	REP	37	40
3	3	40113000102	DEM	31	24
3	3	40113000102	REP	126	167

8.5 Patterns in Gerrymandering

To illustrate the features of our hierarchical pattern structure, we use precinct data as shown in Table 8.1. It shows the votes for each party per precinct in the state of Oklahoma in the 2016 and 2020 presidential elections. The data contain congressional districts for two redistricting cycles. For example, the 2010 cycle for the congressional districts in Oklahoma starts on May 10, 2011 and ends on Dec 31, 2021, whereas the 2020 cycle is from Nov 22, 2021 to Jun 30, 2031 [1]. We compare statistics between the two major parties. For example, the party having a higher voting share percentage (VSP) in a district, which is calculated by Equation (8.4) given $i \in precincts$ and $j \in parties$, wins the election for a House seat in the U.S. Congress.

$$VSP(votes|party = j) = \frac{\sum_i votes_i \wedge party = j}{\sum_i votes_i}. \tag{8.4}$$

Pattern $P(\{votes, party\})$ indicates the relationship between the votes variable (x_m) and

the party (x_n) variable. In our example, we use voting share percentage as the aggregate measure over precincts, and apply a binary ranking pattern on it, such as $VSP(votes|party = Dem) > VSP(votes|party = Rep)$. The pattern can be generalized to two variables with some statistical analysis on one of the variables, as shown in Equation (8.5).

$$P(\{x_m, x_n\}) : stat(x_m|x_n = a) > stat(x_m|x_n = b). \tag{8.5}$$

in which the ranking can be greater than ($>$) or smaller than ($<$); we exclude the possibility of a tie ($=$) in a two-party contest.

In the gerrymandering case, the *overall pattern* $P(\{votes, party\})$ indicates which party has a higher voting share percentage in the state-level data, while a *subgroup pattern* is the pattern in the district-level (i.e., $P(\{votes, party\}, D_i)$, in which D_i is a partition of the data for a specific congressional *District* value (i.e., $district = c$). Congressional districts are divisions of a state. Seeing the pattern in each district could provide an important perspective on disparate results from elections.

Given the patterns, a distance function can be applied to measure if two binary ranking patterns $P(\{x_m, x_n\})$ and $P(\{x_m, x_n\}, D_i)$ are the same or reversed:

$$d(P(\{x_m, x_n\}), P(\{x_m, x_n\}, D_i)) = \begin{cases} 0 & \text{if the same pattern} \\ 1 & \text{if the reverse pattern} \end{cases} \tag{8.6}$$

For example, if $P(\{votes, party\})$ is the *overall pattern* that Republicans have a higher voting share percentage at the state-level and $P(\{votes, party\}, D_i))$ is a *subgroup pattern* having lower voting share percentage for Republicans in a specific district, the distance between the two patterns is 1. Using a distance function alone can not reveal if a districting plan of a state is gerrymandered. One approach to assess gerrymandering is to examine all districts' distances. We propose a summary function S, defined in Equation 8.7, using pattern distance d and $|x_s|$ as the number of the unique distinct values in variable x_s

(i.e, district). The value of S indicates what proportion of the districts do not follow the state-wide pattern. If we compare the summary of district distances with the voting share percentage of the minority party at the state-level, the disparity suggests that the districting plan is gerrymandered. For example, the minority party has a 40% voting share percentage at the state-level, but the sum of the distances is 0.6 in a state with five districts. The sum indicates that the minority party should win the majority of the seats (3 out of 5). The different outcomes (i.e, 40% vs 60%) may be caused by gerrymandering.

$$S(x_m, x_n, x_s) = \frac{1}{|x_s|} \sum_{\forall D_i \in D} d(P(\{x_m, x_n\}), P(\{x_m, x_n\}, D_i)) \tag{8.7}$$

For the binary ranking pattern, we further provide a strength for exploring how well a pattern represents the data being examined. We apply the winning margin, as shown in Equation 8.8, to represent the pattern strength s in the study of gerrymandering. The winning margin helps to examine the competitiveness of congressional elections. By comparing the competitiveness of different redistricting plans, analysts can find additional support for the hypothesis that a redistricting plan is partisan gerrymandering.

$$\begin{aligned} s(P(\{x_m, x_n\})) &= |stat(x_m|x_n = a) - stat(x_m|x_n = b)| \\ &= |VSP(votes|party = a) - VSP(votes|party = b)| \end{aligned} \tag{8.8}$$

8.6 Color Selection

Color is a powerful visual channel to guide attention. We try to pick better color schemes for different components in our complex hybrid visualization system. We first consider the well-known colors in our application domain which is political science. The color blue is associated with the Democratic Party, and the red is for the Republican Party. The political colors are reserved for the embedded charts that convey political parties' information. Next, we select a yellow-green sequential color palette to encode the *distance* values in both tree

nodes and relevant elements in the virtual layering. The color green and yellow are visually distinguishable to the predefined political colors. The yellow-green sequential color palette is more easily distinguishable between low distance values and high distance values than the green only sequential color palette, especially when the distances in the leaf nodes are either 0 or 1. Last but not least, we exclude the predefined colors above and use other colors based on the different hue channel for the charts in the hybrid visualization.

8.7 Implementation

As shown in Figure 8.6, the *PatternTree* system architecture consists of three major parts: (1) a web browser, (2) a Flask Server, and (3) a Python computation library. The web browser allows users to interact with the functions of the *PatternTree* web application, which was developed using HTML, CSS, JavaScript, and D3 [22]. The D3.js library provides flexibility and interactability to create visualizations on the client side. Flask [3] is a web application microframework that allows connection between a Python computation library and JavaScript-powered visual analytics in the browser. The Flask server processing involves two main components: a controller and a model. The controller accepts user input and interacts with the Python computation library to execute tasks such as validating input, computing patterns, etc. The controller calls the model to handle the processed data in order to prepare it for visualization; for example, the controller sends the result table to the model and the model generates and returns distance matrices. An Ajax request allows a web application to send and retrieve JSON formatted data between the web browser and the Flask server. The Python computation library consists of all of the computational features, and is executed in the back-end server. *PatternTree* has been factored as a modular framework. Each individual component can be changed or augmented, e.g., to implement a different clustering method for data augmentation, add a new trend type, etc.

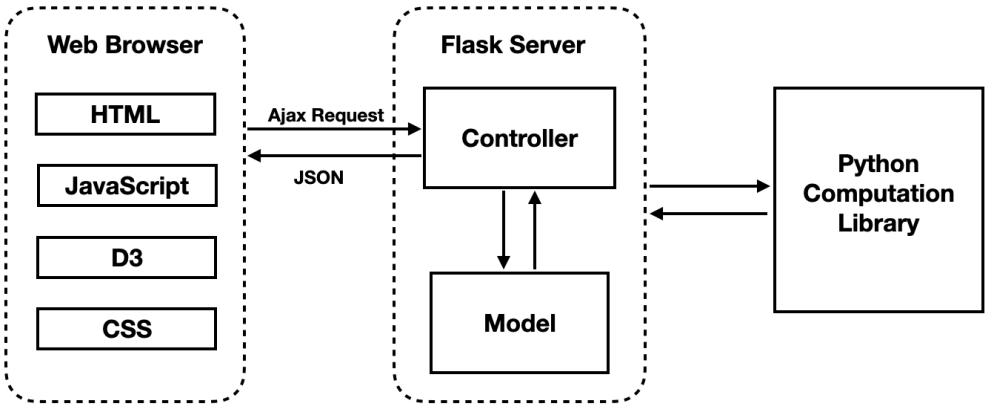


Figure 8.6: An overview of the PatternTree system architecture.

8.8 Analysis Examples

In this section, we describe how *PatternTree* can be applied to the electoral precinct data of the U.S. presidency from *Dave's Redistricting App* [2] and congressional redistricting data from *All About Redistricting* [1]. For our analysis, we applied the 2016 and 2020 presidential vote share data at the precinct level. The election data was disaggregated to 2020 census blocks following the method described by Amos, et al. [10].

8.8.1 Evaluating the Efficiency Gap

This use case was inspired by an evaluation of the efficiency gap [135]. The relationship between seats and votes in the two-party system [143] could indicate the efficiency gap. We inspect changes of the efficiency gap before and after the redistricting plan.

At the beginning of the analysis, we start by looking at the structure of the hierarchical data. The *PatternTree* has four levels, as shown in Figure 8.7(a). Looking at the heatmap view of the root node, we observe that there are two patterns in the electoral data set. The first pattern is *P({Votes2016, Party})*, representing the binary ranking pattern extracted from the precinct-level votes data of the 2016 Presidential election and each party's information. The second pattern *P({Votes2020, Party})* applies to the 2020 Presidential election. In the first level of the tree, the two patterns are split into corresponding branches. The

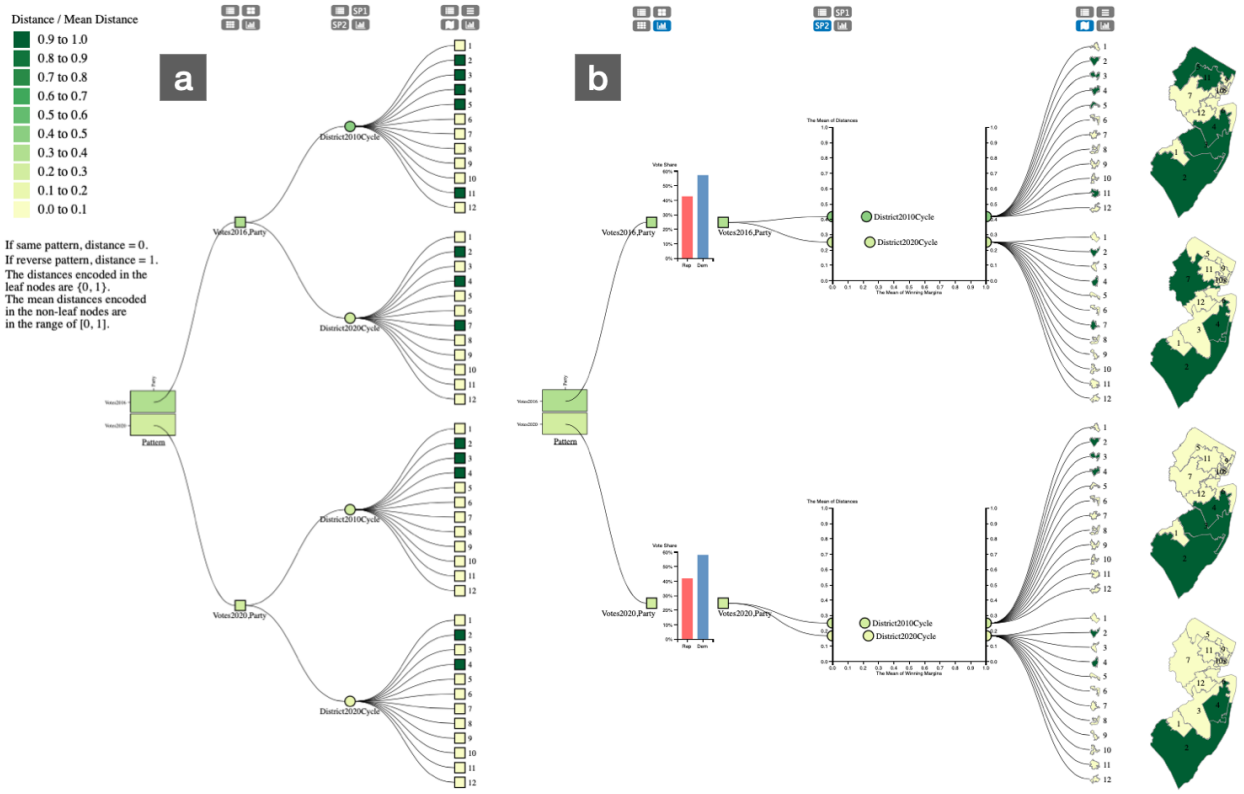


Figure 8.7: An example demonstrating the basic design of the PatternTree technique for evaluating the efficiency gap. The user first visualizes an initial PatternTree (a). Later, the user selects three views to embed into the PatternTree (b): a bar chart comparing vote shares between two parties, a scatterplot showing summary statistics, and a map view providing more context for districts in New Jersey.

next level shows the *splitby* variables, in which we see that the congressional districts for each election year are split into 2010 and 2020 cycles. By comparing these two cycles, we can gain insight into how the new redistricting plan has an impact on the election. The last level of the *PatternTree* shows the leaf nodes that present the congressional districts for each combination of cycle and election year.

We further investigate the details in each *PatternTree* level, as shown in Figure 8.7(b). In the first level, the mean distance aggregated for the second pattern *P({Votes2020, Party})* is 0.208. By clicking the show view button for views in the first level, we see that the pattern *P({Votes2020, Party})* at the state level is that Democrats receive a higher vote share than Republicans in the presidential election (58% vs 42%). In the *splitby* level, we

show a scatterplot to view the statistics calculated for the two districting cycles. We find that the mean distance in the 2020 cycle is 0.167. Comparing this number to the vote share for Republicans, it is obvious that the gap between overall statewide support (42%) for Republicans and the proportion of their winning districts (16.7%) indicates partisan bias and hence supports a contention of gerrymandering. In addition, we see that the 2020 cycle has a lower mean distance than the 2010 cycle. This shows that partisan bias is worse under the new districting plan.

To understand the disparity and examine the congressional district lines, we inspect individual districts in the leaf nodes. We observe that the distances for District 2, 3, and 4 in the 2010 cycle are equal to 1, which represents a reverse pattern compared to the pattern at the state level. In contrast, the districts whose distance is 1 in the 2020 cycle are District 2 and District 4. Since District 3 is no longer showing the reverse pattern, we investigate several aspects of gerrymandering. One crucial aspect relating to gerrymandering is whether the congressional district lines of the flipping district are drawn differently in the new districting plan. The embedded map view clearly shows that the congressional district lines for District 3 are quite different between the two cycles. Another aspect is to check the degree of partisanship. Switching to a strip plot at the leaf level, we find that the vote shares for the two parties in District 3 are extremely close in the 2010 cycle (50.1% Republican versus 49.9% for Democrat), while the vote shares in the 2020 cycle are much further apart (58.1% Democrat versus 41.9% Republican). Meanwhile, the vote shares for the Democratic party in the adjoining districts including Districts 2 and 4, decreased in the 2020 cycle, whereas the vote shares for the Republican party increased. This indicates possible gerrymandering techniques such as cracking and packing. In this case, the Republican voters may have been cracked from District 3 into District 2 and 4; consequently, those votes are diluted. Concomitantly, the Democratic voters from District 2 and 4 might have been packed into District 3.

Through this exploration, we gained understanding of how the district lines are drawn

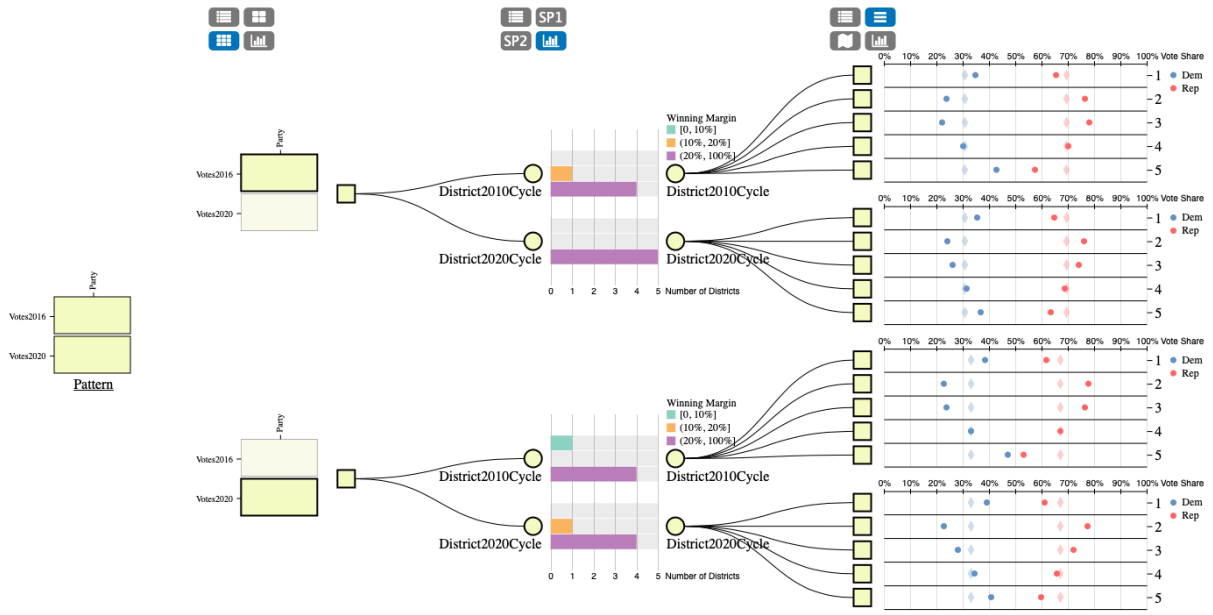


Figure 8.8: Example of a PatternTree to assess electoral competition. The user selects a heatmap view in the first level, a grouped bar chart in the splitby level to explore changes in competitiveness between the two districting cycles, and a strip plot in the leaf level to show vote details for districts in Oklahoma.

and what the possible consequences might be in future elections. This use case validates the effectiveness of the *PatternTree* technique to identify potential gerrymandering, and suggests further exploration.

8.8.2 Assessing Electoral Competition

In a second use case, we demonstrate how we utilize *PatternTree* to investigate the redistricting plan that appears to impact electoral competition.

Figure 8.8 shows the *PatternTree* for the Presidential election in Oklahoma. In this case, the root level and the first level are not connected by lines; instead, the two duplicated and highlighted view portions in the first level implicitly represent the parent-child relationships with the root node. The mean distances of the internal nodes and the distances of the leaf nodes are all 0, as indicated by their respective fill colors. This clearly indicates that the district-level patterns are compatible with the statewide pattern in which the Republican

party always wins every district in Oklahoma.

Since we are particularly interested in the recent election, we track the lower branch which presents the pattern *P({Votes2020, Party})*. When we look at the bar chart in the splitby level, we find that one highly competitive district (i.e., a winning margin of 0% to 10%) in the 2010 cycle becomes less competitive in the 2020 cycle; the number of moderately competitive districts (i.e., a winning margin of 10% to 20%) changes from 0 to 1 after redistricting.

To delve more deeply into these changes, we more closely examine individual districts' voting shares. We embed a strip plot in the leaf level, and examine the changes of the voting shares for the two parties. We find that the competition is close in District 5 in the 2010 cycle (i.e., Republican 52.9% vs. Democrat 47.1%), and that the difference between the shares of the votes for the two parties becomes larger in the 2020 cycle (i.e., 59.5% Republican vs. 40.5% Democrat). This indicates that District 5 is probably safer and more secure for Republicans after redistricting. In addition, we find that District 3, which is a neighbor of District 5, shrinks its winning margin in the 2020 cycle. We assume that the district lines for both districts were redrawn in the new redistricting plan. To confirm this, we switch to map views in the leaf nodes and observe a notable difference in the border between the two districts.

Our findings in Oklahoma are a strong sign that gerrymandering can occur even without overturning elections. Based on our exploration, we conclude that Districts 3 and 5 were likely gerrymandered to protect Republican candidates especially in District 5. Overall, the *PatternTree* technique aids in exploring hierarchical patterns and gaining a deeper understanding of underlying information.

8.9 Visual Correspondence Assessment

In order to build a hybrid tree visualization, the theory of visual correspondence can be applied to connect the host view (node-link tree) with various types of embedded views. In

this section, we describe how we refine the design of PatternTree. We also provide a basic assessment of the theory of visual correspondence through application to the PatternTree.

In the PatternTree, correspondence between the node-link tree view and the embedded views focuses on visually associating the nodes in the tree with the graphical objects in their view portions, such as dots in a scatterplot or bars in a bar chart. To refine the design, we first need to know which visual channels are already used and how effective those visual channels are. In Figure 8.9, the table for the host view shows the used visual channels, the corresponding aspect likeness, and the degree of likeness. The embedded view table shows a target set of embedded views. Based on the design guidelines from the visual correspondence theory, several pairs of visual channels can be chosen to establish or enhance visual correspondence. For example, we can utilize shape in the host view and embedded map views to show the relationship between the leaf nodes and the components in the map. The other two examples similarly apply visual correspondence to suggest refinements to visualizations at different levels in PatternTree.

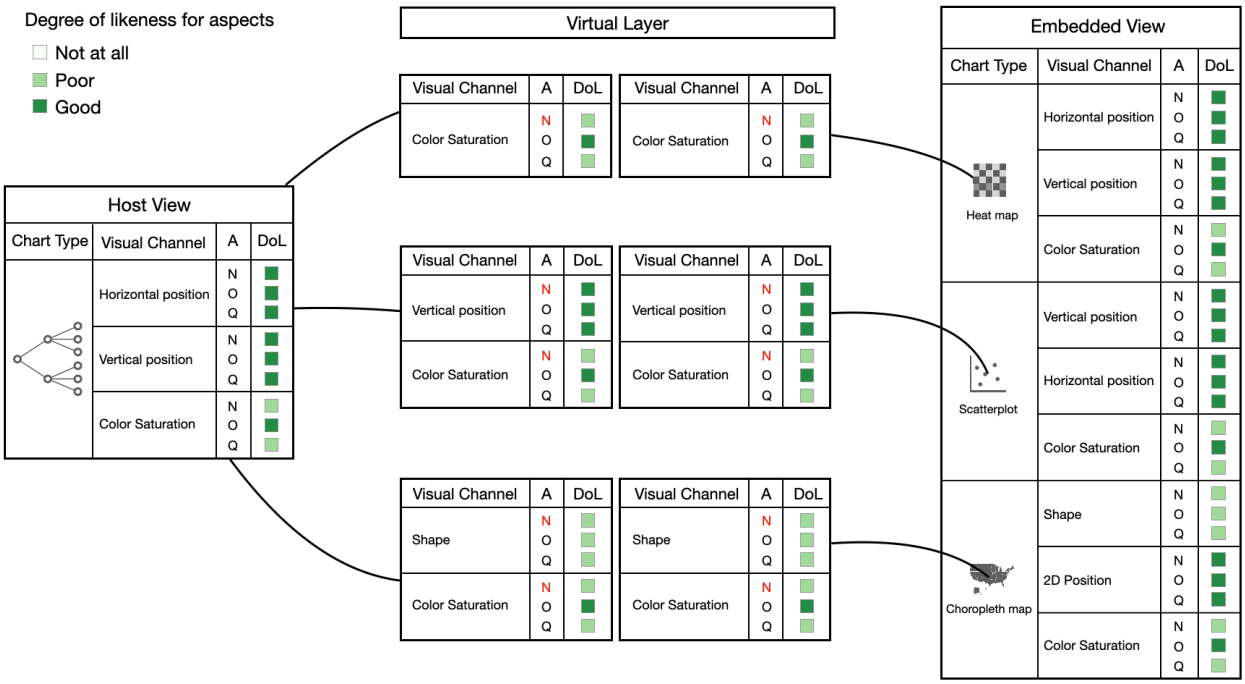


Figure 8.9: An example of refining choices of host and embedded visualizations in PatternTree based on the theory of visual correspondence.

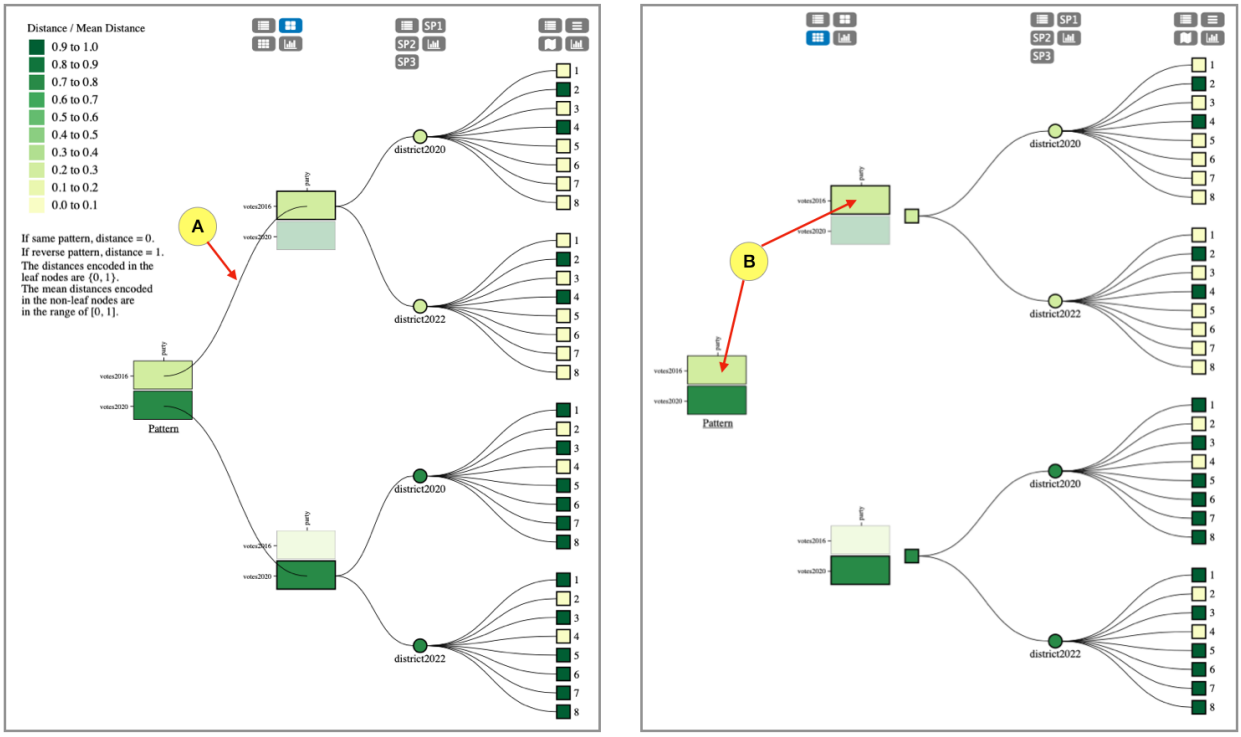


Figure 8.10: Designs for embedding a heat map in the first level of a PatternTree. (A) Visual linking connects heatmap cells (B) Positions and colors of cells provides sufficient visual correspondence without the drawbacks of visual linking.

Figure 8.10 shows two approaches to embed a heatmap into the first level of the tree. Figure 8.10 (A) shows the original design before applying visual correspondence. Drawing connecting lines explicitly reveals the relationship between graphical objects in different tree levels. The view portions implicitly represent node identity. In Figure 8.10 (B), we take advantage of visual correspondence. We remove the connecting lines from the heat map in the root to the heat maps in the first level. The same shape indicates that the two heat maps represent the same information. Moreover, color and position of each highlighted cell in the two heatmaps indicate their relationship to the corresponding cells in the root heatmap. In addition, squares explicitly show the child identity portion to the right of each embedded heatmap. The visual correspondence for a one-to-one relationship between the child identity nodes in the first level and the parent identity nodes in the next level is easily established by the same color. As the number of rows and columns in the heat map increases, following

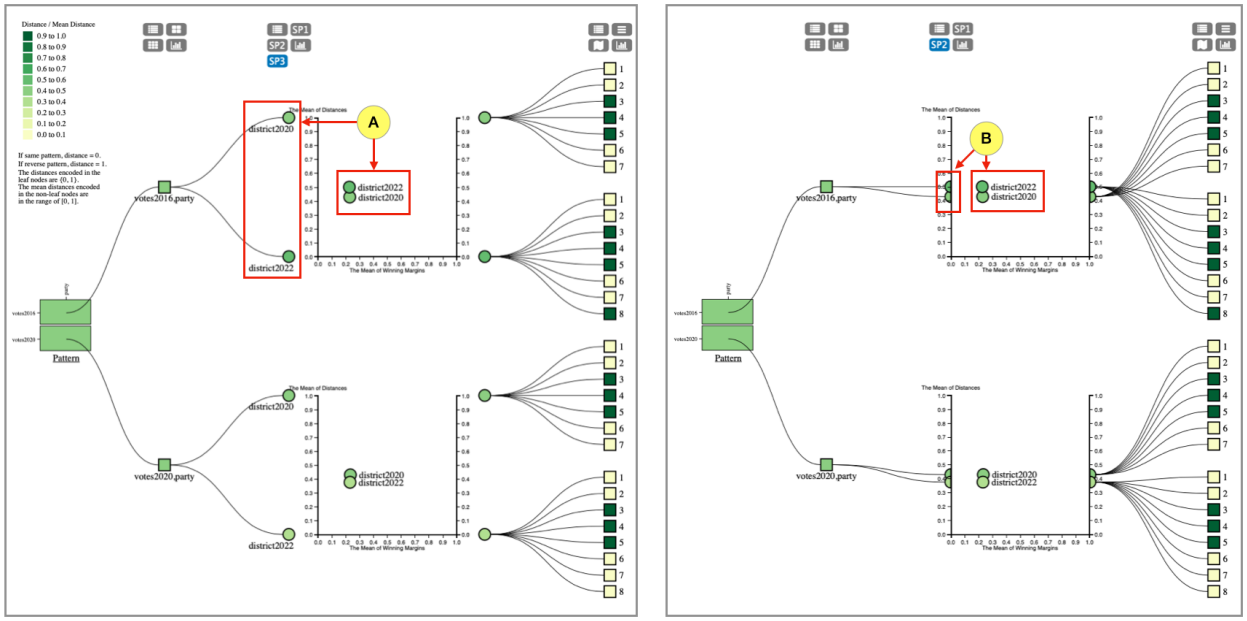


Figure 8.11: Designs for embedding a scatterplot in the second level of the tree visualization. In (A), there is no visual correspondence between the tree nodes and the dots of the scatterplot in the original design. In (B), visual correspondence is established by pairing of vertical position channels.

the paths of individual connecting lines would become more difficult. The new design avoids this issue.

Figure 8.11 shows how a scatterplot can be embedded into the second level of the tree visualization. Figure 8.11 (A) shows the design before we apply the visual correspondence theory. There is no correspondence between the tree nodes and the dots in the scatterplot (ignoring the symbolic correspondence of their labels). In the design shown in Figure 8.11 (B), vertical position is used to establish the correspondence. Consequently, it is easy to perceive the visual correspondence between the tree nodes and the dots in the scatterplot. We also add a right hand axis to the scatterplot, and duplicate the tree nodes of the second level along that axis. The same position for the nodes on the left axis and on the right axis indicates that they encode the same identity information about nodes in that level. Although a pair of vertical channels generally works well when the data points are spread out vertically, the visual correspondence could be diminished if the dots in the scatterplots are too close to each other. In this way, the embedded view type and the distribution of data

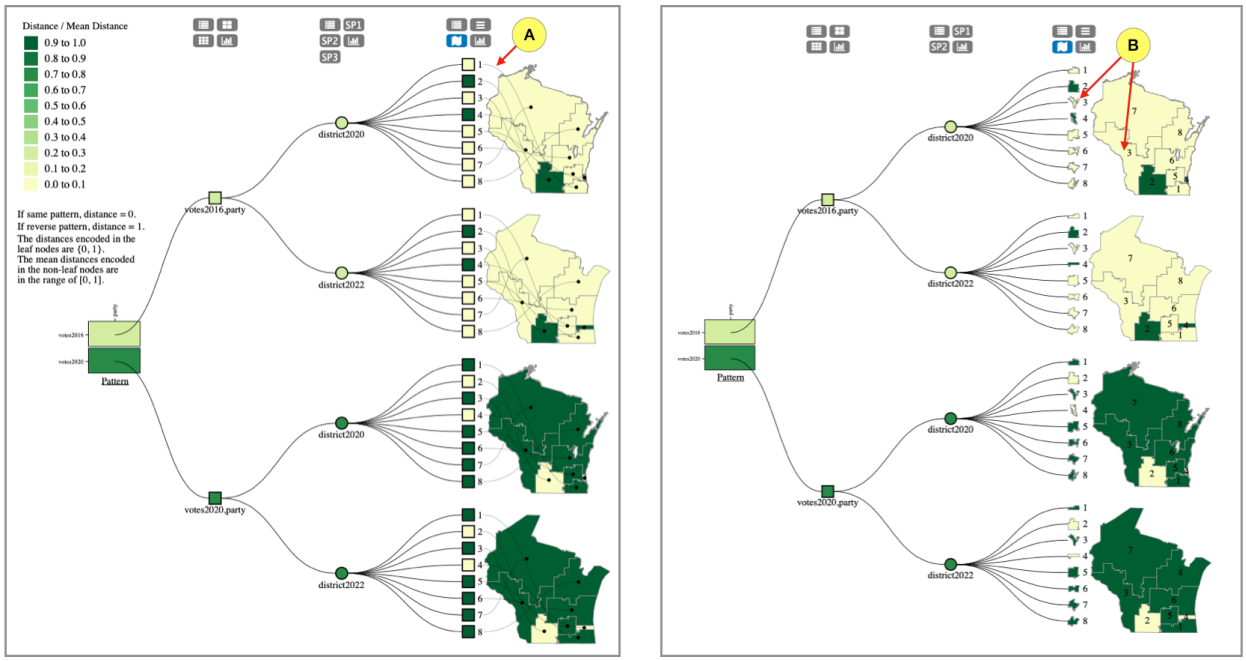


Figure 8.12: Designs for embedding a map view in the leaf level. (A) The original design uses connecting lines to establish correspondence. (B) A pair of shape channels is used for visual correspondence.

in it can have an impact on visual correspondence when using a pairing of position channels (or otherwise).

Figure 8.12 shows how a map view can be embedded into the leaf level. Figure 8.12 (A) is the original design that uses connecting lines to establish correspondence between the leaf nodes and the districts in the map. Figure 8.12 (B) is the design after applying the theory of visual correspondence; color saturation and shape are both viable ways to provide or enhance visual correspondence. Although pairing of shape channels is suitable for use with maps and can work well for enhancing correspondence, there exist limitations. First, the number of shapes and one's ability to distinguish them has an influence on visual correspondence. When we apply PatternTree to states having more than 20 congressional districts (e.g., California, Texas, Florida), it is hard to recognize and distinguish the corresponding shapes. The cardinality of data plays a crucial role in visual correspondence in general. Moreover, visual channels often interact with each other, and the visual correspondence for any given visual channel can be affected by other visual channels. For example, because size can affect the

perception of shape, it is harder to perceive correspondence when the congressional districts are drawn very small, and even more so for districts that are geographically small to begin with, as in the most populous cities. More sophisticated approaches are needed to tackle these issues.

8.10 Limitations and Future Work

In this section, we discuss the limitations of our current system and future directions.

8.10.1 Design Space

There are abundant ways to visualize hierarchical data as tree visualizations. Exploring different aspects of the design space benefits in understanding different approaches, and helps to absorb the strengths and weaknesses. In this work, we only consider a node-link diagram as the base representation for the hierarchical data, and our hybrid tree visualization inherits limitations and drawbacks from the explicit tree visualization, such as low space utilization efficiency. Some techniques among the treemap variations not only are highly space-efficient graphical representations but also highlight the hierarchical structure, for example, *Cascaded Treemaps* [92]. To explore the new layout, we may take into account different design spaces of composite visualization. Our approach in *PatternTree* nests one or more charts inside the node-link diagram based on the relationship between the nodes in node-link diagram and the visual elements in the chart. In the future, we plan to investigate other design options using composite visualization for enhancing various types of tree visualization, for example, using juxtaposition for Cascaded Treemaps. Moreover, node-link tree visualizations treat nodes as discrete entities, but it is possible that the next level of detail contains a set of ranges based on a continuous variable. The design of the visual representation for the continuous leaf nodes and their corresponding view needs further exploration and study. In addition, applying the virtual layering technique to other graph-structured data and visualizations

leads to directions on how to extend and generalize the virtual layering technique in the future.

8.10.2 Scalability

The node-link layout tree visualization suffers from its low space utilization efficiency. Our *PatternTree* inherits the downside of the node-link layout when there exist a large number of branches and leaves. The number of branches grows when the number of patterns increases since each pattern is a branch. Also, if there are more leaves, it will cost more vertical screen space. For example, large states, such as Texas or California, have much more congressional districts than other states. Consequently, low space efficiency could have a detrimental effect on comparison tasks, especially when users need to scroll vertically to compare congressional districts across different districting cycles and it generates an excess of the comparative burden on users' memory. Future work could design interaction techniques to increase the space-efficiency. For example, using an ellipsis to omit less essential leaf nodes could save the vertical space at the initial stage, then interactions activate the collapse or expansion of the leaves.

8.11 Summary

We have introduced *PatternTree* as a hybrid tree visualization via a virtual layering technique to explore hierarchical patterns. We presented a formal description of a design space for embedding the virtual layering in a node-link tree visualization. We illustrated the usefulness of the features of our hierarchical pattern structure by applying it to the gerrymandering domain. In two use cases, we demonstrated that *PatternTree* allows users to gain new insights about gerrymandering on presidential election data. Our study suggests that *PatternTree* enhances not only exploration of hierarchical patterns, but also understanding of complex relationships in the underlying data. Finally, we discussed limitations and future directions.

In the near future, we plan to conduct a usability study of the effectiveness of PatternTree for analysis of gerrymandering.

Chapter 9

Conclusion

The new visualization capabilities described in this dissertation together support interactive exploration of data to detect patterns that reveal mix effects. The suite of features implemented in Wiggum is designed as a modular framework to allow each individual component to be modified or extended. Data analysts can extend the system by adding new statistics features via simple module insertion. Visualization researchers can use the system as a platform for exploring patterns of patterns in visualizations that utilize various types of compositional techniques. Taken together, the contributions of this dissertation open up new paths to grow the complexity and capabilities of tools for exploring statistical anomalies visually.

9.1 Contributions

This dissertation presents the following thesis: *Combining composite visualizations for drill down exploration with interactions into a visualization system supports effective comprehension and analysis of statistical anomalies in multidimensional data for the user with basic statistical knowledge. Adding capabilities to drill down into dimensional combinations hierarchically allows the user to see correspondences between dimensional information visually. These capabilities allow the user to more readily discover statistical anomalies and explore*

their relationships in multidimensional data. The research described in this dissertation makes the following contributions:

- *view designs* for understanding and efficiently detecting mix effects and Simpson's paradox;
- a Python *library* of statistical computations and methods for detecting mix effects;
- an *implementation* of a *visual analysis application*, Wiggum, that integrates the library with a JavaScript-powered visual analytics user interface in the browser;
- an *evaluation* of users' ability to comprehend and apply statistical concepts in the views in the user interface;
- a *conceptual model* for effectively associating graphical items which encode the same data through pairwise visual channels;
- a set of *design guidelines* for establishing and evaluating visual correspondence; and
- a *design space* for embedding views in a node-link tree visualization for smooth visual transitions.

These contributions validate the effectiveness of the visualization techniques and Wiggum for detecting statistical anomalies by demonstrating learnability, flexibility, and general applicability of the statistical methods for exploratory visual data analysis. Wiggum has been used to support useful statistical analysis in several data domains. As mix effects become increasingly important to study for problems like gerrymandering, new computational methods and visualization techniques can be integrated into Wiggum as extensions to the work described in this dissertation. Wiggum and the models of compositional design can also serve as a platform to conduct future research bridging statistical analysis and visualization.

9.2 Future Work

9.2.1 Summary Statistics

Wiggum presents a computational framework that enables the detection and examination of mix effects. We use the term *pattern* to refer to any parametric relationship between a set of variables. For example, a line is a pattern with slope and intercept parameters. Wiggum currently implements patterns for linear regression, Pearson correlation, rank by mean/percentage, and supports exploration and comparison of such patterns at different levels of analysis. Formalization of patterns of these various types would help to generalize and extend current computational capabilities. The development of theory and interactive tools for pattern discovery promises to push forward research in many different domains. With an increasingly generalized framework, we can apply Wiggum more broadly to meaningful real world problems (e.g., gerrymandering, medical data, fairness) with impact on inspiring research in new areas.

9.2.2 Tree Visualization

Although PatternTree focuses on node-link tree visualizations, there are a rich variety of other techniques for visualizing hierarchical data that could serve as the host visualization. The space efficiency of space-filling techniques such as treemaps might help to address limitations and drawbacks of node-link techniques. Other tree visualization layouts introduce new constraints to the design spaces of composite visualization to explore. Moreover, since a tree is a special case of a graph, broader consideration of graph-structured data and visualizations will open up directions to extend and generalize the visual correspondence and virtual layering models.

9.2.3 Evaluation

The model of visual correspondence described in Chapter 7 focuses on only a few perceptual factors to ground predictions for visualization design. The effectiveness of visual encoding channels requires careful development and application of methods to assess human perceptual capabilities. Studying a variety of subjects could help to gain insight into how people establish and process visual correspondence. The intersection of eye tracking with visualization research may reveal patterns when users perform comparison tasks designed for visual correspondence. Furthermore, the relation between the quality of visual encoding channels and the quality of visual correspondence remains to be determined. Evaluation is needed to identify factors in visual effectiveness. Patterns observed through empirical studies can serve to clarify the rankings of visual channels and help to establish new guidelines for visualization designers and developers of manual and automatic visualization systems to create more perceptually effective visualizations.

9.3 Conclusions

The principle goal of this dissertation has been to develop and assess a visual analysis approach in which complex statistical capabilities can be made accessible to wider groups of users with a basic knowledge of statistics. The implementation and evaluation of the Wiggum visual analysis system indicate substantial progress toward achieving this goal. This work opens up a myriad of directions for future research on the way to greater data democratization.

Bibliography

[1] All About Redistricting. `https://redistricting.lls.edu`.

[2] Dave's Redistricting App. `https://davesredistricting.org/maps#home`.

[3] Flask web framework. `http://flask.ocoo.org`.

[4] Ngrok: secure introspectable tunnels to localhost. `https://ngrok.com`.

[5] Yongsu Ahn and Yu-Ru Lin. Fairsight: Visual analytics for fairness in decision making. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1086–1095, 2019.

[6] Hayder M Al-maneea and Jonathan C Roberts. Towards quantifying multiple view layouts in visualisation as seen from research publications. In *2019 IEEE Visualization Conference (VIS)*, pages 121–125. IEEE, 2019.

[7] Nazanin Alipourfard, Peter G Fennell, and Kristina Lerman. Can you trust the trend?: Discovering Simpson's paradoxes in social data. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 19–27. ACM, 2018.

[8] Nazanin Alipourfard, Peter G Fennell, and Kristina Lerman. Using Simpson's paradox to discover interesting patterns in behavioral data. In *Twelfth International AAAI Conference on Web and Social Media*, 2018.

[9] Robert Amar, James Eagan, and John Stasko. Low-level components of analytic activity in information visualization. In *Proceedings of the IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, pages 111–117. IEEE, 2005.

[10] Brian Amos, Michael P McDonald, and Russell Watkins. When boundaries collide: Constructing a national database of demographic and voting statistics. *Public Opinion Quarterly*, 81(S1):385–400, 2017.

[11] Onyebuchi A Arah. The role of causal reasoning in understanding Simpson's paradox, Lord's paradox, and the suppression effect: covariate selection in the analysis of observational studies. *Emerging Themes in Epidemiology*, 5(1):5, 2008.

[12] Daniel Archambault, Tamara Munzner, and David Auber. Grouseflocks: Steerable exploration of graph hierarchy space. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):900–913, 2008.

[13] Zan Armstrong and Martin Wattenberg. Visualizing statistical mix effects and Simpson's paradox. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2132–2141, 2014.

[14] Prasanta S Bandyoapdhyay, Davin Nelson, Mark Greenwood, Gordon Brittan, and Jesse Berwald. The logic of Simpson's paradox. *Synthese*, 181(2):185–208, 2011.

[15] Rachel KE Bellamy, Kuntal Dey, Michael Hind, Samuel C Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilović, et al. AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias. *IBM Journal of Research and Development*, 63(4/5):4–1, 2019.

[16] Jürgen Bernard, Nils Wilhelm, Björn Krüger, Thorsten May, Tobias Schreck, and Jörn Kohlhammer. MotionExplorer: Exploratory search in human motion capture data based on hierarchical aggregation. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2257–2266, 2013.

[17] Jacques Bertin. *Semiology of graphics*. University of Wisconsin Press, 1983.

[18] Peter J Bickel, Eugene A Hammel, J William O'Connell, et al. Sex bias in graduate admissions: Data from Berkeley. *Science*, 187(4175):398–404, 1975.

[19] Sarah Bird, Miro Dudík, Richard Edgar, Brandon Horn, Roman Lutz, Vanessa Milan, Mehrnoosh Sameki, Hanna Wallach, and Kathleen Walker. Fairlearn: A toolkit for assessing and improving fairness in AI. *Microsoft, Tech. Rep. MSR-TR-2020-32*, 2020.

[20] Colin R Blyth. On Simpson's paradox and the sure-thing principle. *Journal of the American Statistical Association*, 67(338):364–366, 1972.

[21] Michelle A Borkin, Azalea A Vo, Zoya Bylinskii, Phillip Isola, Shashank Sunkavalli, Aude Oliva, and Hanspeter Pfister. What makes a visualization memorable? *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2306–2315, 2013.

[22] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.

[23] Matthew Brehmer, Jocelyn Ng, Kevin Tate, and Tamara Munzner. Matches, mismatches, and methods: Multiple-view workflows for energy portfolio analysis. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):449–458, 2015.

[24] Cynthia A Brewer. Guidelines for use of the perceptual dimensions of color for mapping and visualization. In *Color Hard Copy and Graphic Arts III*, volume 2171, pages 54–64. International Society for Optics and Photonics, 1994.

[25] Andreas Buja, Dianne Cook, and Deborah F Swayne. Interactive high-dimensional data visualization. *Journal of Computational and Graphical Statistics*, 5(1):78–99, 1996.

[26] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on Fairness, Accountability and Transparency*, pages 77–91. PMLR, 2018.

[27] Michael Burch, Fabian Beck, and Stephan Diehl. Timeline trees: visualizing sequences of transactions in information hierarchies. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 75–82, 2008.

[28] Ángel Alexander Cabrera, Will Epperson, Fred Hohman, Minsuk Kahng, Jamie Morgenstern, and Duen Horng Chau. FairVis: Visual analytics for discovering intersectional bias in machine learning. In *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 46–56. IEEE, 2019.

[29] Stuart K Card and Jock Mackinlay. The structure of the information visualization design space. In *Proceedings of VIZ'97: Visualization Conference, Information Visualization Symposium and Parallel Rendering Symposium*, pages 92–99. IEEE, 1997.

[30] Clive R Charig, David R Webb, Stephen Richard Payne, and John E Wickham. Comparison of treatment of renal calculi by open surgery, percutaneous nephrolithotomy, and extracorporeal shockwave lithotripsy. *British Medical Journal (Clin Res Ed)*, 292(6524):879–882, 1986.

[31] Aiyou Chen, Thomas Bengtsson, and Tin Kam Ho. A regression paradox for linear models: Sufficient conditions and relation to Simpson's paradox. *The American Statistician*, 63(3):218–225, 2009.

[32] Jowei Chen and Jonathan Rodden. Unintentional gerrymandering: Political geography and electoral bias in legislatures. *Quarterly Journal of Political Science*, 8(3):239–269, 2013.

[33] Xi Chen, Wei Zeng, Yanna Lin, Hayder Mahdi Ai-Maneea, Jonathan Roberts, and Remco Chang. Composition and configuration patterns in multiple-view visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1514–1524, 2020.

[34] Arjun Choudhry, Mandar Sharma, Pramod Chundury, Thomas Kapler, Derek WS Gray, Naren Ramakrishnan, and Niklas Elmqvist. Once upon a time in visualization: Understanding the use of textual narratives for causality. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1332–1342, 2020.

[35] David HS Chung, Daniel Archambault, Rita Borgo, Darren J Edwards, Robert S Laramee, and Min Chen. How ordered is it? on the perceptual orderability of visual channels. *Computer Graphics Forum*, 35(3):131–140, 2016.

[36] Jarry HT Claessen and Jarke J Van Wijk. Flexible linked axes for multivariate data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2310–2316, 2011.

[37] William S Cleveland and Robert McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association*, 79(387):531–554, 1984.

[38] Christopher Collins and Sheelagh Carpendale. VisLink: Revealing relationships amongst visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1192–1199, 2007.

[39] Nathan Conklin, Sandeep Prabhakar, and Chris North. Multiple foci drill-down through tuple and attribute aggregation polyarchies in tabular data. In *Proceedings of the IEEE Symposium on Information Visualization, 2002. INFOVIS 2002*, pages 131–134. IEEE, 2002.

[40] Kate Crawford. The trouble with bias. In *Conference on Neural Information Processing Systems, Keynote*, Long Beach, CA, 2017.

[41] Anamaria Crisan, Shannah Elizabeth Fisher, Jennifer L Gardy, and Tamara Munzner. GEViTRec: Data reconnaissance through recommendation using a domain-specific visualization prevalence design space. *IEEE Transactions on Visualization and Computer Graphics*, 28(12):4855–4872, 2021.

[42] Anamaria Crisan, Jennifer L Gardy, and Tamara Munzner. A systematic method for surveying data visualizations and a resulting genomic epidemiology visualization typology: Gevit. *Bioinformatics*, 35(10):1668–1676, 2019.

[43] Andrew Crotty, Alex Galakatos, Emanuel Zgraggen, Carsten Binnig, and Tim Kraska. Vizdom: interactive analytics through pen and touch. *Proceedings of the VLDB Endowment*, 8(12):2024–2027, 2015.

[44] Russell Davis, Xiaoying Pu, Yiren Ding, Brian D Hall, Karen Bonilla, Mi Feng, Matthew Kay, and Lane Harrison. The risks of ranking: Revisiting graphical perception to model individual differences in visualization performance. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–16, 2022.

[45] Steven M Day. Simpson's paradox and Major League Baseball's Hall of Fame. *Red*, 4:8, 1994.

[46] Daryl R DeFord, Nicholas Eubank, and Jonathan Rodden. Partisan dislocation: A precinct-level measure of representation and gerrymandering. *Political Analysis*, pages 1–23, 2021.

[47] Çağatay Demiralp, Michael S Bernstein, and Jeffrey Heer. Learning perceptual kernels for visualization design. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1933–1942, 2014.

[48] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 214–226, 2012.

[49] Barry Edwards, Michael Crespin, Ryan D Williamson, and Maxwell Palmer. Institutional control of redistricting and the geography of representation. *The Journal of Politics*, 79(2):722–726, 2017.

[50] Niklas Elmqvist and Jean-Daniel Fekete. Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):439–454, 2009.

[51] Niklas Elmqvist and Philippas Tsigas. Causality visualization using animated growing polygons. In *Proceedings of the IEEE Symposium on Information Visualization 2003 (IEEE Cat. No. 03TH8714)*, pages 189–196. IEEE, 2003.

[52] Robert S Erikson. Malapportionment, gerrymandering, and party fortunes in congressional elections. *American Political Science Review*, 66(4):1234–1245, 1972.

[53] Carem C Fabris and Alex A Freitas. Discovering surprising patterns by detecting occurrences of Simpson's paradox. In *Research and Development in Intelligent Systems XVI*, pages 148–160. Springer, 2000.

[54] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of Human Genetics*, 7(2):179–188, 1936.

[55] Alex A Freitas. On objective measures of rule surprisingness. In *European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 1–9. Springer, 1998.

[56] Yaniv Frishman and Ayellet Tal. Dynamic drawing of clustered graphs. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 191–198. IEEE, 2004.

[57] Wojciech Froelich. Mining association rules from database tables with the instances of Simpson's paradox. In *Advances in Databases and Information Systems*, pages 79–90. Springer, 2013.

[58] Laura Garrison, Juliane Müller, Stefanie Schreiber, Steffen Oeltze-Jafra, Helwig Hauser, and Stefan Bruckner. DimLift: Interactive hierarchical data exploration through dimensional bundling. *IEEE Transactions on Visualization and Computer Graphics*, 27(6):2908–2922, 2021.

[59] Nils Gehlenborg and Bang Wong. Networks: we describe graphing techniques to support exploration of networks. *Nature Methods*, 9(2):115–116, 2012.

[60] Michael Gleicher. Considerations for visualizing comparison. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):413–423, 2017.

[61] Michael Gleicher, Danielle Albers, Rick Walker, Ilir Jusufi, Charles D Hansen, and Jonathan C Roberts. Visual comparison for information visualization. *Information Visualization*, 10(4):289–309, 2011.

[62] Samuel Gratzl, Nils Gehlenborg, Alexander Lex, Hanspeter Pfister, and Marc Streit. Domino: Extracting, comparing, and manipulating subsets across multiple tabular datasets. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2023–2032, 2014.

[63] Samuel Gratzl, Alexander Lex, Nils Gehlenborg, Hanspeter Pfister, and Marc Streit. LineUp: Visual analysis of multi-attribute rankings. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2277–2286, 2013.

[64] John Alexis Guerra-Gomez, Michael L Pack, Catherine Plaisant, and Ben Shneiderman. Visualizing change over time using dynamic hierarchies: Treeversity2 and the StemView. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2566–2575, 2013.

[65] Yue Guo, Carsten Binnig, and Tim Kraska. What you see is not what you get!: Detecting Simpson's paradoxes during data exploration. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics*, page 2. ACM, 2017.

[66] Wes Gurnee and David B Shmoys. Fairmandering: A column generation heuristic for fairness-optimized political districting. In *SIAM Conference on Applied and Computational Discrete Algorithms (ACDA21)*, pages 88–99. SIAM, 2021.

[67] Steffen Hadlak, Heidrun Schumann, and Hans-Jörg Schulz. A survey of multi-faceted graph visualization. In *EuroVis (STARs)*, pages 1–20, 2015.

[68] Jeffrey Heer and Michael Bostock. Crowdsourcing graphical perception: using Mechanical Turk to assess visualization design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 203–212, 2010.

[69] Jeffrey Heer, Nicholas Kong, and Maneesh Agrawala. Sizing the horizon: the effects of chart size and layering on the graphical perception of time series visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 1303–1312, 2009.

[70] John A Henderson, Brian T Hamel, and Aaron M Goldzimer. Gerrymandering incumbency: does nonpartisan redistricting increase electoral competition? *The Journal of Politics*, 80(3):1011–1016, 2018.

[71] Miguel A Hernán, David Clayton, and Niels Keiding. The Simpson's paradox unraveled. *International Journal of Epidemiology*, 40(3):780–785, 2011.

[72] Kenneth Holstein, Jennifer Wortman Vaughan, Hal Daumé III, Miro Dudik, and Hanna Wallach. Improving fairness in machine learning systems: What do industry practitioners need? In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–16, 2019.

[73] Danny Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006.

[74] Nicole Jardine, Brian D Ondov, Niklas Elmqvist, and Steven Franconeri. The perceptual proxies of visual comparison. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1012–1021, 2019.

[75] Waqas Javed and Niklas Elmqvist. Exploring the design space of composite visualization. In *Proceedings of the 2012 IEEE Pacific Visualization Symposium*, pages 1–8, Songdo, South Korea, 2012. IEEE.

[76] Zhuochen Jin, Shunan Guo, Nan Chen, Daniel Weiskopf, David Gotz, and Nan Cao. Visual causality analysis of event sequence data. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1343–1352, 2020.

[77] Steven A Julious and Mark A Mullee. Confounding and Simpson's paradox. *British Medical Journal*, 309(6967):1480–1481, 1994.

[78] Johannes Kehrer and Helwig Hauser. Visualization and visual analysis of multifaceted scientific data: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 19(3):495–513, 2012.

[79] Daniel A Keim and Daniela Oelke. Literature fingerprinting: A new method for visual literary analysis. In *Proceedings of the 2007 IEEE Symposium on Visual Analytics Science and Technology*, pages 115–122. IEEE, 2007.

[80] Rogier A Kievit, Willem E Frankenhuis, Lourens J Waldorp, and Denny Borsboom. Simpson's paradox in psychological science: a practical guide. *Frontiers in psychology*, 4, 2013.

[81] Younghoon Kim and Jeffrey Heer. Assessing effects of task and data distribution on the effectiveness of visual encodings. *Computer Graphics Forum*, 37(3):157–167, 2018.

[82] Younghoon Kim, Kanit Wongsuphasawat, Jessica Hullman, and Jeffrey Heer. GraphScape: A model for automated reasoning about visualization similarity and sequencing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 2628–2638, 2017.

[83] Kurt Koffka. *Principles of Gestalt psychology*. Routledge, 2013.

[84] Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. *Advances in Neural Information Processing Systems*, 30, 2017.

[85] Heidi Lam, Melanie Tory, and Tamara Munzner. Bridging from goals to tasks with design study analysis reports. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):435–445, 2017.

[86] Ricardo Langner, Ulrike Kister, and Raimund Dachselt. Multiple coordinated views at large displays for multiple users: Empirical findings on user behavior, movements, and distances. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):608–618, 2018.

[87] Kristina Lerman. Computational social scientist beware: Simpson's paradox in behavioral data. *Journal of Computational Social Science*, pages 1–10, 2017.

[88] Harry A Levin and Sorelle A Friedler. Automated congressional redistricting. *Journal of Experimental Algorithmics (JEA)*, 24:1–24, 2019.

[89] Justin Levitt. A citizen's guide to redistricting. *Available at SSRN 1647221*, 2008. `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1647221`.

[90] M. Lichman. UCI machine learning repository, 2013.

[91] Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

[92] Hao Lü and James Fogarty. Cascaded treemaps: examining the visibility and stability of structure in treemaps. In *Proceedings of Graphics Interface 2008*, pages 259–266, 2008.

[93] Sehi LYi, Jaemin Jo, and Jinwook Seo. Comparative layouts revisited: Design space, guidelines, and future directions. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1525–1535, 2020.

[94] Jock Mackinlay. Applying a theory of graphical presentation to the graphic design of user interfaces. In *Proceedings of the 1st annual ACM SIGGRAPH Symposium on User Interface Software*, pages 179–189, 1988.

[95] Jock Mackinlay, Pat Hanrahan, and Chris Stolte. Show Me: Automatic presentation for visual analysis. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1137–1144, 2007.

[96] Justin Matejka and George Fitzmaurice. Same stats, different graphs: generating datasets with varied appearance and identical statistics through simulated annealing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 1290–1294, 2017.

[97] Benedikt Mayer, Kai Lawonn, Karsten Donnay, Bernhard Preim, and Monique Meuschke. Vehicle: Validation and exploration of the hierarchical integration of conflict event data. *Computer Graphics Forum*, 40(3):1–12, 2021.

[98] Caitlyn M McColeman, Fumeng Yang, Timothy F Brady, and Steven Franconeri. Rethinking the ranks of visual channels. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):707–717, 2021.

[99] Michael McGuffin. Visualization Course Figures. `https://www.michaelmcguffin.com/courses/vis/`, 2014.

[100] Michael J McGuffin. Simple algorithms for network visualization: A tutorial. *Tsinghua Science and Technology*, 17(4):383–398, 2012.

[101] Tamara Munzner. *Visualization analysis and design*. CRC press, 2014.

[102] Floyd Norris. Median pay in us is stagnant, but low-paid workers lose. *New York Times*, Apr. 26, 2013.

[103] Christine Nothelfer and Steven Franconeri. Measures of the benefit of direct encoding of data deltas for data pair relation perception. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):311–320, 2019.

[104] Brian Ondov, Nicole Jardine, Niklas Elmqvist, and Steven Franconeri. Face to face: Evaluating visual comparison. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):861–871, 2018.

[105] Donovan H Parks and Robert G Beiko. Quantitative visualizations of hierarchically organized data in a geographic context. In *2009 17th International Conference on Geoinformatics*, pages 1–6. IEEE, 2009.

[106] Donovan H Parks, Michael Porter, Sylvia Churcher, Suwen Wang, Christian Blouin, Jacqueline Whalley, Stephen Brooks, and Robert G Beiko. GenGIS: A geospatial information system for genomic data. *Genome Research*, 19(10):1896–1904, 2009.

[107] Marios G Pavlides and Michael D Perlman. How likely is Simpson's paradox? *The American Statistician*, 63(3):226–233, 2009.

[108] Judea Pearl. *Causality*. Cambridge University Press, 2009.

[109] Judea Pearl. Simpson's paradox: An anatomy. *Department of Statistics, UCLA*, 2011.

[110] Judea Pearl. Comment: Understanding Simpson's paradox. *The American Statistician*, 68(1):8–13, 2014.

[111] Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic Books, 2018.

[112] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[113] Zening Qu and Jessica Hullman. Evaluating visualization sets: Trade-offs between local effectiveness and global consistency. In *Proceedings of the Sixth Workshop on Beyond Time and Errors on Novel Evaluation Methods for Visualization*, pages 44–52, 2016.

[114] Zening Qu and Jessica Hullman. Keeping multiple views consistent: Constraints, validations, and exceptions in visualization authoring. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):468–477, 2017.

[115] J Ross Quinlan. Combining instance-based and model-based learning. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 236–243, Amherst, MA, USA, 1993.

[116] Jonathan C Roberts. Multiple view and multiform visualization. In *Visual Data Exploration and Analysis VII*, volume 3960, pages 176–185. SPIE, 2000.

[117] Jonathan C Roberts. State of the art: Coordinated & multiple views in exploratory visualization. In *Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV 2007)*, pages 61–71, Zurich, Switzerland, 2007. IEEE.

[118] Gerta Rücker and Martin Schumacher. Simpson's paradox visualized: the example of the Rosiglitazone meta-analysis. *BMC Medical Research Methodology*, 8(1):34, 2008.

[119] Debjani Saha, Candice Schumann, Duncan Mcelfresh, John Dickerson, Michelle Mazurek, and Michael Tschantz. Measuring non-expert comprehension of machine learning fairness metrics. In *International Conference on Machine Learning*, pages 8377–8387. PMLR, 2020.

[120] Pedro Saleiro, Benedict Kuester, Loren Hinkson, Jesse London, Abby Stevens, Ari Anisfeld, Kit T Rodolfa, and Rayid Ghani. Aequitas: A bias and fairness audit toolkit. *arXiv preprint arXiv:1811.05577*, 2018.

[121] Babak Salimi, Corey Cole, Peter Li, Johannes Gehrke, and Dan Suciu. HypDB: a demonstration of detecting, explaining and resolving bias in OLAP queries. *Proceedings of the VLDB Endowment*, 11(12):2062–2065, 2018.

[122] Babak Salimi, Johannes Gehrke, and Dan Suciu. Bias in OLAP queries: Detection, explanation, and removal. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1021–1035. ACM, 2018.

[123] Sarah Schöttler, Yalong Yang, Hanspeter Pfister, and Benjamin Bach. Visualizing and interacting with geospatial networks: A survey and design space. *Computer Graphics Forum*, 40(6):5–33, 2021.

[124] H-J Schulz and Heidrun Schumann. Visualizing graphs-a generalized view. In *Proceedings of the Tenth International Conference on Information Visualisation (IV'06)*, pages 166–173. IEEE, 2006.

[125] Hans-Jorg Schulz. Treevis. net: A tree visualization reference. *IEEE Computer Graphics and Applications*, 31(6):11–15, 2011.

[126] Hans-Jorg Schulz, Steffen Hadlak, and Heidrun Schumann. The design space of implicit hierarchy visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 17(4):393–411, 2010.

[127] Andrew D Selbst, Danah Boyd, Sorelle A Friedler, Suresh Venkatasubramanian, and Janet Vertesi. Fairness and abstraction in sociotechnical systems. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 59–68. ACM, 2019.

[128] Abdul Rahman Shaikh, David Koop, Hamed Alhoori, and Maoyuan Sun. Toward systematic design considerations of organizing multiple views. In *Proceedings of the 2022 IEEE Visualization and Visual Analytics (VIS)*, pages 105–109. IEEE, 2022.

[129] Ben Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on Graphics (TOG)*, 11(1):92–99, 1992.

[130] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, pages 336–343. IEEE, 1996.

[131] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *The craft of information visualization*, pages 364–371. Elsevier, 2003.

[132] Kacper Sokol, Raul Santos-Rodriguez, and Peter Flach. FAT Forensics: A python toolbox for algorithmic fairness, accountability and transparency. *Software Impacts*, 14:100406, 2022.

[133] John Stasko and Eugene Zhang. Focus+ context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In *Proceedings of the IEEE Symposium on Information Visualization 2000. INFOVIS 2000.*, pages 57–65. IEEE, 2000.

[134] Markus Steinberger, Manuela Waldner, Marc Streit, Alexander Lex, and Dieter Schmalstieg. Context-preserving visual links. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2249–2258, 2011.

[135] Nicholas O Stephanopoulos and Eric M McGhee. Partisan gerrymandering and the efficiency gap. *The University of Chicago Law Review*, 82:831, 2015.

[136] Joshua Stevens. Bivariate choropleth maps: A how-to guide. `http://www.joshuastevens.net/cartography/make-a-bivariate-choropleth-map/`, 2015.

[137] Chris Stolte, Diane Tang, and Pat Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):52–65, 2002.

[138] Harini Suresh and John V Guttag. A framework for understanding unintended consequences of machine learning. *preprint arXiv:1901.10002*, 2019.

[139] Florian Tramer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, Jean-Pierre Hubaux, Mathias Humbert, Ari Juels, and Huang Lin. FairTest: Discovering unwarranted associations in data-driven applications. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 401–416. IEEE, 2017.

[140] Bruce E Trumbo. A theory for coloring bivariate statistical maps. *The American Statistician*, 35(4):220–226, 1981.

[141] Yu-Kang Tu, David Gunnell, and Mark S Gilthorpe. Simpson's paradox, Lord's paradox, and suppression effects are the same phenomenon–the reversal paradox. *Emerging Themes in Epidemiology*, 5(1):2, 2008.

[142] Edward Tufte. *Envisioning Information*. Graphics Press, Cheshire, CT, 1990.

[143] Edward R Tufte. The relationship between seats and votes in two-party systems. *American Political Science Review*, 67(2):540–554, 1973.

[144] Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, CT, second edition, 2001.

[145] Stef van den Elzen and Jarke J van Wijk. Small multiples, large singles: A new approach for visual data exploration. *Computer Graphics Forum*, 32(3pt2):191–200, 2013.

[146] Stef Van den Elzen and Jarke J Van Wijk. Multivariate network exploration and presentation: From detail to overview via selections and aggregations. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2310–2319, 2014.

[147] Jarke J Van Wijk and Edward R Van Selow. Cluster and calendar based visualization of time series data. In *Proceedings of the 1999 IEEE Symposium on Information Visualization (InfoVis' 99)*, pages 4–9. IEEE, 1999.

[148] Christophe Viau and Michael J McGuffin. ConnectedCharts: explicit visualization of relationships between data graphics. *Computer Graphics Forum*, 31(3pt4):1285–1294, 2012.

[149] Jun Wang and Klaus Mueller. The visual causality analyst: An interactive interface for causal reasoning. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):230–239, 2015.

[150] Jun Wang and Klaus Mueller. Visual causality analysis made practical. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 151–161. IEEE, 2017.

[151] Qianwen Wang, Zhenhua Xu, Zhutian Chen, Yong Wang, Shixia Liu, and Huamin Qu. Visual analysis of discrimination in machine learning. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1470–1480, 2020.

[152] Xiaozhi Wang, Ziqi Wang, Xu Han, Wangyi Jiang, Rong Han, Zhiyuan Liu, Juanzi Li, Peng Li, Yankai Lin, and Jie Zhou. Maven: A massive general domain event detection dataset. *arXiv preprint arXiv:2004.13590*, 2020.

[153] Michelle Q Wang Baldonado, Allison Woodruff, and Allan Kuchinsky. Guidelines for using multiple views in information visualization. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 110–119, Palermo, Italy, 2000.

[154] Robert L Wardrop. Simpson's paradox and the hot hand in basketball. *The American Statistician*, 49(1):24–28, 1995.

[155] Chris Weaver. Building highly-coordinated visualizations in improvise. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 159–166. IEEE, 2004.

[156] Chris Weaver. Patterns of coordination in Improvise visualizations. In *Proceedings of SPIE (Visualization and Data Analysis)*, pages 1–12, San Jose, CA, January 2007. SPIE.

[157] Chris Weaver. Cross-filtered views for multidimensional visual analysis. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):192–204, March-April 2010.

[158] Chris Weaver. Multidimensional data dissection using attribute relationship graphs. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 75–82, Salt Lake City, UT, October 2010. IEEE.

[159] James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. The what-if tool: Interactive probing of machine learning models. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):56–65, 2019.

[160] Linda Woodburn, Yalong Yang, and Kim Marriott. Interactive visualisation of hierarchical quantitative data: an evaluation. In *2019 IEEE Visualization Conference (VIS)*, pages 96–100. IEEE, 2019.

[161] Eugene Wu. View composition algebra for ad hoc comparison. *IEEE Transactions on Visualization and Computer Graphics*, 28(6):2470–2485, 2022.

[162] Xiao Xie, Fan Du, and Yingcai Wu. A visual analytics approach for exploratory causal analysis: Exploration, validation, and applications. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1448–1458, 2020.

[163] Cindy Xiong, Vidya Setlur, Benjamin Bach, Eunyee Koh, Kylie Lin, and Steven Franconeri. Visual arrangements of bar charts influence comparisons in viewer takeaways. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):955–965, 2021.

[164] Chenguang Xu, Sarah M Brown, and Christan Grant. Detecting Simpson's paradox. In *Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pages 221–224, 2018.

[165] Panpan Xu, Yingcai Wu, Enxun Wei, Tai-Quan Peng, Shixia Liu, Jonathan JH Zhu, and Huamin Qu. Visual analysis of topic competition on social media. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2012–2021, 2013.

[166] Jing Nathan Yan, Ziwei Gu, Hubert Lin, and Jeffrey M Rzeszotarski. Silva: Interactively assessing machine learning fairness using causality. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2020.

[167] Yalong Yang, Tim Dwyer, Sarah Goodwin, and Kim Marriott. Many-to-many geographically-embedded flow visualisation: An evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):411–420, 2016.

[168] Meike Zehlike, Francesco Bonchi, Carlos Castillo, Sara Hajian, Mohamed Megahed, and Ricardo Baeza-Yates. FA*IR: A fair top-k ranking algorithm. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1569–1578, 2017.

[169] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2979–2989, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[170] Shengdong Zhao, Michael J McGuffin, and Mark H Chignell. Elastic hierarchies: Combining treemaps and node-link diagrams. In *Proceedings of the IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, pages 57–64. IEEE, 2005.