ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

THESIS PRESENTED TO
ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
A MASTER'S DEGREE IN INFORMATION TECHNOLOGY ENGINEERING
M.Eng.

BY
Jean-François IM

VISUALIZATION OF LARGE AMOUNTS OF MULTIDIMENSIONAL MULTIVARIATE BUSINESS-ORIENTED DATA

MONTREAL, NOVEMBER 24, 2014

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS:

Mr. Michael J. McGuffin, Memorandum Director
Department of Software and IT Engineering, École de technologie supérieure

Mr. Jean-Marc Robert, Committee President
Department of Software and IT Engineering, École de technologie supérieure

Mr. Abdelouahed Gherbi, Examiner
Department of Software and IT Engineering, École de technologie supérieure

THIS DISSERTATION WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND PUBLIC

ON SEPTEMBER 5, 2014

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGEMENTS

This research would not have been possible without the financial support of SAP's Academic Research Center and NSERC. Special thanks at SAP go to Rock Leung and Julian Gosper for their invaluable feedback during development of the various GPLOM prototypes. Of course, the additional feedback from Qiang Han, Ivailo Ivanov, Alex MacAulay and Alexei Potiagalov was duly appreciated.

I would also like to thank Félix Giguère Villegas and Hisham Mardam-Bey of Mate1.com for their support during the development of VisReduce, without whom the realization of that project would not have been possible.

Further thanks go to my good friends Alexandre Elias and Tennessee Carmel-Veilleux for their feedback on both the GPLOM and VisReduce papers. I have a profound respect for your wide breadth of technical knowledge and appreciated your very insightful comments.

Last, but not least, I would like to thank Michael McGuffin, whose ability to turn raw ideas into publishable papers and deep pool of knowledge about visualization have been invaluable throughout the last two years.

VISUALIZATION OF LARGE AMOUNTS OF MULTIDIMENSIONAL MULTIVARIATE BUSINESS-ORIENTED DATA

Jean-François IM

ABSTRACT

Many large businesses store large amounts of business-oriented data in data warehouses. These data warehouses contain fact tables, which themselves contain rows representing business events, such as an individual sale or delivery. This data contains multiple dimensions (independent variables that are categorical) and very often also contains multiple measures (dependent variables that are usually continuous), which makes it complex for casual business users to analyze and visualize. We propose two techniques, GPLOM and VisReduce, that respectively handle the visualization front-end of complex datasets and the back-end processing necessary to visualize large datasets.

Scatterplot matrices (SPLOMs), parallel coordinates, and glyphs can all be used to visualize the multiple measures in multidimensional multivariate data. However, these techniques are not well suited to visualizing many dimensions. To visualize multiple dimensions, "hierarchical axes" that "stack dimensions" have been used in systems like Polaris and Tableau. However, this approach does not scale well beyond a small number of dimensions.

Emerson *et al.* (2013) extend the matrix paradigm of the SPLOM to simultaneously visualize several categorical and continuous variables, displaying many kinds of charts in the matrix depending on the kinds of variables involved. We propose a variant of their technique, called the Generalized Plot Matrix (GPLOM). The GPLOM restricts Emerson *et al.* (2013)'s technique to only three kinds of charts (scatterplots for pairs of continuous variables, heatmaps for pairs of categorical variables, and barcharts for pairings of categorical and continuous variable), in an effort to make it easier to understand by casual business users. At the same time, the GPLOM extends Emerson *et al.* (2013)'s work by demonstrating interactive techniques suited to the matrix of charts. We discuss the visual design and interactive features of our GPLOM prototype, including a textual search feature allowing users to quickly locate values or variables by name. We also present a user study that compared performance with Tableau and our GPLOM prototype, that found that GPLOM is significantly faster in certain cases, and not significantly slower in other cases.

Also, performance and responsiveness of visual analytics systems for exploratory data analysis of large datasets has been a long standing problem, which GPLOM also encounters. We propose a method called VisReduce that incrementally computes visualizations in a distributed fashion by combining a modified MapReduce-style algorithm with a compressed columnar data store, resulting in significant improvements in performance and responsiveness for constructing commonly encountered information visualizations, e.g., bar charts, scatterplots, heat maps, cartograms and parallel coordinate plots. We compare our method with one that queries three other readily available database and data warehouse systems — PostgreSQL, Cloudera

Impala and the MapReduce-based Apache Hive — in order to build visualizations. We show that VisReduce's end-to-end approach allows for greater speed and guaranteed end-user responsiveness, even in the face of large, long-running queries.

Keywords: scatterplot matrix, SPLOM, generalized plot matrix, GPLOM, mdmv, VisReduce, MapReduce, incremental visualization

VISUALISATION DE JEUX DE DONNÉES D'AFFAIRES MULTIDIMENSIONNELS MULTIVARIÉS VOLUMINEUX

Jean-François IM

RÉSUMÉ

Plusieurs grandes entreprises stockent des volumes importants de données d'affaires dans des entrepôts de données. Ces entrepôts de données contiennent des tables de faits, qui elles mêmes contiennent des rangées représentant des évènements d'affaires, comme une vente ou une livraison. Ces données comprennent plusieurs dimensions (variables indépendantes et catégoriques) et fréquemment plusieurs mesures (variables dépendantes et habituellement continues), ce qui rend ardue la tâche d'analyser et de visualiser ces types de données par des utilisateurs non-experts. Nous proposons deux techniques, GPLOM et VisReduce, qui gèrent respectivement la visualisation de jeux de données complexes et le traitement nécessaire à la visualisation de jeux de données volumineux.

Les matrices de nuages de points (Scatter PLOt Matrices, ou SPLOMs), les coordonnées parallèles et les glyphes peuvent être utilisés pour visualiser plusieurs mesures dans les jeux de données multidimensionnels multivariés. Cependant, ces techniques ne sont pas efficaces pour la visualisation de plusieurs dimensions. Pour visualiser plusieurs dimensions, des axes hiérarchiques qui imbriquent les dimensions ont été utilisés dans des systèmes comme Polaris et Tableau. Cependant, cette approche fonctionne mal lorsqu'appliquée à plus que quelques dimensions.

Emerson *et al.* (2013) étend le paradigme de la SPLOM pour visualiser simultanément plusieurs variables catégoriques et continues, affichant plusieurs types de graphiques dans la matrice selon la combinaison de variables impliquées. Nous proposons une variante de leur technique, appelée la matrice de graphiques généralisée (Generalized PLOt Matrix, ou GPLOM). La GPLOM restreint la technique d'Emerson *et al.* (2013) pour n'utiliser que trois types de graphiques (des nuages de points pour les paires de variables continues, des thermogrammes pour les paires de variables catégoriques et des graphiques à bâtons pour les paires de variables continues et catégoriques) afin de la rendre plus accessible à des utilisateurs non-experts. En même temps, la GPLOM augmente le travail d'Emerson *et al.* (2013) en démontrant des techniques d'interaction appropriées à la matrice de graphiques. Nous discutons du design visuel et des fonctionnalités interactives de notre prototype de la GPLOM, entre autres une fonctionnalité de recherche textuelle qui permet aux utilisateurs de chercher des valeurs et des variables par nom. Nous présentons aussi une expérience contrôlée avec des utilisateurs qui compare la performance de Tableau et de notre prototype de la GPLOM qui démontre que la GPLOM est significativement plus rapide dans certains cas et non significativement plus lente dans d'autres cas.

Aussi, la performance et la rapidité de réponse des systèmes d'analyse visuels pour l'exploration de jeux de données volumineux est un problème connu et identifié comme un problème impor-

tant pour la communauté de visualisation, problème auquel la GPLOM n'échappe pas. Nous proposons alors une technique appelée VisReduce qui calcule une visualisation de façon incrémentale et distribuée en combinant un algorithme similaire à MapReduce avec un engin de stockage compressé orienté colonne, résultant en des améliorations significatives de performance et de temps de réponse pour la construction de graphiques fréquemment utilisés, comme les graphiques à bâtons, les nuages de points, les thermogrammes, les cartogrammes et les graphiques à coordonnées parallèles. Nous comparons notre méthode avec une qui interroge trois systèmes de gestion de bases de données et systèmes d'entrepôts de données statu quo — PostgreSQL, Cloudera Impala et Apache Hive — pour construire des visualisations. Nous démontrons que VisReduce permet une meilleure performance et un temps de réponse garanti, même pour des requêtes volumineuses ayant un long temps d'exécution.

Mot-clés : matrice de nuages de points, SPLOM, matrice de graphiques généralisées, GPLOM, VisReduce, MapReduce, visualisation incrementale

CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

CDH	Cloudera Distribution Including Apache Hadoop
CSV	Comma Separated Values
DBMS	Database Management System
ETL	Extract transform load
GPLOM	Generalized Plot Matrix
HDFS	Hadoop Distributed Filesystem
JIT	Just In Time
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
LTS	Long Term Support
MDMV	Multidimensional multivariate
MOLAP	Multidimensional online analytical processing
OLAP	Online analytical processing
PCP	Parallel Coordinate Plot
RCFile	Record Columnar File
ROLAP	Relational online analytical processing
S&P	Standard and Poor's
SPLOM	Scatterplot Matrix
SQL	Structured Query Language
SSD	Solid State Disk

SVG	Scalable Vector Graphics

INTRODUCTION

Business intelligence is a collection of tools and processes that support business decisions by allowing the various business stakeholders to base their decision process on facts. As businesses collect and store ever increasing amounts of data, they seek to discover more key insights and trends that have been previously hidden in their data.

Information visualization supports this process of discovery, as Heer and Shneiderman (2012) discuss, by helping the user discover patterns and correlations contained within data. Ware (2004) explains that visualization has several advantages:

- Visualization provides an ability to comprehend huge amounts of data
- Visualization allows the perception of unanticipated emergent properties that can often be the basis of a new insight
- Visualization often makes problems within the data immediately apparent
- Visualization facilitates understanding of both large-scale and small-scale features of the data
- Visualization facilitates hypothesis formation

All of these advantages can be linked to the fact that information visualization amplifies cognition by taking advantage of the high bandwidth of the human perceptual system to understand patterns in data. These advantages have been well studied; for example, Table 2.2 of Thomas and Cook (2005) lists many other advantages with references to other work that has been done.

Yet, even with those advantages, there are still hurdles with regards to visualizing and understanding large amounts of data by casual business users. Grammel *et al.* (2010) explores this issue with business school students, asking them to build visualizations to answer questions on business-oriented data sets. They identified three steps that were challenging for users: translating questions into data attributes, constructing visualizations that help to answer these

questions and interpreting the visualizations. For these users, having a system that automatically produces visualizations that are easy to understand would be very helpful.

There are also performance challenges that arise with very large databases, which impact usability. When queries to a database require more than a few seconds to process, such queries cannot be performed in a very interactive manner.

We propose two techniques, GPLOM and VisReduce, that improve the front-end and back-end, respectively, of a database visualization system. Both techniques address a specific need that is currently not fulfilled. GPLOM is a front-end for data with many dimensions and measures that enables visualization with minimal user effort. VisReduce is a processing back-end that supports incremental, distributed visualizations of large datasets. GPLOM and VisReduce can be used together or separately.

GPLOM allows the exploration of *multidimensional multivariate* datasets. In a typical tabular database, often occuring in business intelligence systems and elsewhere, some columns are *independent variables*; these are also called *dimensions* and are typically *categorical* variables (or they are discretized, which reduces them to categorical variables). Other columns are best thought of as *dependent variables*, also called *measures*, and are typically *continuous* variables. Steele and Iliinsky (2010) present a few visualization techniques that work well with categorical data: treemaps (Shneiderman and Wattenberg (2001)), mosaic plots (Theus (2003)) and parallel sets (Bendix *et al.* (2005)). These do not scale to many variables and do not allow the simultaneous visualization of continuous variables. There are also several techniques for visualizing multivariate data (i.e., multiple measures), such as parallel coordinates (Inselberg (1985)) and scatterplot matrices (Hartigan (1975)), but they do not handle dimensions very well.

Tableau (Mackinlay *et al.* (2007)), the commercial descendent of Polaris (Stolte *et al.* (2002a)), is a system that allows casual business users to create various types of charts and plots by simple drag and drop operations. It does so by displaying a list of all variables, segregated by type, and picking an appropriate display depending on the type of the variables chosen by the user.

However, as Tableau's design depends on the user explicitly constructing a visualization by choosing variables to plot, it does not initially display any visual overview of the contents of the dataset.

Recent work by Emerson *et al.* (2013) introduced the *generalized pairs plot* which shows all possible pairs of variables, like scatterplot matrices, but uses different types of plots depending on the type of variables paired together. Their technique, which uses complex plot types, provides a rich overview of a dataset, at the expense of understandability by a casual business user.

The GPLOM technique proposed in our work is designed to display multidimensional multivariate data containing categories and yet be easily understood by business users to allow them to explore data sets with minimal set up. We pose the hypothesis that GPLOM allows casual users to be faster than a commercially available tool, Tableau, for certain types of exploratory queries. We evaluate GPLOM, and test our hypothesis, with a controlled experiment involving users.

In business settings, data is often stored in analytical stores and data warehouses. Data that is to be visualized may be fetched through SQL queries, computed as the output of MapReduce jobs or calculated from OLAP datacubes. Unfortunately, SQL is designed to answer queries by computing an exact result, which can lead to long wait times for complex queries on large datasets. MapReduce, similarly, is designed for batch processing of large amounts of data rather than interactive operation. This is a problem, because responsiveness is key to interactive visual analytics. As Mackinlay *et al.* (2007) put it: "Tableau has users with very large databases who are willing to wait minutes for database queries to run so that they can see a graphical view of their valuable data. However, users do not want interactive experiences that include such pauses." SQL and MapReduce, therefore, have severe drawbacks for exploratory analysis of large datasets.

Recently, Agarwal *et al.* (2013) proposed BlinkDB, a database that uses precomputed stratified samples to allow queries with bounded response time, further demonstrating the need for databases to give rapid feedback over absolute result accuracy.

OLAP datacubes offer rapid computation of aggregates by preaggregating the data along dimensions, needing only simple aggregations of aggregates at query time instead of processing all the data. However, there are limits to how many dimensions can be preaggregated, as cube volume increases exponentially with the number of dimensions that are aggregated. Furthermore, some descriptive statistics cannot be computed through the usage of cubes, such as percentiles.

VisReduce enables the incremental rendering of a visualization, so that an analysis of a large dataset can give responsive feedback to the user while the back end still processes the data. Because VisReduce allows incremental construction of a visualization as data is being processed, it enables rapid feedback loops that would not have been otherwise possible if the user had to wait for the completion of a database query in order to see results. While the advantages of incremental visualization are obvious, we also wanted to ensure that the performance of VisReduce was comparable to existing systems; existing user studies with incremental visualization, such as the one done by Fisher *et al.* (2012), only focus on the end user aspects, not implementation. The performance of VisReduce is tested by comparing its query performance with other readily available systems which are not incremental in nature.

GPLOM and VisReduce have each been presented in international conferences (Im *et al.* (2013b,a)). In the following thesis, each of the two techniques is discussed separately in its own chapter, with a background section, implementation details as well as details of the evaluation of each technique.

CHAPTER 1

BACKGROUND

As mentioned in the introduction, business intelligence is a collection of tools and processes that support business decisions by allowing the various business stakeholders to base their decision process on facts. Information visualization is an important part of these processes, as many stakeholders need not only to understand patterns in data but also communicate them. As the old adage goes, "a picture is worth a thousand words."

1.1 The need for visualization

Heer and Shneiderman (2012) explain that information analysis requires human judgement to interpret patterns, groups, trends and outliers so as to understand their domain-specific significance. For example, a business analyst for a coffee chain might notice that sales of coffee are higher in December than in July; while both a computer and a human can discern such a pattern, only a human can infer that such a trend is due to the weather rather than, say, the number of vowels in the name of the month, even though both are correlated with sales. Leinweber (2007) writes an entertaining demonstration on the importance of human common sense during correlation analysis by showing that the performance of the S&P 500 is strongly correlated with butter production in Bangladesh, even though such a correlation is clearly fortuitous and nonsensical.

Furthermore, descriptive statistics are sometimes insufficient to understand the details present in data. Anscombe (1973) presents the four data sets shown in Table 1.1 that appear identical under cursory analysis — each having equal averages $\bar{x}$ and $\bar{y}$, standard deviations σ_x and σ_y, as well as the same linear regression equation and correlation coefficient R^2 — yet are very different when displayed graphically, as in Figure 1.1. Anscombe (1973) makes the case that one should not only rely on statistical analysis but should also visualize the data in order to understand it.

Table 1.1 Data for Anscombe's Quartet: four different data sets of (x, y) points that nevertheless share the same summary statistics.

	Dataset 1		Dataset 2		Dataset 3		Dataset 4	
Values	x	y	x	y	x	y	x	y
	10	8.04	10	9.14	10	7.46	8	6.58
	8	6.95	8	8.14	8	6.77	8	5.76
	13	7.58	13	8.74	13	12.74	8	7.71
	9	8.81	9	8.77	9	7.11	8	8.84
	11	8.33	11	9.26	11	7.81	8	8.47
	14	9.96	14	8.1	14	8.84	8	7.04
	6	7.24	6	6.13	6	6.08	8	5.25
	4	4.26	4	3.1	4	5.39	19	12.5
	12	10.84	12	9.13	12	8.15	8	5.56
	7	4.82	7	7.26	7	6.42	8	7.91
	5	5.68	5	4.74	5	5.73	8	6.89
n	11		11		11		11	
$\bar{x}$	9.00		9.00		9.00		9.00	
$\bar{y}$	7.50		7.50		7.50		7.50	
σ_x	3.32		3.32		3.32		3.32	
σ_y	2.03		2.03		2.03		2.03	
$\sum(x - \bar{x})^2$	110.0		110.0		110.0		110.0	
Equation	$y = 3.00 + 0.50x$		$y = 3.00 + 0.50x$		$y = 3.00 + 0.50x$		$y = 3.00 + 0.50x$	
R^2	0.67		0.67		0.67		0.67	

Ware (2004) explains that data visualization provides an ability to comprehend huge amounts of data by using the perceptual power of the brain's visual processing system. Tory and Möller (2004b) add that by using the advantages of visual perception, we can compensate for cognitive barriers, such as a limited working memory. Table 2.2 of Thomas and Cook (2005) lists additional advantages of visualization.

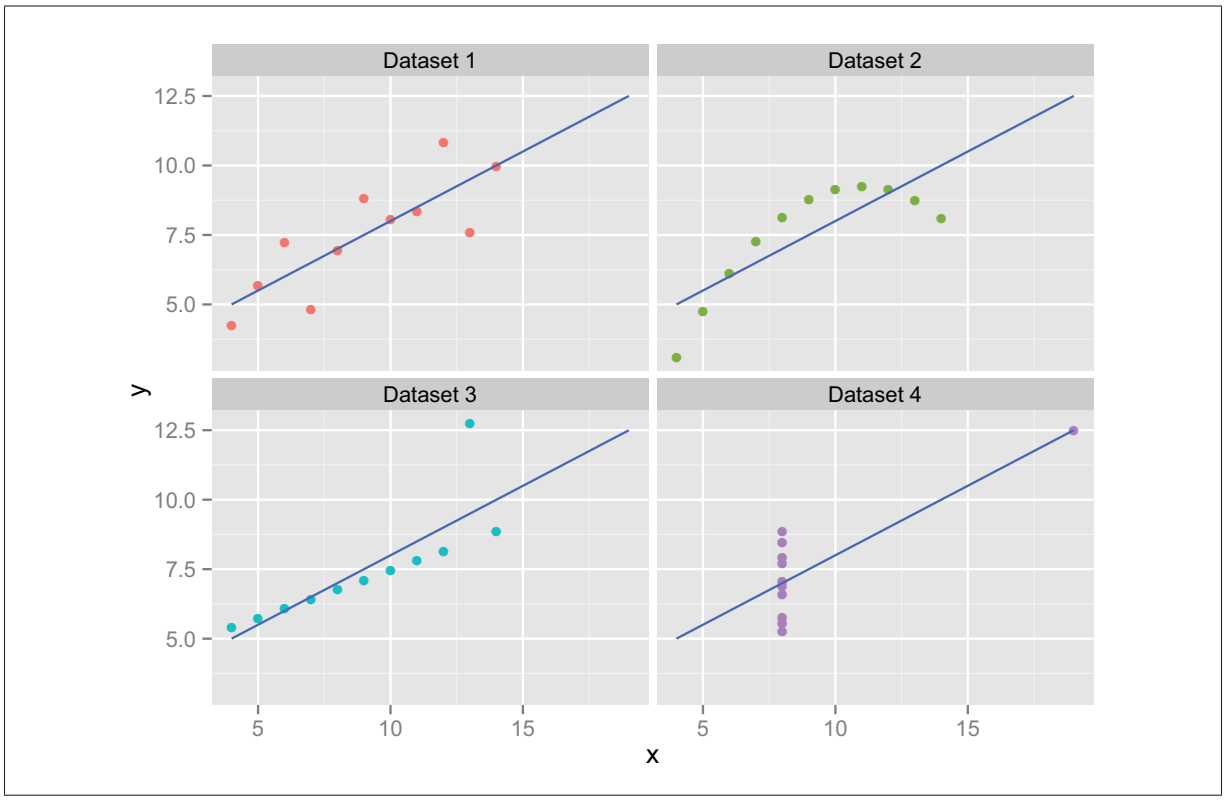

Figure 1.1 Plot of Anscombe's Quartet: visualizing the data shows large differences between datasets. The linear regression is shown in blue.

1.2 Business intelligence technologies

Chaudhuri *et al.* (2011) give an overview of the various technologies used for business intelligence. Typically, business intelligence systems are architected by combining several components. Operational (also called transactional) databases contain the data used in daily operations. For example, a hypothetical factory that makes nuts and bolts could have an operational database that contains addresses of customers and their orders, while another might contain customer service requests. These databases are collated together during a process called extract, transform and load (ETL), during which the transactional records are extracted from various data sources, transformed into business events suitable for analysis and loaded into a data warehousing system. These data warehouses are often also relational databases, although this need not be. For example, some data warehouse systems use parallel processing systems such as Apache Hadoop when dealing with large data sets.

There are different approaches to querying such data warehouses for analytical purposes. A popular approach, online analytical processing (OLAP) — also frequently called multidimensional online analytical processing (MOLAP) — uses structures called data cubes. Data cubes contain precomputed aggregates across sets of predefined dimensions, speeding up certain types of queries. For example, a data cube about sales information for a company with three dimensions (region, quarter and product category) would be as shown in Figure 1.2.

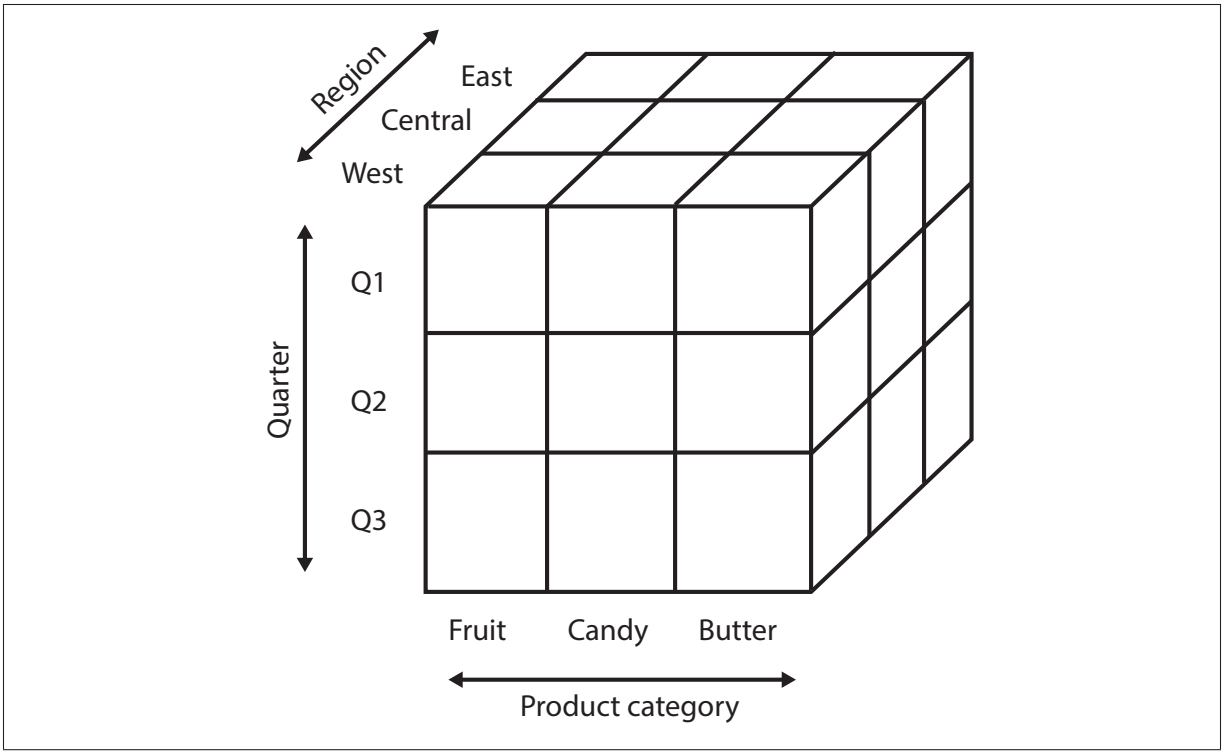

Figure 1.2 Data cube for a hypothetical supermarket. Each cell of the cube represents an individual aggregate for a particular combination of dimensions.

Each cell of the data cube contains aggregate information about the data matching that particular cell, but not the data itself. This means that certain aggregate operations (COUNT, SUM, MIN, MAX, AVG) can be done in constant time, no matter how much data the cube represents. On large data sets, the cubes might be generated offline during the night, allowing analysts to analyze them the next day.

Wilkinson *et al.* (2005) mentions an important caveat with the usage of data cubes: because data cubes do not contain the data but only aggregated information about it, several key statis-

tics cannot be derived from a data cube, such as the median, mode, percentiles or quartiles. They also do not allow ad hoc queries on dimensions that are not part of the cube. For example, it is impossible to query the data cube of Figure 1.2 to only show sales by a particular salesman or that have taken place during a particular month, as neither are dimensions of the cube. It would obviously be possible to add more dimensions to the cube to solve this problem, but this also increases the volume of the cube due to dimensional explosion, making it impractical to use cubes with many dimensions. Wilkinson *et al.* (2005) summarizes the intersection of data cubes and data visualization in a single sentence: "Few of the graphics in this book and in other important applications can be computed from a data cube."

On the other hand, the relational online analytical processing (ROLAP) approach stores the data in a relational database and queries it directly; this allows the data to be queried in different fashions as the data is stored verbatim but performance is a function of the amount of data stored in the system.

Because all of these systems are used for business reporting, they compute exact values for any given query. However, for visual analytics, a fast approximation rather than an exact result improves the end user experience, as explored by Fisher *et al.* (2012). The prototype built by Fisher *et al.* (2012) builds incremental approximations of a database query, letting the user control the threshold between accuracy and timeliness by waiting for the query to have smaller error bounds. As their prototype was built to explore the user experience of such an incremental visual analytics system rather than designing a high performance incremental query processing engine, it does not address the issue of designing such a system, which is still a current research problem. Later, we will show that VisReduce contributes a new approach for such systems.

1.3 Multidimensional multivariate visualization

Many datasets can be represented in tabular form, with one column for each variable, and one row for each tuple. There are two types of columns: *independent variables*, also called *dimensions*, which are usually *categorical* variables (or discretized variables that are almost equivalent to categorical variables), and *dependent variables*, also called *measures*, which are

normally *continuous* variables. Such datasets, with a mix of dimensions and measures, are sometimes called *multidimensional multivariate*, or MDMV, data. Surveys of techniques for visualizing MDMV data can be found in (Wong and Bergeron (1997); Grinstein *et al.* (2001); Keim (2002)). We will consider the most relevant of these techniques, and consider a fictitious "nuts-and-bolts" dataset to illustrate some differences between previous work. The nuts-and-bolts data is stored as a table, and involves three (independent) categorical variables: Region (North, Central, or South), Month (January, February, ...), and Product (Nuts or Bolts). It also involves three (dependent) continuous variables: Sales, Equipment costs, and Labor costs. The values of the categorical variables yield $3 \times 12 \times 2 = 72$ combinations of categorical values, each corresponding to a row in a table, and each mapping to values of the continuous variables:

Table 1.2 The tabular representation of the nuts-and-bolts dataset.

Region	Month	Product	Sales	Equipment costs	Labor costs
North	Jan	Nuts	2.76	0.92	4.30
North	Jan	Bolts	4.92	1.64	4.30
North	Feb	Nuts	4.20	1.00	4.30
North	Feb	Bolts	8.40	2.00	4.30
North	Mar	Nuts	5.28	9.60	4.30
⋮	⋮	⋮	⋮	⋮	⋮
South	Dec	Bolts	9.50	2.44	5.20

TableLens (Rao and Card (1994)) and FOCUS (Spenke *et al.* (1996)) (later renamed InfoZoom) provide ways to aggregate the tuples in a list such as the one above, while still presenting an essentially tabular view to the user. Both systems allow the user to sort tuples by any variable, but have limited ability to ease the understanding of multiple categorical variables.

Scatterplot matrices (SPLOMs) were proposed by Hartigan (1975), and display a scatterplot for every possible pair of variables. Notable more recent work includes Scagnostics (Wilkinson

et al. (2005)), which enable SPLOMs to scale up to many continuous variables, and Scatterdice (Elmqvist *et al.* (2008)), which demonstrates how they can be made highly interactive. SPLOMs nevertheless have shortcomings when used to visualize categorical variables. In Figure 1.3, the top three scatterplots (e.g., Month vs Region) each show a crossing of two categorical variables, resulting in an uninformative grid of points. Scatterplots showing a continuous vs categorical variable suffer from overplotting: in the Sales vs Product scatterplot, it is not obvious which of the products resulted in higher overall sales.

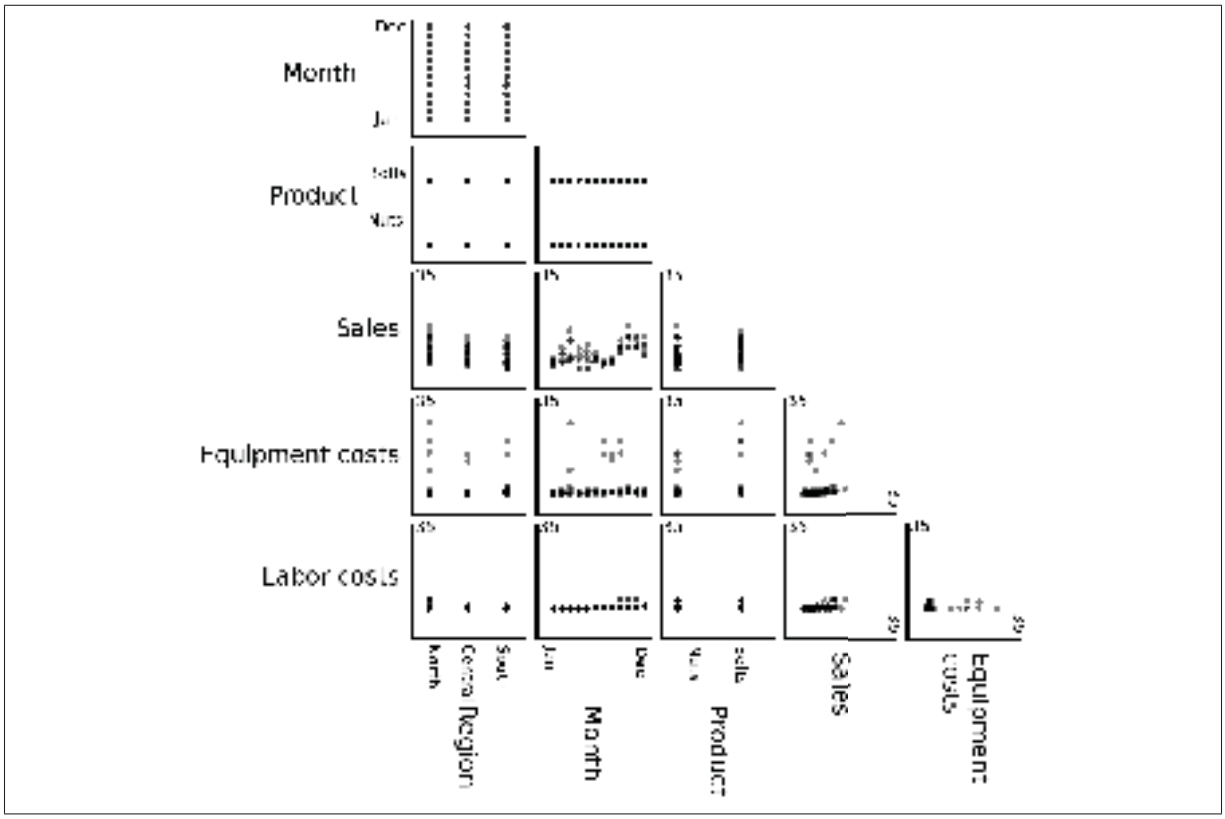

Figure 1.3 A SPLOM of the nuts-and-bolts dataset.

HyperSlice (van Wijk and van Liere (1993)) displays a matrix of slices of a scalar function of many dimensions, but cannot display several (dependent) continuous variables at once. The heatmaps of GPLOM, explained in the next chapter, are similar to HyperSlice, though GPLOM's heatmaps display aggregations of data rather than slices.

Parallel coordinates (Inselberg (1985); Wegman (1990)) show each tuple as a polygonal line intersecting an axis once for each of the variables. Figure 1.4 shows an example. The 3 right-most axes show continuous variables, allowing us to see the distribution of values along them (the range and central tendency of values, and outliers). However, the three left-most axes show categorical variables, where every possible combination of values is covered, resembling complete bipartite graphs. This creates ambiguities that prevent us from visually tracing a tuple across all axes (although interactive highlighting could alleviate this).

A technique called parallel sets (Bendix *et al.* (2005)) displays multiple categorical variables side by side, as in a parallel coordinate plot. While it allows tracing tuples across multiple axes, parallel sets also have very cluttered displays when used with high cardinality variables.

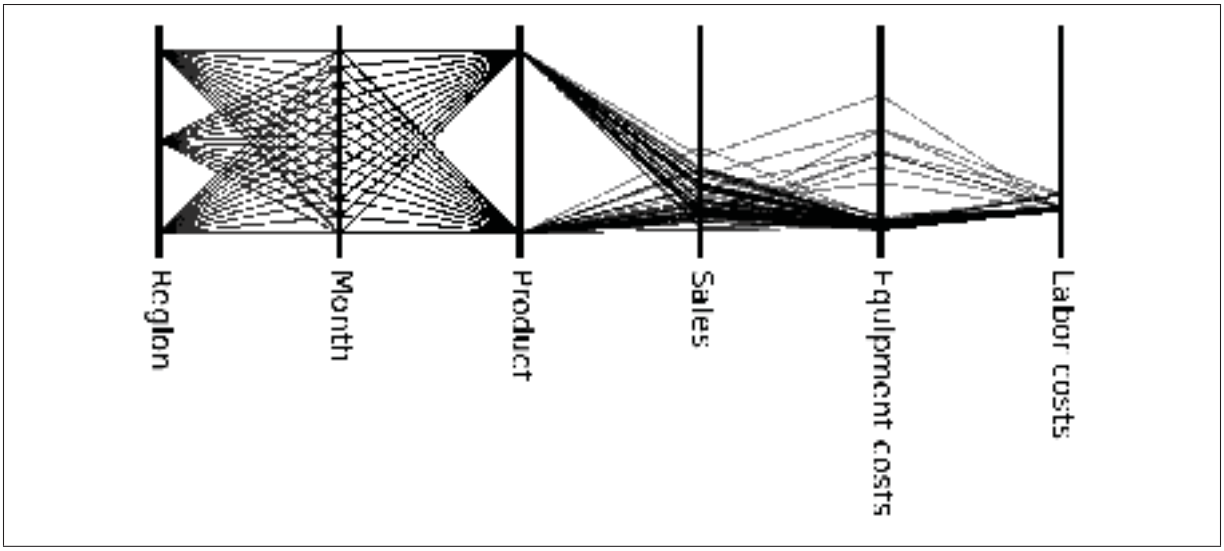

Figure 1.4 A parallel coordinates plot of the nuts-and-bolts dataset.

Various combinations of scatterplots and parallel coordinates have been proposed, displaying them side-by-side (Qu *et al.* (2007); Steed *et al.* (2009)) or more tightly integrated (Yuan *et al.* (2009); Holten and van Wijk (2010); Viau *et al.* (2010); Claessen and van Wijk (2011)), but none of these approaches facilitate the visualization of categorical variables.

Arrays of glyphs can be used to visualize MDMV data, where each glyph shows one tuple (Bertin (1967); Chernoff (1973); Kleiner and Hartigan (1981); Pickett and Grinstein (1988);

Ward (2002)). This works well when there are at most 2 (independent) categorical variables. For example, an arrow plot (Wittenbrink *et al.* (1996)) can display an arrow-shaped glyph at each of the points on a 2D grid, showing wind speed and wind direction over a geographic map. Extending this to 3 spatial dimensions results in occlusion, and beyond 3 dimensions it becomes very difficult to understand the ordering of glyphs along each dimension.

Dimensional stacking (LeBlanc *et al.* (1990); Mihalisin *et al.* (1991)) allows more than one categorical variable to be mapped to the same spatial axis, and has been used in database visualization (Stolte *et al.* (2002a); Mackinlay *et al.* (2007)). Figures 1.5 and 1.6 show examples, each of which shows a total of 4 variables. The two innermost variables of the stacking determine the type of chart shown: if the innermost vertical variable is a continuous variable (e.g., Sales), and the innermost horizontal variable is a categorical variable (e.g., Month), then barcharts are used. On the other hand, scatterplots are used if the two innermost variables are continuous variables (e.g., Equipment costs vs Sales).

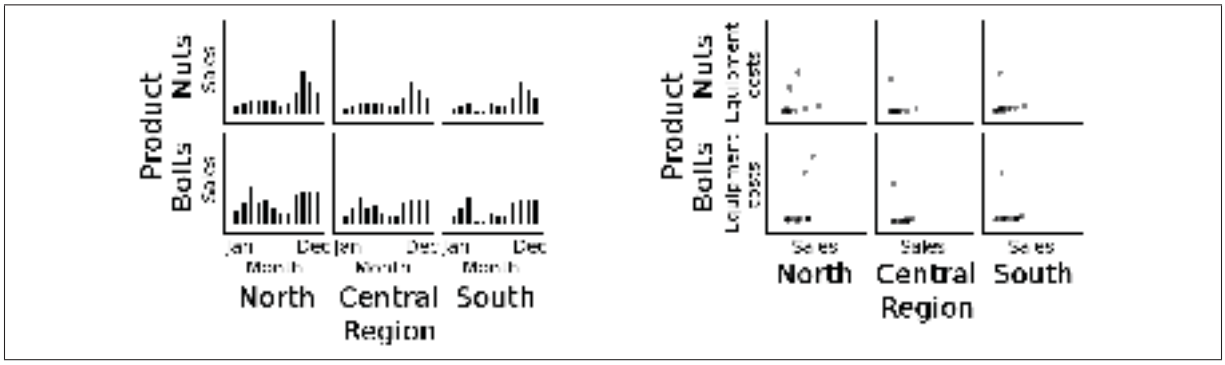

Figure 1.5 Examples of dimensional stacking with the nuts-and-bolts data. Left: Product and Sales are mapped to the vertical axis, Region and Month are mapped to the horizontal. Right: Product and Equipment costs mapped to the vertical, Region and Sales to the horizontal.

Each of the charts in Figures 1.5 and 1.6 show a *slice* of the data, allowing the user to see more detail. For example, Figure 1.6 reveals that sales were very low in the South in April and May. By comparison, in Figure 1.3, the Sales vs Month scatterplot also shows low sales in April and May, without revealing the Region.

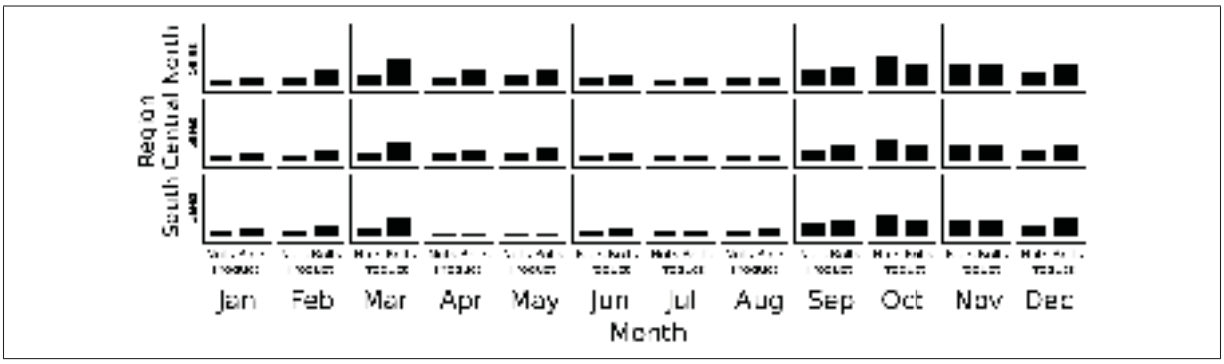

Figure 1.6 Another example of dimensional stacking applied to the nuts-and-bolts data. Region and Sales are mapped to the vertical axis, Month and Product to the horizontal.

The added detail visible in Figures 1.5 and 1.6, however, comes at the cost of exponential growth in space requirements as categorical variables are added. For example, if the dataset had an additional categorical variable Year with values 2001, 2002, ..., 2010, adding this as an outer variable to Figure 1.6 would increase the number of charts by a factor of 10. Partly for this reason, software like Tableau (Mackinlay *et al.* (2007)) does not show the user an initial visualization of the data. Instead, Tableau initially shows a list of variables (Figure 1.7) from which the user may drag and drop to construct a desired visualization (Figure 1.8).

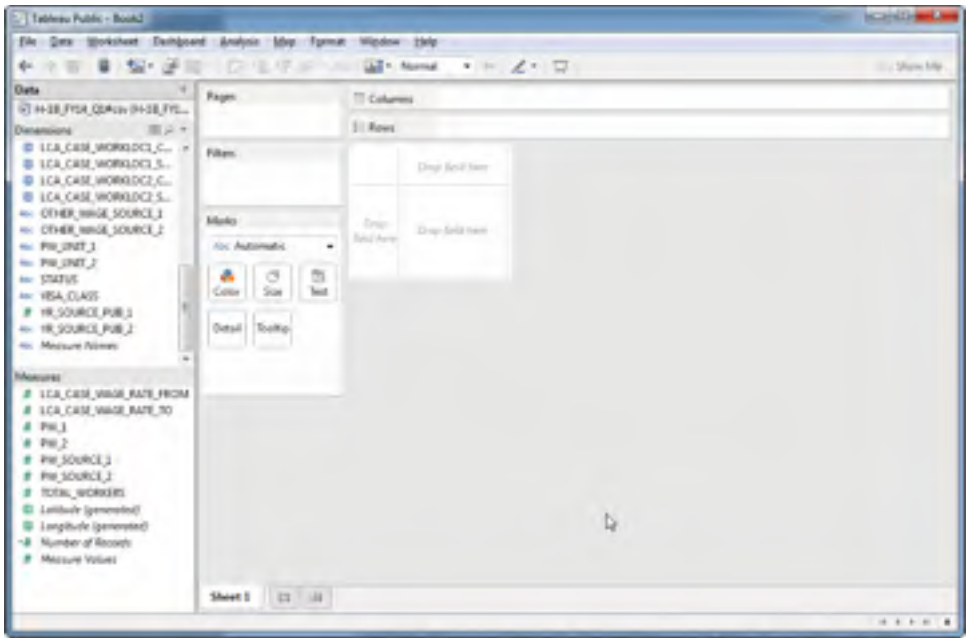

Figure 1.7 An empty workbook in Tableau. At this point, the user must construct a visualization by dragging variables onto the shelves. Tableau's interface does not expose the contents of the various variables, only their names and data types.

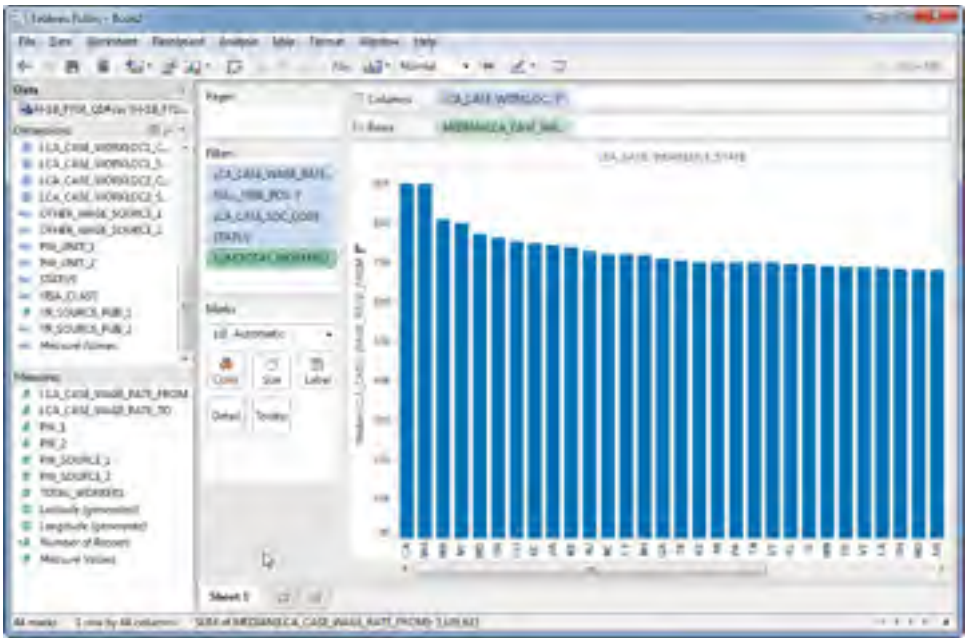

Figure 1.8 A visualization in Tableau. In this case, the user has added several filters to see a subset of the data and built a visualization that shows the median of a variable, broken down by state.

Figure 1.9 A generalized pairs plot generated with the `ggpairs` package, showing all pairs of dimensions. Plot types in this pairs plot are — from top left to bottom right — bar charts of variable cardinality, box plots of variable distribution, barcode plots of variable distribution, scatterplots and correlation coefficients.

Emerson *et al.* (2013) propose the *generalized pairs plot* (Figure 1.9), which extends the SPLOM by using different types of charts depending on the types of variables paired together, alleviating the problems that occur when categorical variables are shown in a SPLOM (Figure 1.3). The generalized pairs plot is promising step in the direction of better visualizations of multidimensional multivariate datasets, because the choice of chart in the matrix is based on the types of variables involved. However, Emerson *et al.* (2013)'s work still leaves room for improvement. First, their implementation only generates static visualizations. Many interactive features could be added, to allow the user to interactively highlight and explore the data in the visualization. Second, their implementation uses several kinds of charts, some of which (such as mosaic plots and boxplots) do not scale well for high-cardinality variables and may

be unfamiliar to casual business users. We therefore propose the GPLOM (Generalized PLOt Matrix) that makes Emerson *et al.* (2013)'s generalized pairs matrix both highly interactive, and simpler to understand.

CHAPTER 2

THE GENERALIZED PLOT MATRIX

2.1 Introduction

Many datasets are stored in tabular form, with one row for each tuple, and one column for each attribute. If the attributes are *dependent variables* (e.g., dependent variables of a key or row id), we speak of *multivariate* data, for which many techniques exist for visualizing several variables at once, such as scatterplot matrices (SPLOMs) (Hartigan (1975)), parallel coordinates (Inselberg (1985)), and glyphs (Bertin (1967); Chernoff (1973); Kleiner and Hartigan (1981); Pickett and Grinstein (1988)). Some of the columns, however, may be best thought of as *independent variables*, in which case we speak of multidimensional multivariate (MDMV) data (Wong and Bergeron (1997)). Stolte *et al.* (2002a) use the term *dimension* for a (categorical or ordinal) independent variable, and *measure* for a dependent variable. We will refer to dimensions as categorical variables, and measures as continuous variables.

The aforementioned techniques, of SPLOMs, parallel coordinates, and glyphs, all suffer from problems when naively applied to datasets with many categorical variables. An alternative approach involves "stacking" multiple categorical variables along axes, used in trellis charts and other techniques (LeBlanc *et al.* (1990); Mihalisin *et al.* (1991); Stolte *et al.* (2002a)) and more recently in the commercially successful product Tableau (Mackinlay *et al.* (2007)). However, dimensional stacking suffers from a combinatorial explosion if too many categorical variables are displayed at once.

Recent work (Emerson *et al.* (2013)) offers a new solution for visualizing MDMV data, based on the observation that SPLOMs need not display scatterplots for all pairs of variables. A plot matrix could instead display different charts for different pairs of variables, which Emerson *et al.* (2013) demonstrated with a wide variety of charts. We adapted this idea with our own technique called the Generalized Plot Matrix (GPLOM). In our approach, the visualization is simpler than Emerson *et al.* (2013)'s, as we use only three kinds of charts, chosen with

rules similar to those of Mackinlay *et al.* (2007): scatterplots for pairs of continuous variables, barcharts to show a continuous variable as a function of a categorical variable, and heatmaps to show a selected continuous variable as a function of a pair of categorical variables. These three charts are the minimum number necessary to cover the three possible pairings of variable types. This makes the matrix easier to understand, which could be beneficial to casual business users and other non-expert users. At the same time, we extend part of Emerson *et al.* (2013)'s work by presenting interactive features for highlighting, selecting, searching, and filtering the data.

Both Emerson *et al.* (2013)'s technique, and our own GPLOM, can comfortably display several categorical and continuous variables at once, avoiding the combinatorial explosion of dimensional stacking because the data can be aggregated within each chart. This makes these approaches appropriate for data with multiple categorical variables, as is common in business intelligence and other domains. These approaches can also provide the initial overview of a database shown to a user, serving as a visual launching point for further investigation. This is in contrast to the approach in Polaris (Stolte *et al.* (2002a)) or Tableau (Mackinlay *et al.* (2007)), where the user must first select one or several variables of interest to explicitly construct a visualization. Finally, for non-expert users, the GPLOM approach has the advantage of only using three kinds of charts, avoiding the more complicated charts such as mosaic plots or box plots that may be difficult for non-expert users to understand and that don't scale as well to high cardinality variables.

Our contributions in this chapter are (1) the GPLOM technique for visualizing multidimensional multivariate data using only three kinds of charts, making it as easy to understand as possible while still showing charts that are adapted to the kinds of variables involved; (2) a description of the visual design choices and features of our prototype implementation, including bendy highlights, associative highlighting, and a text search feature that highlights data, allowing users to quickly find charts of interest; and (3) an experimental comparison of GPLOM and Tableau that found GPLOM to be significantly faster in certain cases.

2.2 Previous Work

The most closely related work to ours is the Generalized Pairs Plot (Emerson *et al.* (2013)), which extends the matrix in a SPLOM to allow a mix of chart types to be displayed, including mosaic plots, box plots, histograms, and density contours. As demonstrated in the next section, this is scalable to a larger number of continuous *and* categorical variables than previous techniques, because the space requirements scale linearly with the number of variables, rather than exponentially as with the previous example of dimensional stacking. Our GPLOM work further explores Emerson et al.'s ideas by (1) only using three kinds of charts, to make the visualization easier to understand by non-expert users who may simply want a visual overview of a business database as a first step in asking analytic questions; and by (2) extending the static plots of Emerson *et al.* (2013) through interactive techniques. We also (3) empirically compared GPLOM to a commercial product and found significant advantages with GPLOM in certain cases.

2.3 Description

Figure 2.1 shows an example GPLOM of 6 variables. In the Sales vs Product chart, we clearly see that Bolts outsold Nuts, thanks to the use of aggregation (via a sum operator) that generated the bar heights. The overplotting seen in Figure 1.3's Sales vs Product scatterplot is thus avoided.

Figure 2.2 shows the layout of a GPLOM for M categorical variables $x_1, ..., x_M$ and N continuous variables $y_1, ..., y_N$. A full matrix would have $(M + N) \times (M + N)$ cells, however we only display the lower triangular half, without the diagonal, as is often done with SPLOMs (e.g., Wilkinson *et al.* (2005)). Thus, our GPLOM saves space compared to the full matrices of Emerson *et al.* (2013), leaving room for interactive elements such as the infobox (discussed shortly).

The red region in Figure 2.2 contains pairs of categorical variables, and GPLOM visualizes these with heatmaps. The green region contains pairings of a continuous vs categorical variable, shown as barcharts. The purple region contains pairs of continuous variables, shown as

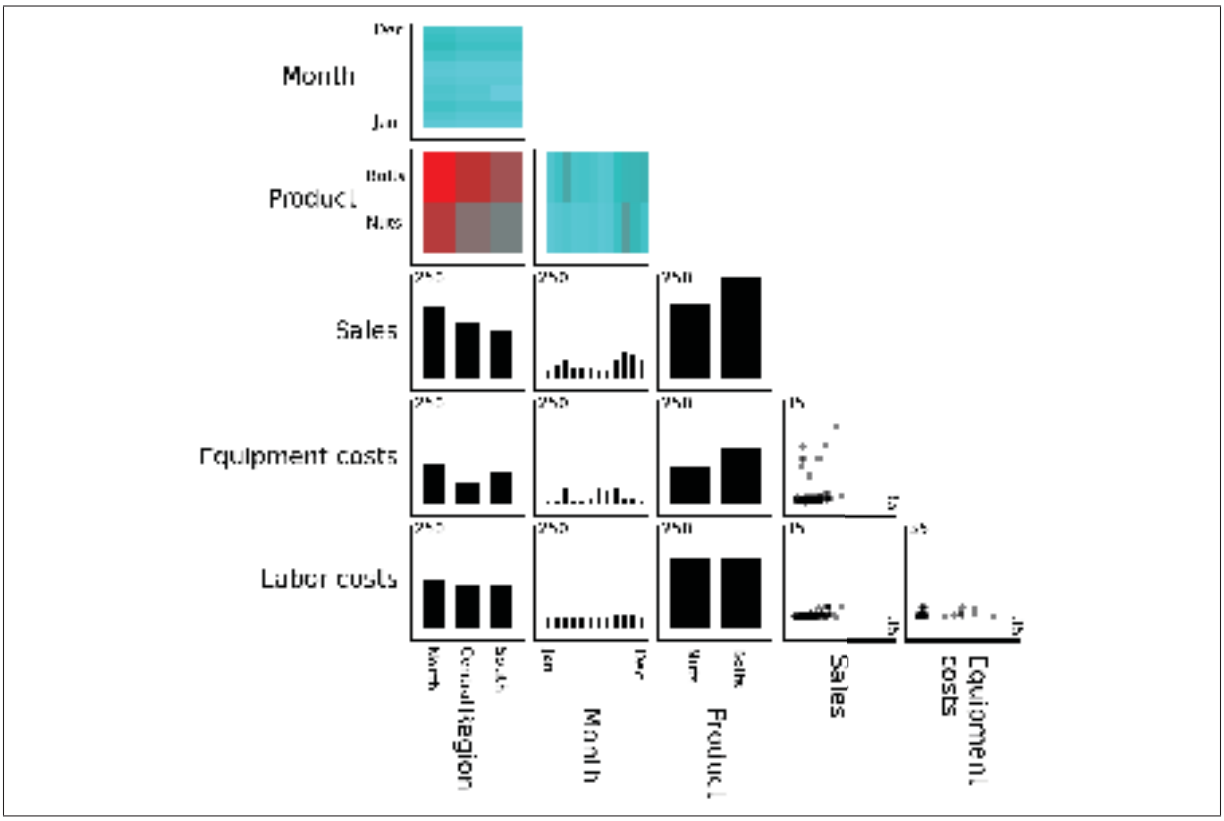

Figure 2.1 A GPLOM of the nuts-and-bolts dataset. Barcharts and heatmaps show data aggregated by sum. The vertical axes on the barcharts extend to 250, to accommodate the larger values than in the scatterplots. The heatmaps are colored to show "Sales" as a function of categorical variables, and use a color scale varying from cyan for low values, through grey for mid values, to red for the highest values.

scatterplots. (This grouping of variable types is comparable to Peng *et al.* (2004)'s ordering of variables in a SPLOM according to their cardinality.) Note that the scatterplots show individual tuples, whereas the barcharts and heatmaps show aggregated data.

Other charts in these regions are possible, as demonstrated by Emerson *et al.* (2013), such as boxplots or linecharts. However, their example plots show categorical variables with at most four distinct values. Complex charts, such as box plots and mosaic plots, become difficult to read with categorical variables with high-cardinality (Figure 2.3).

An interactive prototype of the GPLOM technique was created using D3 (Bostock *et al.* (2011)) and JavaScript. Figure 2.4 shows the prototype displaying a large real-world dataset, where the categorical variables of Year, Day of month, and Carrier have 26, 31, and 32 distinct values,

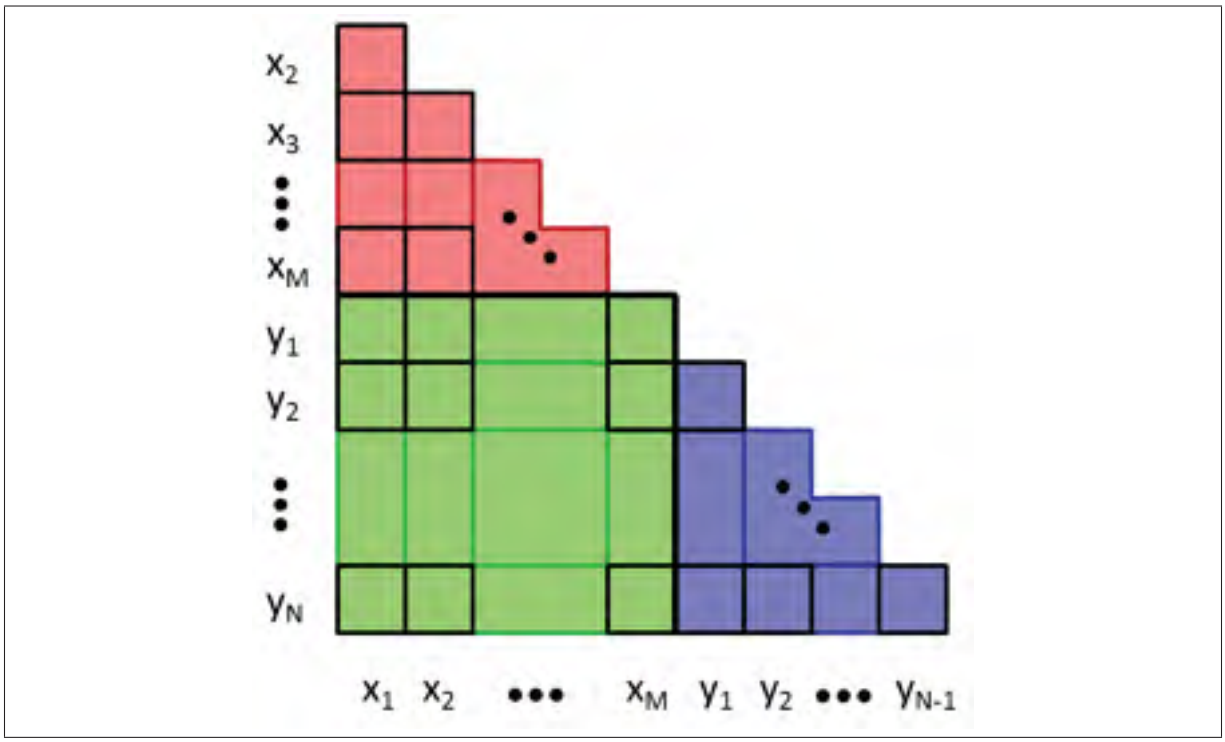

Figure 2.2 Structure of a GPLOM.

respectively. As can be seen in Figure 2.4, restricting the GPLOM to only show three kinds of simple charts — heatmaps, barcharts, and scatterplots — helps keep the charts readable at these higher cardinality values.

One tradeoff in designing a GPLOM is deciding if axes of the same variable should be scaled to the same range (facilitating comparisons of adjacent charts) or scaled to the maximum of the data in the chart. In Figure 1.3, all axes are scaled to 35. However, in Figure 2.1, the barcharts contain (aggregated) sums, and are therefore scaled to a larger range. The scatterplots in Figure 2.1, however, are still scaled to 35, to avoid having all the points clustered in a corner of the scatterplots. Furthermore, the heatmaps in Figure 2.1 share the same color scale, and we notice that only one of the heatmaps has a value close to the maximal red, because the other heatmaps are subdivided into months, reducing the values in them. Figure 2.4 instead scales each chart independently, according to the maximal value within it. This makes better use of spatial (and color) resolution, but makes it more difficult to compare charts.

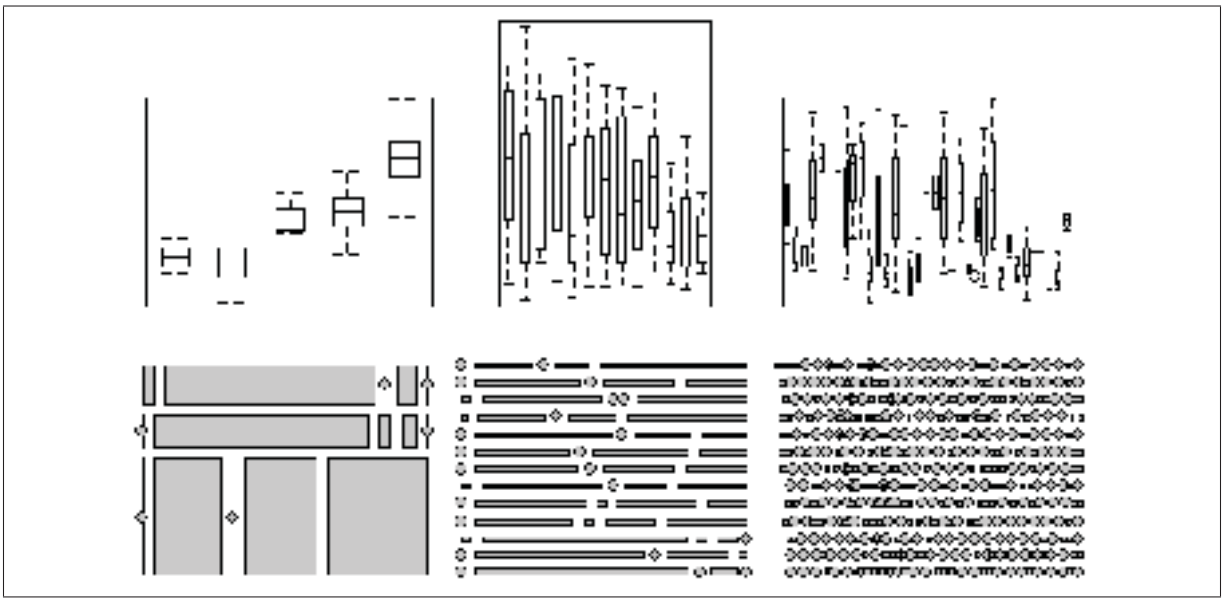

Figure 2.3 Example plots extracted from a matrix generated with the `gpairs` package in R (Emerson and Green (2012)). Top row: boxplots over variables of cardinality 5, 13, and 35, respectively. Bottom row: mosaic plots with cardinality 3×5, 5×13, and 13×35, respectively.

2.3.1 Interaction

The user may interact with the GPLOM in several ways. A GPLOM contains bars and rectangles that afford easier pointing and clicking than the small points or dots in a normal SPLOM. In our GPLOM prototype, rolling the mouse cursor over a barchart bar or heatmap cell causes it to highlight. Clicking on a bar or cell selects it.

2.3.1.1 Linking

Linking (or coordination (Roberts (2007); Wang Baldonado *et al.* (2000); North and Shneiderman (2000))) between charts is shown in two ways: *bendy highlights*, and *associative highlighting*.

Bendy highlights are specialized links that connect different charts, comparable to previous work that also draw links between views (Collins and Carpendale (2007); Steinberger *et al.* (2011); Claessen and van Wijk (2011); Viau and McGuffin (2012)). Bendy highlights are curved links that show the value of a categorical variable during rollover or selection. A text

Figure 2.4 A GPLOM of 8 categorical variables, 8 continuous variables, and 144 million flights from the OnTime dataset. The menubar at the top displays the possible aggregation operators for barcharts and heatmaps: Average (currently selected), Sum, Count, Min, and Max. The menubar also shows that "Departure delay" is the currently selected (dependent) continuous variable for heatmaps. Other interface elements: A: bendy highlight; B: textual search box; C: infobox. Note that heatmaps and barcharts are computed over the whole dataset, but scatterplots only show a random sample of 200 data points each.

string is displayed at the curved corner of the link to show the category (for example, the 5-6pm departure time block is displayed as "1700-1759" on the corner of the bendy highlight in Figure 2.4, A). Bendy highlights can also help understand the relationship between a heatmap cell and other charts (Figure 2.5).

Associative highlighting shows the relationship between charts when a categorical value is selected. There are three types of such highlighting. If the aggregation used in barcharts and heatmaps is the Sum or Count operator, then associative highlighting is achieved by highlighting the fraction of bars in other barcharts that is associated with the selected value (Figure 2.6). This is similar to the proportional highlighting of bars of Zhang and Marchionini (2004). If, instead, the aggregation used is Average, Min, or Max, then associative highlighting is achieved

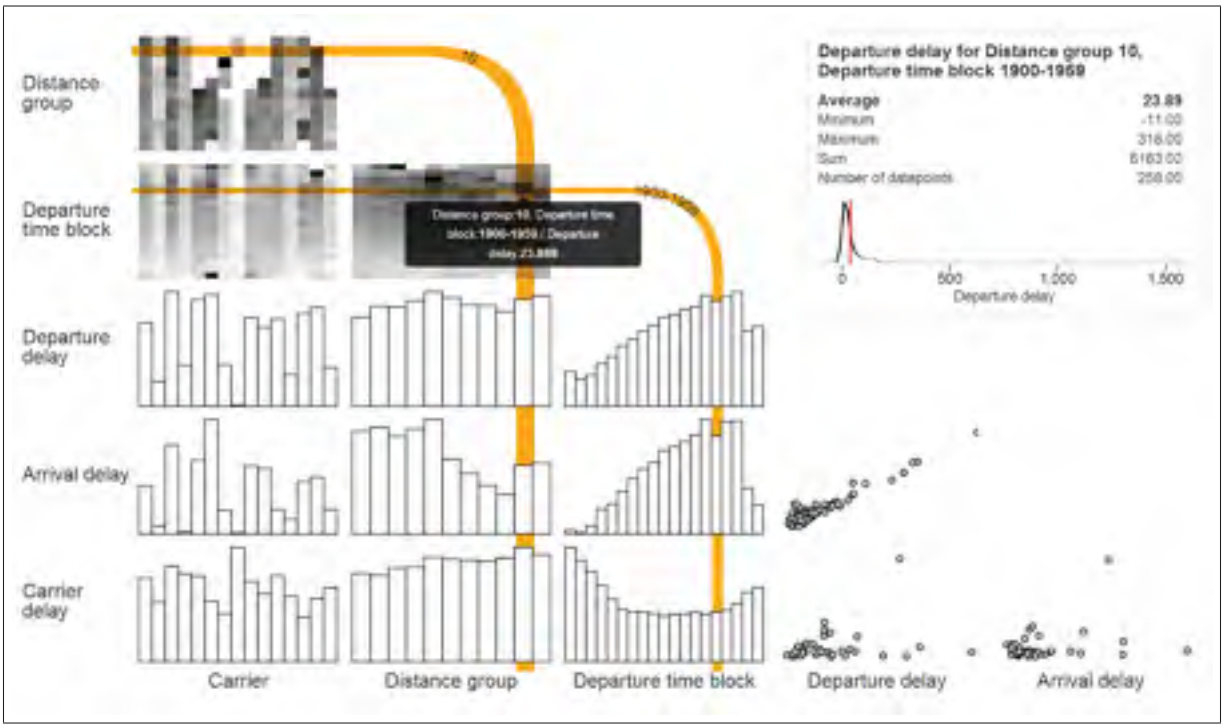

Figure 2.5 Bendy highlights and a tooltip.

by displaying dots to show the average, min, or max value of data for the selected value (Figure 2.7). Finally, regardless of the aggregation operator, the corresponding dots in the scatterplots are highlighted.

2.3.1.2 Filtering

To drill down, the user can double click on a bar (such as a bar for "Year" = 2012), causing a filter to be created that restricts the displayed data to that value. This "sheds" the corresponding categorical variable, removing a row and column of charts from the GPLOM, and creates a filter box that the user can later click on to roll back up. Figure 2.6 shows the result of applying four successive filters: "Year" = 2012, "Quarter" = 1, "Month" = 1 and "Carrier" = EV. We call this feature "dimensional shedding".

2.3.1.3 Infobox

Additional information about the element under the cursor is displayed in the infobox (upper right corner of Figure 2.4), which contains the results of the various aggregation operators

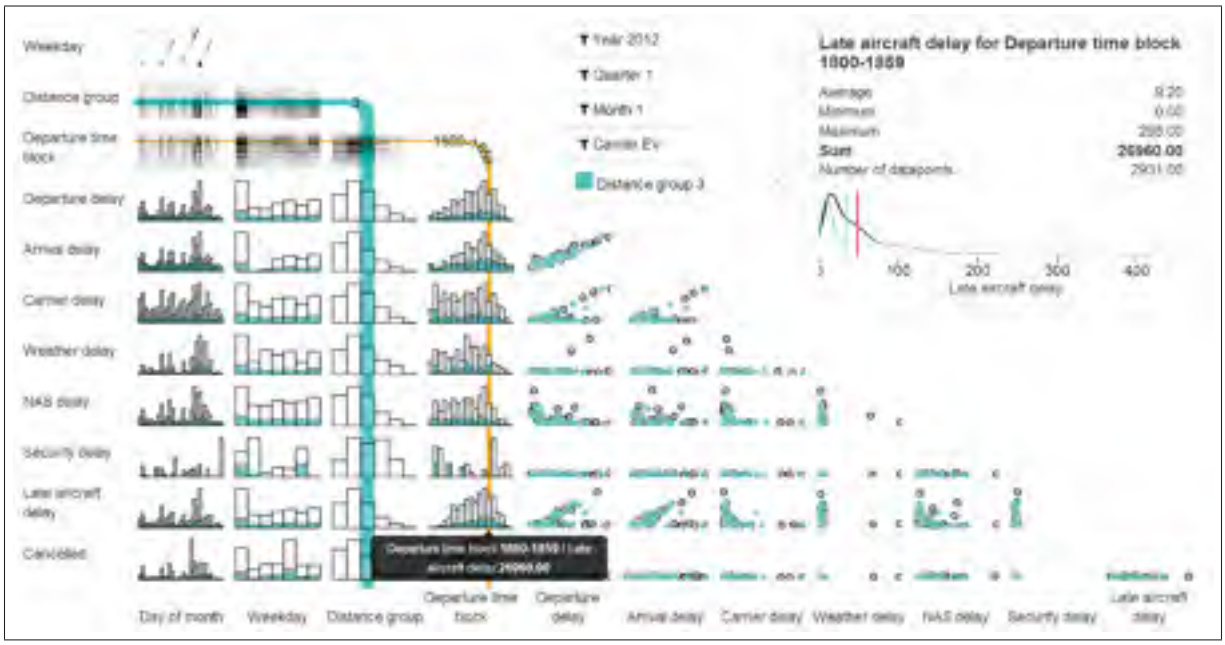

Figure 2.6 With the Sum aggregation operator, associative highlighting fills in the fraction of bars associated with the selected value “Departure time block” = “1800-1859”.

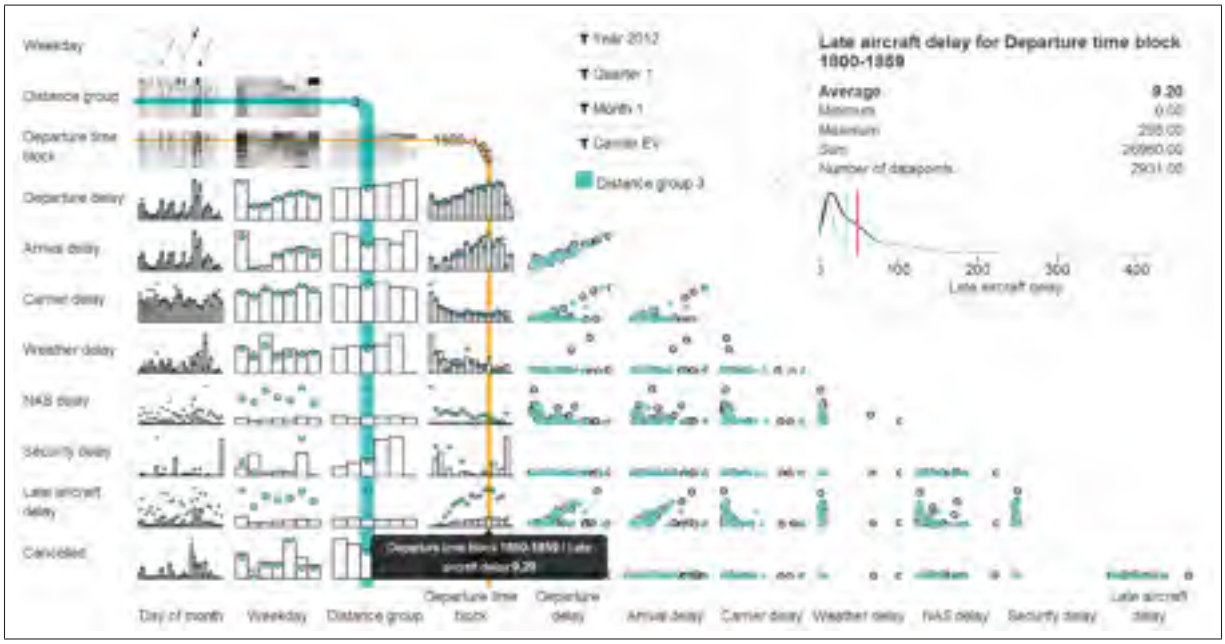

Figure 2.7 With the average aggregation operator, associative highlighting displays circles showing the average values for the selected value “Departure time block” = “1800-1859”.

as well as a kernel density estimate plot, allowing the user to judge whether the underlying distribution is normal or not, its modality and its skewness.

2.3.1.4 Text Search

Because GPLOM displays a large number of charts, it may be time consuming for users to visually scan all variable names to find a desired chart. Thus, a textual search function (Figure 2.8) allows the user to enter a string, suggests autocompletions, and highlights the corresponding elements once the string is entered. Currently, our prototype only allows the user to enter values, however it would be easy to extend the prototype to also allow entering names of variables. This feature is similar to one proposed in section 6.1 of Grammel *et al.* (2010).

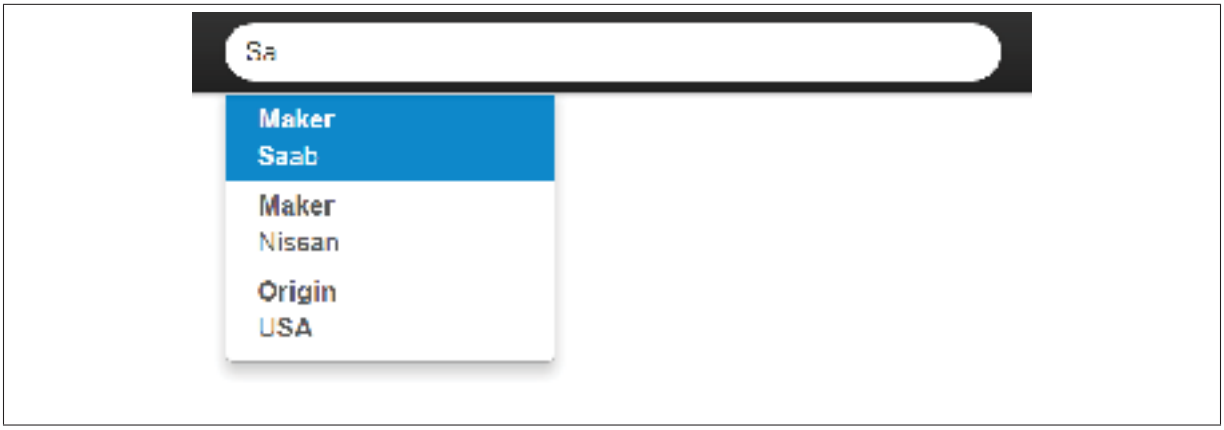

Figure 2.8 Entering the name of a value causes it to be highlighted in the GPLOM.

2.3.1.5 Labels

Due to the density of information displayed in a typical GPLOM, there is often insufficient room for labels showing the values of all categorical variables along their axes. Instead, GPLOM relies heavily on tooltips and bendy highlights that show the value under the cursor. In our first version of the prototype, we arranged categorical values on vertical axes sorted top-to-bottom, resulting in Figure 2.9. This resulted in many crossing bendy highlights. We therefore modified the prototype to sort values bottom-to-top, yielding Figure 2.10, which is the order shown in other figures in this paper. A third possibility is shown in Figure 2.11, which avoids excessive crossed links while maintaining the usual top-to-bottom ordering of values.

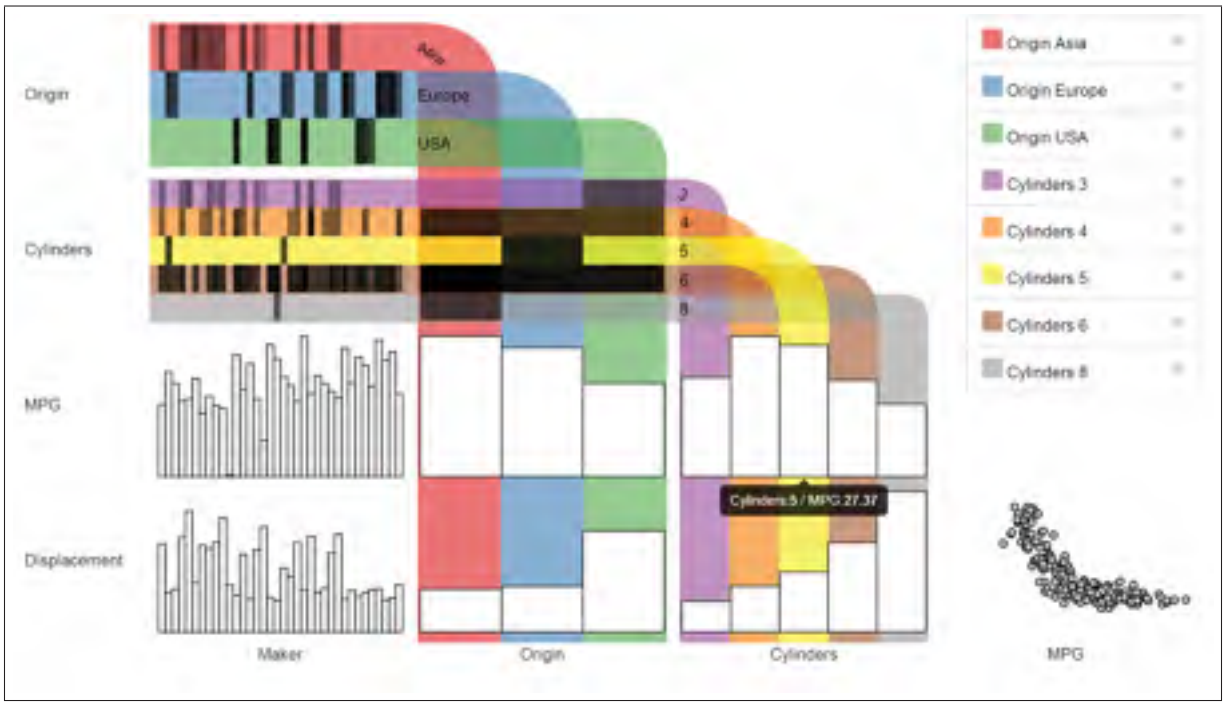

Figure 2.9 A variant of bendy highlights: alphabetical vertical sorting (e.g., Asia, Europe, USA).

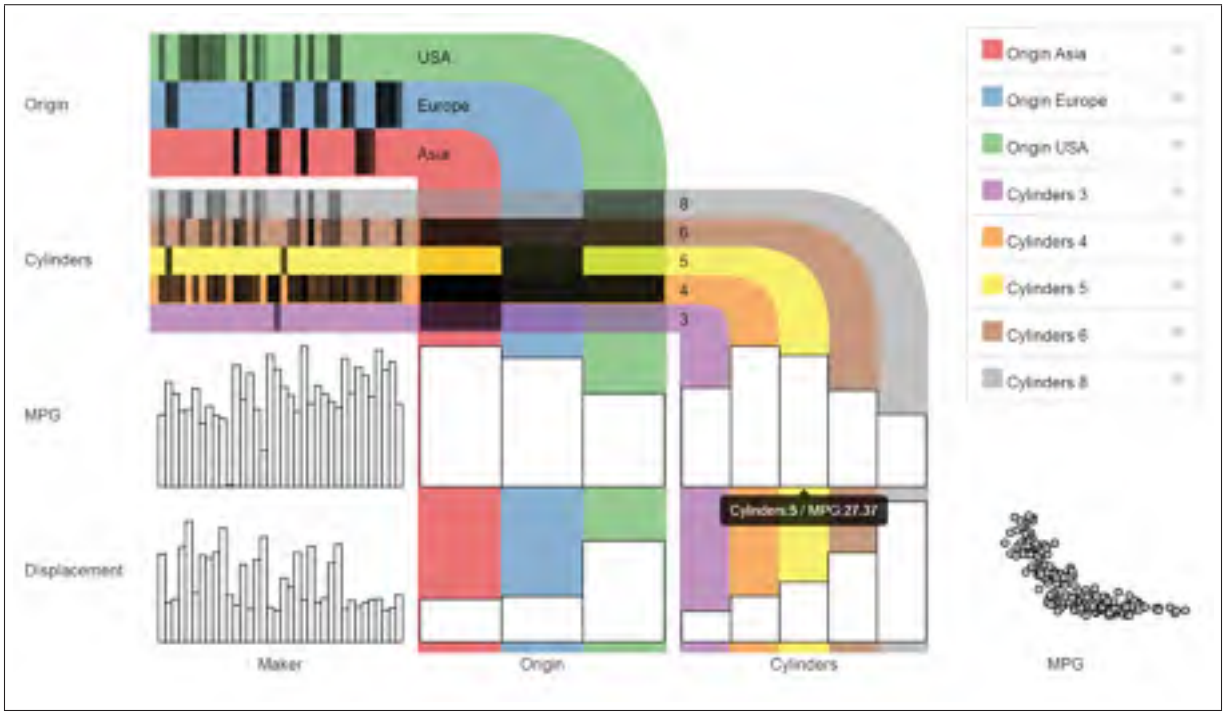

Figure 2.10 A variant of bendy highlights: reverse alphabetical vertical sorting (e.g., USA, Europe, Asia).

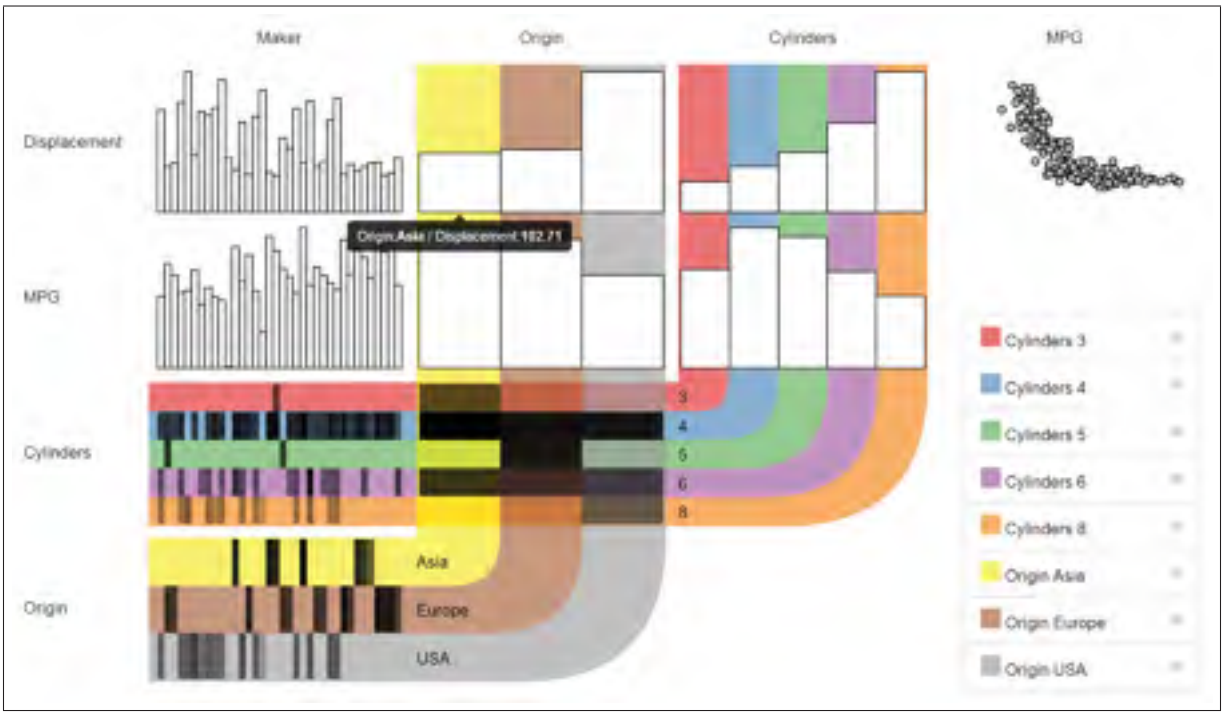

Figure 2.11 A variant of bendy highlights: a "reversed" GPLOM with alphabetical vertical sorting.

2.4 Implementation

The GPLOM architecture follows the three-tier architecture. The data tier is a standard SQL database such as SAP HANA or MySQL, the logic tier consists of a web application implemented using the Play framework[1] and the presentation tier is a JavaScript application that renders SVG on the client's browser using D3.

The logic tier computes aggregates as requested by the presentation tier and serves as an abstraction of the underlying database.

2.5 Experimental Evaluation

We suspect that one of the advantages of GPLOM is that users can answer questions by simply scanning for the appropriate chart, whereas in the commercial product Tableau they must explicitly construct a visualization. To investigate this idea, we performed an experimental comparison of user performance with both tools.

[1] http://www.playframework.com/

We chose three datasets for the experiment: a warm-up dataset (a converted sample SQL server database called Adventureworks) that was used to introduce users to both visualization tools, and two other datasets (Cars[2], and the OnTime[3] airline delay data for the month of December 2012). Cars and OnTime were each used with one of the tools, counterbalanced for dataset and ordering. One quarter of the users did (Tableau+Cars, GPLOM+OnTime), another quarter did (Tableau+OnTime, GPLOM+Cars), another quarter did (GPLOM+OnTime, Tableau+Cars) and the last quarter did (GPLOM+Cars, Tableau+OnTime).

Each trial required the user to answer a question about the dataset. There were two types of questions, and for each type of question, there could be zero or more criteria involved in the question. The *type of question* consisted of either questions that asked which type of trend or correlation (positive, negative or null) exists between two variables, if any, and questions that asked to find a particular data value, such as the year in which the average mileage per gallon for all cars was the highest. The *criteria count* ranged between zero to three criteria, so that a question "find the carrier with the highest average arrival delay" has zero criteria, while the question "find the day of the week when Hawaiian Airlines (HA) has the highest average delay for flights departing between 9:00-9:59" has two criteria (carrier=HA, departureTime=0900-0959).

In total, the experiment involved:

2 types of questions (trend or data)
$\times$ 4 criteria counts (0 through 3)
$\times$ 2 technique-dataset pairs (GPLOM and Tableau 7.0)
$\times$ 12 users
= 192 trials

The 12 students who participated (11 male, 1 female) were either from the software engineering or information technology engineering programs at ETS, at both the undergraduate and graduate levels. Each student was assigned to one of the four between-subjects groups and was

[2]http://lib.stat.cmu.edu/datasets/
[3]http://www.transtats.bts.gov/Fields.asp?Table_ID=236

asked to explore the warm-up dataset for five minutes with one of the two techniques (either Tableau or GPLOM, depending on the participant's group), as an exploration phase. Once the five minutes were over, each participant was shown how to use the software in order to answer the questions, then presented with eight questions for the warm-up dataset. After the questions on the warm-up dataset were answered, a second dataset (either Cars or OnTime, depending on the participant's group) was shown and the participant was asked to answer questions about the new dataset. Then, the participant explored the warm-up dataset again, using the other technique, answered the same eight questions using the other technique and, finally, answered a set of questions on a different dataset than the one explored with the first technique.

None of the participants indicated that they had any prior experience with Tableau or with the GPLOM prototype. During the exploration phases and warm up trials, users were free to ask questions, and were shown all the features of the user interfaces that were necessary to answer the questions in the experiment.

The participants used a single monitor workstation equipped with a 24 inch monitor, keyboard and mouse.

The GPLOM prototype consisted of a web application built using D3 and JavaScript, running in the Chrome web browser (version 25), as well as a server-side backend. The server-side backend managed communication between the client and a MySQL server, computing agregates to be consumed by D3. It was built using the Play framework 2.0.4 and ran in production mode during user tests.

Tableau and GPLOM both connected to the same MySQL database over a wired gigabit Ethernet network.

As the GPLOM prototype was not optimized for performance, each time a participant added a filter by double clicking, a full page load by Chrome was executed, requiring Chrome to re-interpret and run JavaScript code (in theory, this could be eliminated with more careful coding) and also regenerating all charts (this is unavoidable with the GPLOM approach). We

subsequently measured that the median time for all this to occur was 1.7 seconds for the Cars dataset, and 5.7 seconds for OnTime.

The questions were displayed to users on a second monitor controlled by the experimenter. When the user indicated they were ready to start a trial, the experimenter clicked a button to display the question and start a timer. The user then read the question and interacted with the visualization tool until they said they could answer the question, at which point the experimenter stopped the timer (triggering a simultaneous screen grab of the user's screen), and transcribed the user's verbal answer. The time elapsed was recorded. If the user decided they wished to check or change their answer by performing further interactions with the visualization tool, the time of their last answer determined the recorded duration. No feedback was given to indicate to the user if their final answer was correct or incorrect.

2.5.1 Results

Because each participant was only exposed to half of the four {GPLOM, Tableau} $\times$ {Cars, OnTime} combinations, the performance data were separated by dataset for analysis. Some of the main results are summarized in Figure 2.12 and below:

Table 2.1 Breakdown of error rate and median time to complete tasks by dataset and technique.

	GPLOM		Tableau	
	median time (s)	error rate	median time (s)	error rate
Cars	23.67	13%	41.14	10%
OnTime	48.68	17%	59.58	33%

The time taken by participants to answer was non-normally distributed (Shapiro-Wilk normality test, $p < 0.01$). The non-parametric ANOVA-type statistic (ATS) revealed that GPLOM was significantly faster than Tableau for the Cars dataset ($p < 0.01$), and that the criteria count had a significant effect on time in both the Cars dataset ($p < 0.01$) and the OnTime dataset

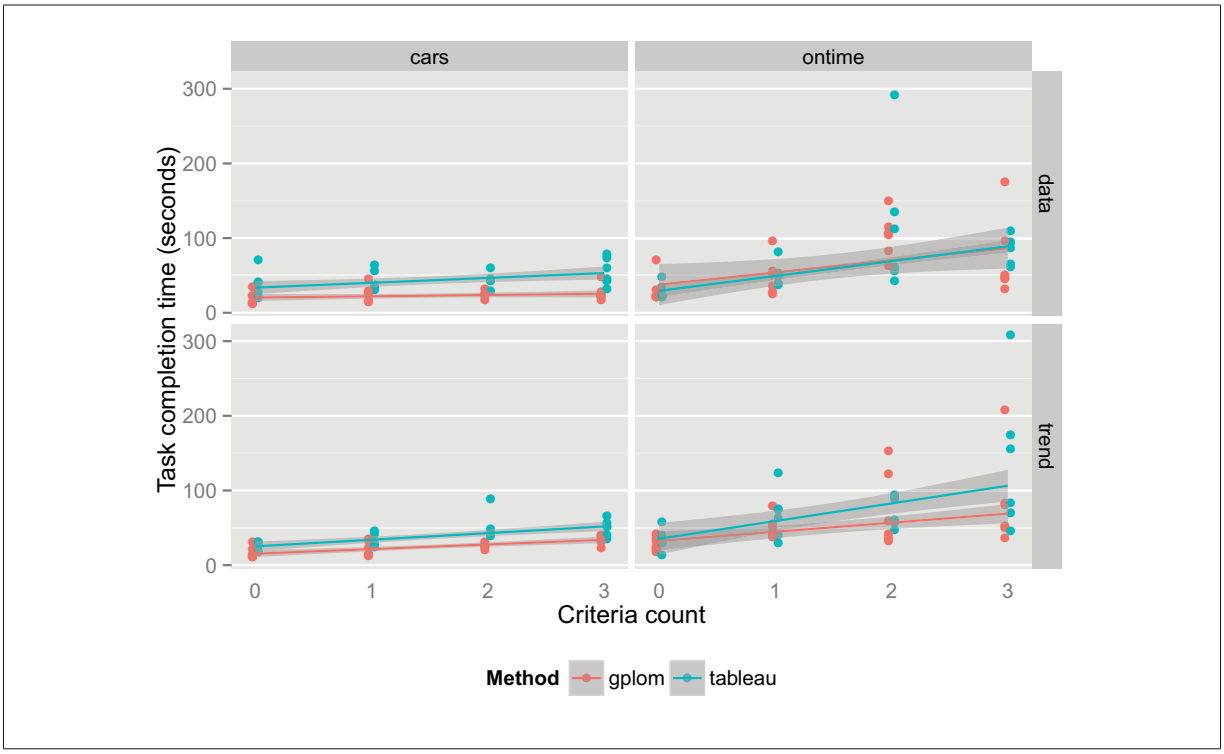

Figure 2.12 Task completion time for each method, as a function of number of criteria, broken down by dataset and question type. A robust linear model was fitted to yield the straight lines.

($p < 0.01$), with time increasing with criteria count. There was no significant difference in time between GPLOM and Tableau for the OnTime dataset, although GPLOM had a lower median time (48.68 seconds for GPLOM vs 59.58 seconds for Tableau).

Examining Figure 2.12, we note that the case where GPLOM seemed to have the least advantage with respect to Tableau was with "data" questions about the OnTime dataset when the criteria count was 2. This particular case corresponds to the only question that required the user to use the Count aggregation operator in GPLOM. In hindsight, we recall several users having difficulty with this question, and suspect that this question was relatively easier in Tableau because Tableau has a pre-defined variable "Number of records", obviating the need for users to select a Count aggregation operator in Tableau.

A logistic regression revealed that questions about the OnTime dataset had a significantly higher error rate than questions about Cars. GPLOM had a lower overall error rate than Tableau, but not significantly.

Post-questionnaires asked users to rate the two interfaces against nine criteria such as “intuitive”, “easy to learn”, etc. On average, users gave a higher (i.e., more positive) rating to GPLOM than Tableau for all of these criteria, but Wilcoxon signed rank tests showed that only two were significant: users judged GPLOM to be significantly more “fast” ($p < 0.01$) and significantly more “fluid” than Tableau ($p < 0.05$). Additional details about the statistical analysis are available in Annex I.

2.5.2 Discussion

We did not attempt to subtract the full page load time per filter incurred by the GPLOM prototype from the recorded time taken by the participant to answer, as there is no way to differentiate between the user waiting for Chrome to render the page and the user thinking about the next filter to enter while the page is loading. It is possible that, if the page load time were reduced with better coding, this would further differentiate GPLOM from Tableau, as Tableau did not incur such an overhead.

On the other hand, the error rate with Tableau was sometimes rather high, and this may be because the users were too inexperienced with it, despite the warm up trials. It is possible that a follow-up study with more experienced users would yield different results.

Nevertheless, in our study, GPLOM resulted in a lower median time in both datasets, and a significantly lower time in the Cars dataset. A possible explanation for this difference would be the difference between the process of building a filter in each visualization.

In Tableau, the process to build a filter comprises the following steps:

a. Pick the categorical variable to filter from the list of dimensions

b. Drag the selected categorical variable to the filter shelf

c. Select the desired value for the filter from the list of possible values for the categorical variable

d. Click OK to dismiss the filter dialog box

On the other hand, in GPLOM, the process to build a filter requires the following steps:

a. Locate one bar chart whose x axis corresponds to the categorical variable to filter

b. Locate the particular bar that corresponds to the desired value on which to filter by hovering over the bar and reading its associated tooltip

c. Double-click the bar

Alternatively, the user can build a filter in GPLOM in the following fashion:

a. Move the mouse cursor to the search box and click it

b. Enter the desired value to search for using the keyboard

c. Move the mouse cursor to one of the highlighted bars on a bar chart

d. Double-click the bar

Another explanation for the faster performance of GPLOM relative to Tableau would be the dimensional shedding feature of GPLOM. As participants drilled down in GPLOM by double clicking, the number of displayed charts was correspondingly reduced and the possible values on each chart's horizontal axis only contained the list of allowed values. In contrast, Tableau's design requires the list of dimensions (categorical variables) to stay static and building a filter often listed values incompatible with other filters. For example, even if a previous filter filtered out cars by Asian manufacturers, Honda and Toyota would still appear if the user attempted to add a filter for the manufacturer's name. Furthermore, when users built an invalid combination of filters, Tableau displayed no data at all, which stumped some participants and caused them to search (often for an extended period of time – see for example the outlier points in Figure 2.12) for a reason as to why the display was completely blank. As GPLOM always shows

all available data and filtering is done by picking a particular subset of the displayed data, it is impossible for a user to build such a filter combination.

Another problem participants encountered with Tableau was their building of a chart that contained too many categorical variables or did not answer the question they were asking; on the other hand, some participants answered some questions using the wrong chart in GPLOM, so the problem could be one of user education or wanting to please the experimenter by answering something.

2.5.3 Improvements

Several improvements can be made to the GPLOM prototype. In its current iteration, the search box only contains the data contained in the database, without mapping it to more user friendly concepts (carrier name "WN" instead of Southwest or "1" as a day of the week, instead of Monday). This confused some of the users, who tried several times, unsuccessfully, to get the search box to find the values they were looking for. Ensuring that there is a rich data dictionary that has multiple synonyms for values would significantly improve the users' experience in that regard.

Another problem was that the search box's color contrast was insufficient (see top right of Figure 2.4) and eight of the twelve users missed it entirely during the five minute exploration period. Improving its contrast and adding a magnifying glass icon might make it easier for users to discover the feature. Even when they were told that the search box existed, most users did not use it, instead using the mouse to find and select values to filter on.

One significant problem that was repeatedly encountered during user testing of the GPLOM prototype is the lack of clear affordances for interaction. Users did not seem compelled to click, much less double click, on charts. During the exploration phase, out of twelve users, only one found that it was possible to filter data by double clicking on bars, although some tried right-clicking (which only brought Chrome's default right-click menu). This lack of clear affordances meant that users often tried to click and double click on the brightly colored bendy

highlight, which did nothing; in retrospect, it seems like an obvious affordance for user interaction which could be used for highlighting and filtering.

The associative highlighting, while useful for part-to-whole comparisons, was often misunderstood by users; it was almost never used to answer questions on datasets, even though it displays the exact same data that double-click filtering on a particular value would.

Another misunderstood feature was the kernel density estimate plot, which confused users much more than it helped them. We postulate that histograms, density plots, Q-Q plots, rug plots and other statistical tools, while very important to evaluate distribution shape, are unlikely to be understood by average business users.

2.6 Conclusion and Future Directions

Despite a large variety of charts and visualizations, it remains unclear to many non-expert users how to visualize the contents of a typical database in a way that gives them an overview of variables they may not be familiar with. Many advanced techniques have been proposed (Wong and Bergeron (1997); Grinstein *et al.* (2001); Keim (2002)), but most of these have seen limited real-world deployment, and almost none of them are designed for the simultaneous visualization of multiple categorical and continuous variables, with the exception of Emerson *et al.* (2013). Both Emerson *et al.* (2013)'s approach and the GPLOM can give users a visual overview of more than 10 variables, allowing them to visually scan for interesting relationships and allow for serendipitously discovering outliers or thinking of unplanned questions for further analysis.

Compared to Emerson et al., GPLOM (1) only uses three kinds of charts, the minimum number necessary to cover the three kinds of pairings of variables, which may make it easier to understand for non-expert users and scale better to high-cardinality categorical variables; (2) demonstrates two ways of interactively linking charts, through bendy highlights and associative highlighting; (3) demonstrates text search to quickly find values of interest; and (4) saves screen space by only displaying the lower triangular half of the matrix. We also presented ex-

perimental evidence that GPLOM is sometimes significantly faster than Tableau, a commercial product, for the kinds of questions we tested.

Future work might compare the performance of the reversed GPLOM (Figure 2.11) with the upright version. During the exploration phase of our study, most of the participants seemed to explore the software from top to bottom, left to right and spent most of their time trying to understand the heat maps. The reversed GPLOM would mean that the first visual elements encountered by users would be barcharts, which are easier to understand. This might better ease novice users into the GPLOM.

Future work could also compare GPLOM with other MDMV visualizations or database tools, such as xmdv, ggobi or Mondrian.

The GPLOM prototype could be modified to accommodate a wider range of user skills. Novice users could be shown only the matrix of barcharts, while more advanced types of plots (such as those of Emerson *et al.* (2013)) could be available for expert users.

There might also be hybrid ways to combine the matrix layout of GPLOM with the dimensional stacking of Polaris / Tableau, giving the user more control over the tradeoff of number of charts and level of detail.

Finally, we plan to explore ways of improving the performance of GPLOM with extremely large datasets. While performance of the GPLOM prototype on the complete OnTime dataset ($\approx$144 million records) was still within acceptable bounds for interactive exploration, it required the usage of an in-memory columnar database running on a server with 16 dual-core processors equipped with 512 gigabytes of RAM. Incremental approaches for large data visualization, such as VisReduce (Im *et al.* (2013b)), could also be applied to reduce the perceived system latency.

CHAPTER 3

VISREDUCE

3.1 Introduction

Visualization for business intelligence often involves querying large databases to generate plots such as line charts, barcharts, scatterplots, or potentially more exotic visualizations (Wong and Bergeron (1997); Grinstein *et al.* (2001); Keim (2002)) such as parallel coordinate plots (Inselberg (1985)). These visualizations provide more insight when the user can interact with them to quickly refine queries, drill down, or choose new paths of exploration. Unfortunately, common back-end systems for processing very large datasets have one or both of the following problems: (1) they exhibit high latency, precluding feedback at interactive rates, or (2) they require expensive pre-computations (such as indices or datacubes) that accelerate restricted classes of queries, but do not accelerate *all* of the common queries involved in data visualization.

For example, systems based on SQL are not designed to run arbitrary code as part of a query[1]. Hence, generating a scatterplot or parallel coordinates plot of a large dataset with SQL requires running a query that transfers all the data tuples from the database to the client, and then generating the plot on the client. The client becomes a bottleneck, and the work of generating the visualization cannot be distributed over multiple nodes. Scalability is thus severely limited.

OLAP datacubes (Chaudhuri and Dayal (1997)) support fast aggregation queries, but only over the dimensions that were included during the construction of the cube. With large datasets involving 20 or more dimensions, constructing a "full" cube with all dimensions is often not feasible (Liu *et al.* (2013)) due to memory restrictions, and queries involving dimensions that are left out will not be possible.

[1]Technically, many DBMS support user defined aggregation functions, as proprietary extensions to SQL. These have different caveats depending on the DBMS. For example, MySQL requires them to be written in C, Oracle requires PL/SQL, C, C++ or Java, SQL Server requires them to run on the .NET framework while PostgreSQL supports many languages. Furthermore, the C APIs are incompatible between DBMS. Also, depending on the database and foreign language combination, there may be a significant overhead to calling a foreign function millions of times. Finally, DBMS-internal languages have no intrinsic support for image manipulation.

MapReduce (Dean *et al.* (2004)) has recently gained attention as a tool for processing large data, but is designed for batch jobs, and has high latency. Just starting up a new job can require several seconds with typical implementations. To quote Heer and Shneiderman, "While popular platforms for large data analysis such as MapReduce achieve adequate throughput, their high latency and lack of online processing limit fluent interaction" (Heer and Shneiderman (2012)).

Interactive visualization of large data is an important problem, covering two challenges listed by Johnson ("human-computer interaction" and "scalable, distributed, and grid-based visualization" (Johnson (2004))) and two more listed by Chen ("usability" and "scalability" (Chen (2005))). Tableau, a leading commercial front-end for visualization of databases for business intelligence, has only limited features for dealing with very large data. However, more powerful solutions would clearly be valuable: "Tableau has users with very large databases who are willing to wait minutes for database queries to run so that they can see a graphical view of their valuable data. However, users do not want interactive experiences that include such pauses." (Mackinlay *et al.* (2007))

We present a simple yet promising solution called VisReduce, that (1) allows queries to run arbitrary code on multiple computers called *worker nodes* (unlike SQL-based solutions); (2) scales up in speed as the number of worker nodes is increased; (3) gives continual feedback to the client about the progress of a query, so that the user knows roughly how long they will need to wait; (4) incrementally updates the result displayed by the client, so that the user sees an approximate visualization of the data processed so-far during the entire query, as illustrated by the bar charts of Figure 3.1. The visualization is displayed within 1 second of launching the query and is updated frequently and continually during the query, gradually converging toward the final result, allowing the user to cancel a query before it has finished if they so desire. VisReduce achieves these properties by avoiding all large transfers of data between nodes, never writing large output files to disk, leveraging compressed columnar storage data formats, and keeping runtime environments on worker nodes "warm" (i.e. persistent) rather than starting

up new runtime environments each time a new query is initiated so that processing can start and visual feedback can be displayed within 1 second.

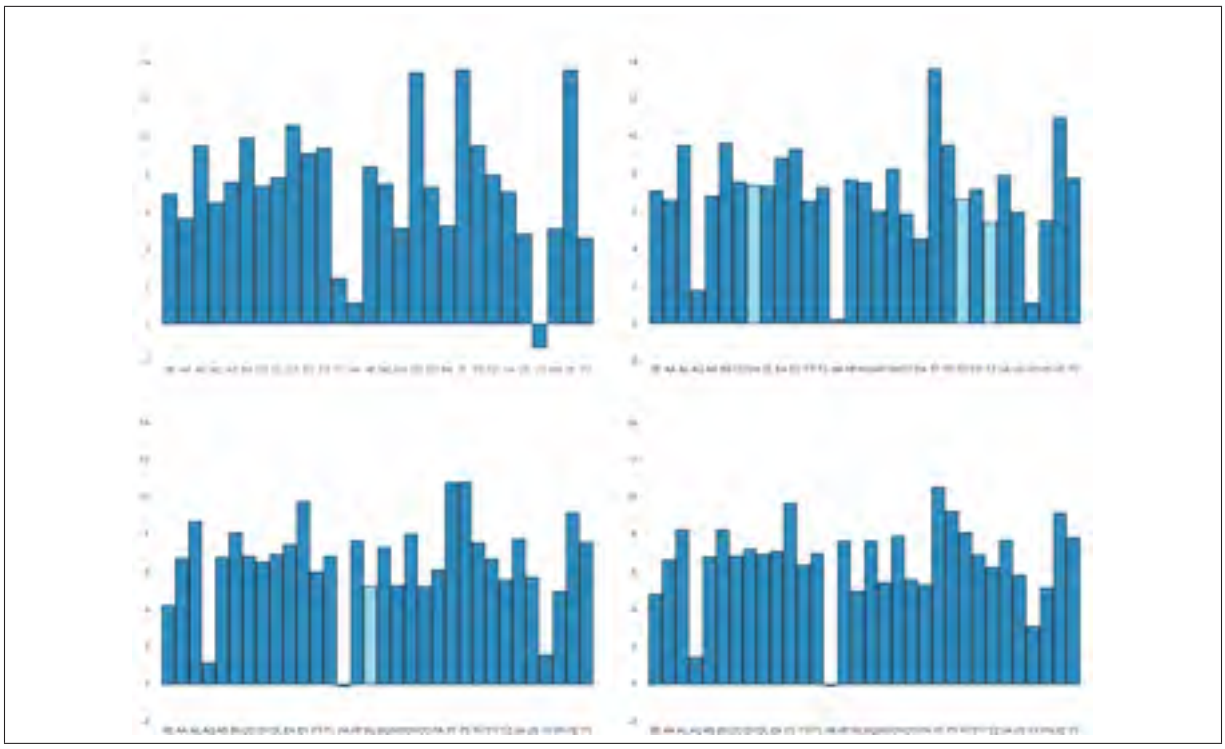

Figure 3.1 In this example, the output being displayed by the VisReduce client is a barchart. As the query progresses, the barchart is updated incrementally, shown here as a sequence of screenshots, starting at top left and ending at bottom right. Respective times since the start of the query are 270 ms, 1137 ms, 3988 ms and the completed result at 5021 ms shows average delay for all 149.5 million flights. The bars initially oscillate in length, but quickly converge to their final lengths as the query progresses and becomes more accurate. As the query progresses, new bars are occasionally created and inserted (these have been manually highlighted in this figure with a lighter color).

We evaluated VisReduce by comparing its performance with three other popular systems: Apache Hive[2] (built on top of Hadoop MapReduce), Cloudera Impala[3] (a recently-released implementation of Dremel for Hadoop), and PostgreSQL[4]. Tests were performed with the On-Time[5] flight data set, which has approximately 150 million records and over 100 columns. Of these systems, VisReduce is the only one that provides an incrementally updated visualization

[2]https://hive.apache.org/
[3]http://impala.io/
[4]http://www.postgresql.org/
[5]http://www.transtats.bts.gov/Fields.asp?Table_ID=236

during the query, and the results of our tests indicate that it also completes queries significantly faster than the other systems. We feel that the design ingredients of VisReduce provide valuable lessons for the design of future interactive visualizations engines for large datasets.

3.2 Previous Work

Rapid interactive visual queries on databases were pioneered with Shneiderman's *dynamic queries* (Shneiderman (1994)), which emphasized the value of providing real-time feedback to the user for a tight interaction loop. TreeJuxtaposer (Munzner *et al.* (2003)) is an example of a visualization that achieves a guaranteed constant frame rate during navigation through a large tree structure. It achieves this by progressively rendering data into the video card's front buffer (rather than the usual back buffer), drawing the more important data and landmarks first, and stopping the rendering whenever a new frame must be started. This way, if there is not enough time to draw the full data set in the allocated time for a frame, the user at least sees the most salient information before the next frame is started. VisReduce is designed to scale up to much larger data sets, but still adheres to the idea of displaying a visualization that is updated often (several times per second) and becomes increasingly accurate as the query progresses.

We now survey the most relevant types of backends for large data processing.

SQL-based systems are programmer-friendly because the query language is familiar and easy-to-understand. Examples of systems that expose an SQL-like language include Dremel (Melnik *et al.* (2010)), a query system developed at Google for low-latency querying of large data sets stored on their infrastructure, and Cloudera Impala, a recently-released open-source implementation of the same concept. As explained in the introduction, such systems cannot efficiently generate a scatterplot or parallel coordinates plot, because they would require all the data to be first transferred to the client for rendering.

Of particular note is the work of Hellerstein et al. (Hellerstein *et al.* (1997)) on implementing online aggregation inside of a database engine, in which they explain how they implemented online aggregration of simple aggregate statistics (sum, count, avg, var and std dev) in Postgres.

OLAP datacubes (Chaudhuri and Dayal (1997)) only accelerate queries on the cubes that have been pre-computed. As mentioned in the introduction, with large data sets, these cubes cannot incorporate all dimensions, otherwise they become too large. The imMens (Liu *et al.* (2013)) system has a clever workaround for this problem: it only pre-computes small cubes of 3 or 4 dimensions, and can afford to pre-compute many such cubes, each with a different set of 3 or 4 dimensions. This allows the user to perform brushing and linking on overviews of data with very rapid visual feedback. However, business intelligence tasks can require the user to filter along 3 or 4 dimensions (e.g., to examine only the data for a particular country, a particular year, and a particular month) and then explore further, and such filtering along many dimensions means these small datacubes will no longer be useful. In addition, although many small cubes require far less space than one full cube, they can still require significant space and pre-computation time: in a data set with 20 dimensions, computing all possible 4-dimensional cubes requires storage for $\binom{20}{4} = 4845$ cubes.

MapReduce (Dean *et al.* (2004)) is a framework used at Google for writing distributed programs that can be run on large clusters of computers. An open-source implementation, Hadoop MapReduce, is a popular approach for handling large datasets. Both are optimized for throughput of large batch jobs, not latency, and thus involve overhead that is a significant barrier to real-time interaction. Just starting up a new job can take several seconds, partly because new Java Virtual Machines (JVMs) must be started on each worker node. Furthermore, running MapReduce on large datasets involves moving lots of data between machines (Figure 3.2). Finally, common implementations of MapReduce do not return any partial results; the client must wait for a job to complete before seeing feedback.

Some previous work (Jermaine *et al.* (2006); Joshi and Jermaine (2008); Fisher (2011)) has studied how to structure databases so that queries iterate over data in a statistically random order. This allows confidence bounds on the current result to be computed and displayed throughout the query, so that the user not only sees the current result, but also bounds on what the final result may be, giving the user more information to decide whether they can stop a query before it completes. User studies (Fisher *et al.* (2012)) have shown that such

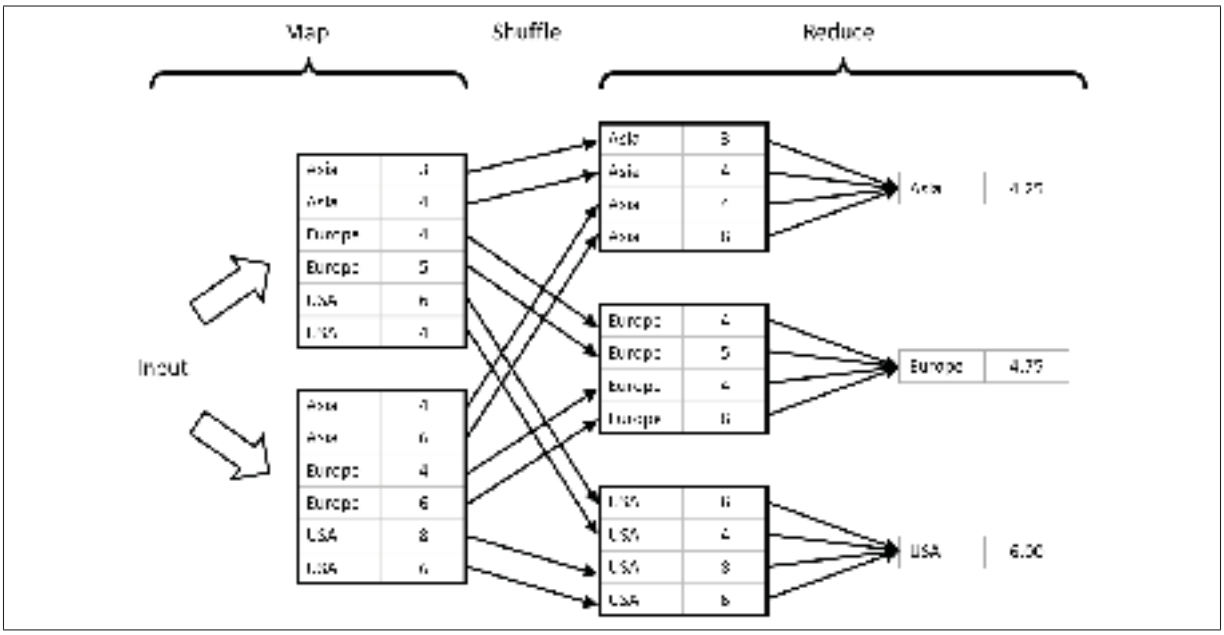

Figure 3.2 Computing averages with MapReduce. The map, shuffle, and reduce operations are each distributed over multiple nodes. Tables with thick borders may be very large, resulting in slow disk I/O operations and heavy network traffic.

confidence bounds provide end users with valuable information to accelerate decisions by the user. Unfortunately, this approach comes with a significant up-front cost: randomly shuffling all records in the database (Jermaine *et al.* (2006)) or constructing an "ACE Tree" (Joshi and Jermaine (2008)) structure which itself requires multiple complete passes through the dataset. Our current VisReduce prototype performs no such pre-processing but this also means we cannot provide confidence bounds on the incrementally updated result during a query. Note also that, because VisReduce uses a columnar data format, it can greatly benefit from having data that is *not* randomly shuffled: run-length encoding can greatly compress the columns if there are many repeated values. For example, generating an "average sales by month" barchart of a large dataset with VisReduce would only require reading in two columns, one of which (the "month" dimension) may be greatly compressed because it contains many repeated values, thus saving disk read time.

There have also been efforts to extend MapReduce to allow for online aggregation (Böse *et al.* (2010); Condie *et al.* (2010)). These approaches require some time to start up before they can return results. For example, Figure 4 of Condie *et al.* (2010) shows almost 20 seconds

elapsing before any progress is made. The latency achieved with VisReduce is much lower (under 1 second) and more suitable for interactive feedback; this faster start up time allows rapid sequences of partial queries to be explored with no perceived waiting time by the user, such as a user executing multiple sequential drilldowns in GPLOM (Im *et al.* (2013a)) without waiting for query completion.

Table 3.1 summarizes some key differences between SQL, MapReduce, and our proposed VisReduce.

Table 3.1 A comparison of SQL, MapReduce, and VisReduce. VisReduce is the only approach to have all the advantages listed, because it is designed especially for interactive visualizations.

		SQL	MapReduce	VisReduce
Gives feedback about progress of query		no	Yes	Yes
Returns results incrementally		no	not typically	Yes
Queries run within a persistent runtime environment, reducing start-up time		Yes	no	Yes
Distributed, hence scalable to large data sets		not typically	Yes	Yes
Uses a no-unpack data format, reducing file read operations		sometimes	sometimes	Yes
No large transfers of data over the network	Raw data stays on same node (even when generating a scatterplot or parallel coordinates plot)	no	Yes	Yes
	Output from worker is 'small' (a few MB at most), avoiding file write operations	no	no	Yes
Queries can run arbitrary code		no	Yes	Yes

3.3 Description

VisReduce differs from MapReduce by making two assumptions in order to increase the performance to interactive levels:

- the resulting output aggregate is small enough to fit in memory and be transmitted in a reasonable time (less than 250 ms) over the network;

- there always exists an inverse aggregate, so that partial results can be removed from the output aggregate.

An overview of VisReduce running on two worker nodes is shown in Figure 3.3, which can be contrasted with Figure 3.2 for MapReduce. One key difference is that MapReduce doesn't return a result until the entire job is finished. With VisReduce, however, results are computed incrementally, so that at time t, the 2 workers produce partial results $r_{t,1}$ and $r_{t,2}$, respectively. These results are combined by the master node with its previous partial result yielding $r_t = r_{t-1} \oplus r_{t,1} \oplus r_{t,2}$. In the example shown in Figure 3.3, this partial result r_t might be displayed by the client in the form of a barchart, with the heights of bars gradually converging toward their final heights, i.e., r_t converging toward its final value as t increases.

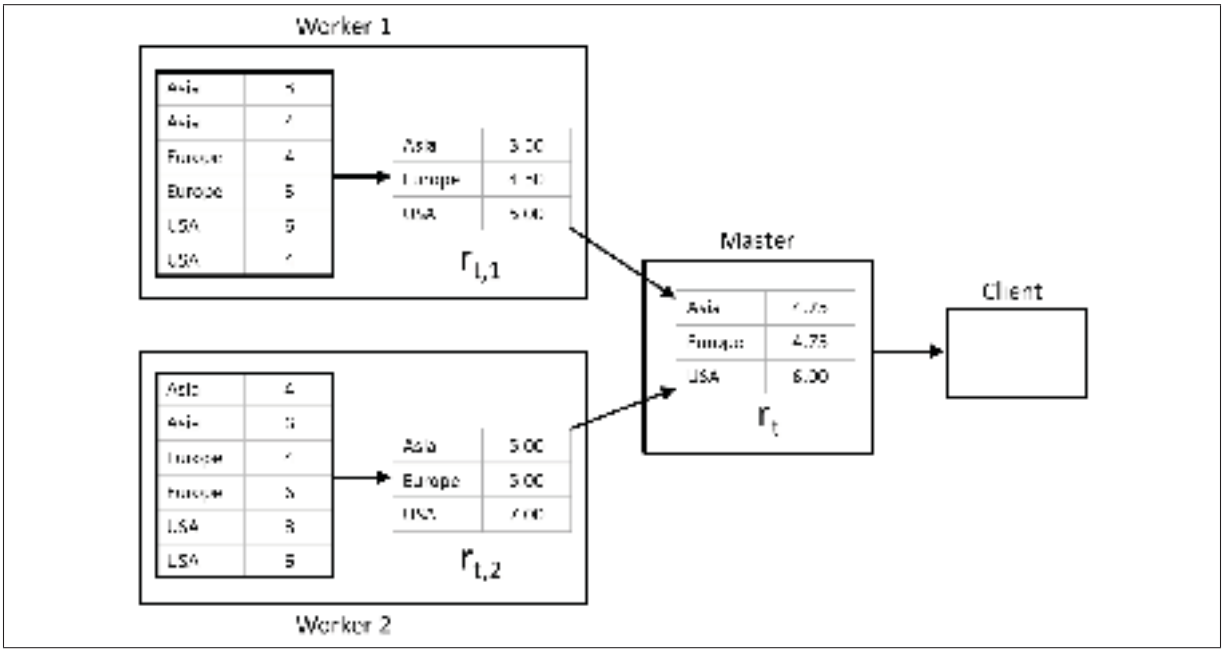

Figure 3.3 Computing averages with VisReduce over two worker nodes. Tables with thick borders may be very large, but do not leave the worker node they are stored on. Only relatively small amounts of data are transferred between nodes. At time t, each worker w produces result $r_{t,w}$; the master node assembles these into result r_t to be displayed by the client. Specifically to compute averages, both the sum and the number of data points seen need to be tracked.

Another key difference between Figure 3.2 and Figure 3.3 is seen by noting which tables are "big" (drawn with a thick border). In Figure 3.2, there are 5 big tables, requiring large volumes

of data to be transferred over the network. MapReduce supports an optional optimization where a combiner is run over the output of the mapping phase before transmitting it over the network, but it still requires writing the potentially large output of the mapping phase to disk. With VisReduce, however (Figure 3.3), the two large tables remain on their respective worker nodes, and only small results $r_{t,w}$ and r_t need to be transferred between nodes with no disk writes. In Figure 3.3, the transmitted results are heights of bars in a barchart, and the master node's $\oplus$ operator simply computes average values. Alternatively, if we were using VisReduce to compute a scatterplot or parallel coordinates plot, the partial results $r_{t,w}$ and r_t could be bitmap images, with the master node's $\oplus$ operator performing image compositing.

Because the partial results $r_{t,w}$ and r_t are small in size, VisReduce doesn't need to write results to disk, saving time.

Notice also that, if we increase the number of worker nodes, VisReduce's speed will increase almost linearly, because worker nodes can work in parallel and still only send small results over the network, even if the raw data set grows in size.

Also, no matter how large the dataset is, workers can send frequent partial results to the master (several times per second), to display incrementally updated feedback to the user. This is unlike most database approaches.

As a partial result can be "removed" by inverting it and reducing it into the current state, results can be sent to the client without waiting for completion of an input segment while still maintaing fault tolerance. In contrast, MapReduce's fault tolerance mechanisms require an input segment to be completely mapped before reducing the mapper's output, leading to long delays before the first results are sent to the client in online approaches, as shown in Figure 4 of Condie *et al.* (2010).

Unlike MapReduce, the runtime environments on VisReduce's worker nodes are kept "warm", to decrease start-up time.

VisReduce further saves time by using a simple columnar data store, which is combined with a dictionary encoding (e.g., replacing each string "Dallas, TX" with an integer value) and run-length encoding. Unlike a general database, with VisReduce we care much more about fast agregation performance, and not individual row lookups or updating the data.

Previous work has demonstrated the performance advantages of such columnar approaches. Pavlo *et al.* (2009) compared Hadoop MapReduce with two commercially available massively parallel databases — an unnamed row-oriented database[6] and a column-oriented one (Vertica) — and show that the column database is faster than the other two approaches for aggregation workloads. Abadi *et al.* (2006) discuss how adding various simple compression techniques to a column-oriented DBMS offer significant gains in both query run time and storage size.

3.4 Theory

Formally, if R is the set of possible results (e.g., key-value pairs for barcharts, or bitmap images of scatterplots), then the $\oplus$ operator used to combine partial results is a binary operator from $R \times R \mapsto R$. We furthermore require that $(R, \oplus)$ be an Abelian group:

Closure $\forall a, b \in R \Rightarrow a \oplus b \in R$

Associativity $a \oplus (b \oplus c) = (a \oplus b) \oplus c$

Commutativity $\forall a, b \in R, a \oplus b = b \oplus a$

Identity element $\forall a \in R, \exists \mathbf{0} \in R$ such that $a \oplus \mathbf{0} = a$

Inverse element $\forall a \in R, \exists -a \in R$ such that $a \oplus -a = \mathbf{0}$

Associativity and commutativity mean that the master node can combine partial results from W workers in any order with $r_t = r_{t-1} \oplus r_{t,1} \oplus \ldots \oplus r_{t,W}$. The existence of inverses means that the master node can also remove a partial result $r_{t,i}$ from r_t if the master decides that worker

[6]Many commercial database vendors prohibit publishing benchmarks in their licensing agreements. In the case of Pavlo et al., the database is known as DBMS-X, "a parallel DBMS from a major relational database vendor."

node i is malfunctioning and the partial result needs to be recomputed, either by a different worker node or by the same worker node.

We further define an operator $\odot : R \times D \mapsto R$ which combines a single data element $d \in D$ with a partial result $r \in R$ to produce a new partial result. This operator is used by the worker nodes to construct their partial results. Each worker node w begins with an empty result $\mathbf{0} \in R$ (e.g., $\mathbf{0}$ could be a blank bitmap image) and combines it with data elements to generate $r_{t,w} = \mathbf{0} \odot d_1 \odot \ldots \odot d_n$.

The $\odot$ operator in VisReduce is analogous to the map operator in MapReduce. However, with MapReduce, the map operator must map to a list of key-value pairs, whereas our $\odot$ operator can map to any object of reasonable size, including key-value pairs and bitmap images.

VisReduce jobs can be implemented by defining the four side effect-free functions listed below. These could be the basis for four methods in a Java interface or an abstract base class in C++, that must be implemented by the programmer.

- a $\oplus$ function that combines two partial results
- a $\odot$ function that combines a partial result with a data element
- an identity function which returns the empty result (identity element) $\mathbf{0}$
- an inverse function which returns the inverse $-r$ of a partial result r

Those four functions, or methods, can be packaged together to form a work-object, which can then be distributed across a cluster (as a single `.class` file, in the case of Java) for parallel processing.

3.5 Implementation

Our VisReduce prototype has been implemented as a web application using Play[7] for serving the web content and Akka[8] for clustering and actor-based inter-node communication. Each

[7] http://www.playframework.org/
[8] http://akka.io/

worker node has a copy of the entire data set which is saved on disk in a compressed and bit-packed read-only column oriented format, split in several segments called *tablets*. Upon receiving a request from a client, the master node gathers a list of tablets for a particular table, then sends a list of tablets and a work-object to each worker node. Worker nodes then run the work-object on the tablet, sending the resulting state from the processing to the master node and requesting additional work.

The master node aggregates the results of tablet processing on the cluster as they are received and pushes back the aggregate onto the client at an appropriate speed for the client, which updates the data visualisation shown to the user. To ensure interactivity, the master node can request a partial result from the processing of a tablet by a worker, so that slow tasks running on large tablets still display partial aggregates.

Complete node failures are detected using the φ accrual failure detector (Hayashibara *et al.* (2004)), while exceptions caught by VisReduce are reported to the master node. Should a node fail during tablet processing with its partial state sent to the client, the partial state can be removed by inverting it and sending the negative delta to the client, while scheduling the execution of the failed tablet on another node, thus ensuring a consistent final state.

3.5.1 Examples

The bar chart in Figure 3.1 was generated using simple operations for each operator:

$\mathbf{0}$ Initialize an empty associative array of grouping keys to sum and count pairs

$-$ Invert all sums and counts in the associative array

$\oplus$ Combine both associative arrays, removing values with a count of 0

$\odot$ Add or update the value in the associative array

The resulting associative array is then turned into an animated bar chart by the VisReduce client. The heat map of Figure 3.4 was generated using the following operators:

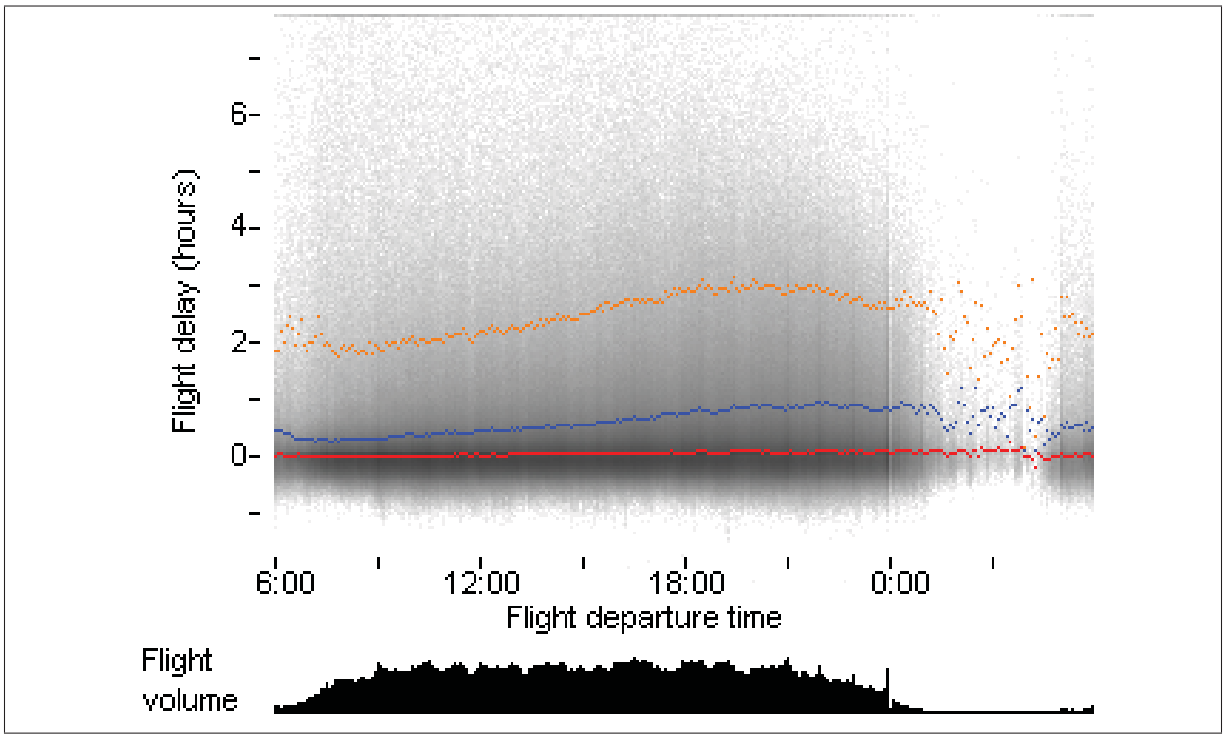

Figure 3.4 Example of a heat map and accompanying histogram of arrival delay by time of day, with the median, 90th and 99th percentiles of the arrival delay highlighted.

0 Initialize an empty array of bins, each containing a count of 0

$-$ Invert all counts of the array

$\oplus$ Sum both arrays together

$\odot$ Increment the count for the appropriate bin

The bins are then turned into an image and the appropriate percentiles for each time block are highlighted.

3.6 Architecture

We use a client-server model where the master node brokers communications between the client and the workers. First, the client sends a work-object to the master, containing the functions to execute over the dataset. The master then breaks the work to be done over all the input tablets and splits the execution of the work-object over the workers. As the workers

process the dataset, partial results are returned to the master on a regular basis, which are then aggregated with the $\oplus$ operator and sent back to the client. Thus, the client sees an incremental view of the data that converges towards the final value of r as more data is loaded from disk.

Communication between the parts of the system are implemented using the actor model of computation in which messages are sent between actors using the ! operator and messages are processed by an actor one at a time from its mailbox. We define the following three messages:

- `BeginComputation`
- `PartialResult`
- `CompleteResult`

Given those messages, the message flow in a system with one worker and a dataset that has two tablets would be as illustrated in Figure 3.5.

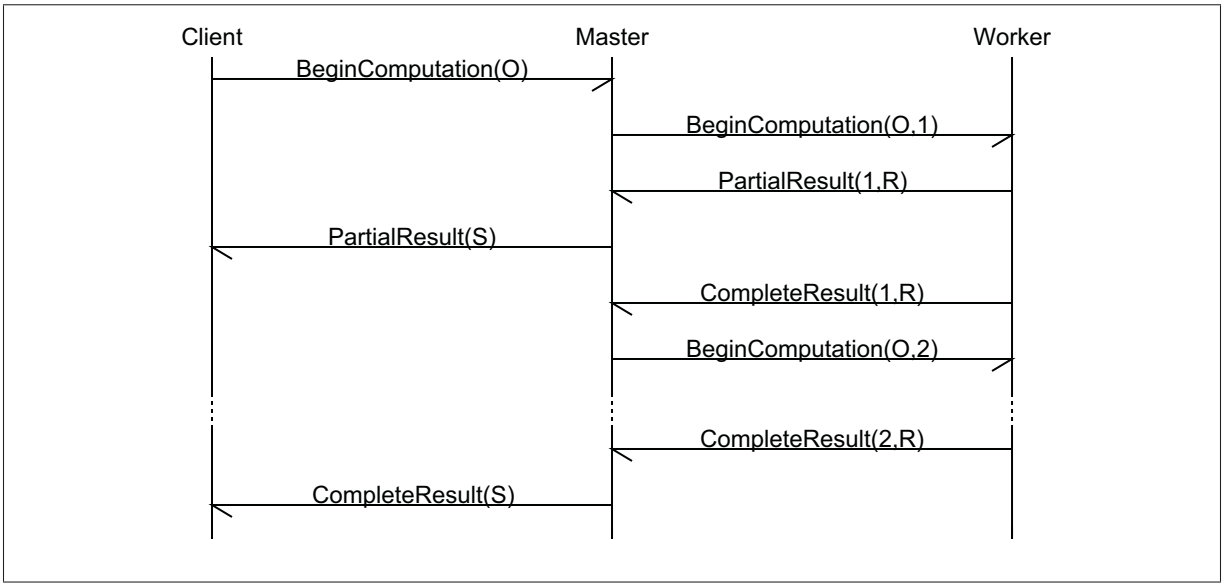

Figure 3.5 Message flow for a data set with two tablets

3.6.1 Client Actor

The client actor is responsible for displaying the visualisation to the user and maintains an internal state S for the computation. It can send the `BeginComputation` and must handle the `PartialResult` and `CompleteResult` messages.

Of all the actors in the system, the client actor is the simplest. It simply sends a message to the **master** actor to begin the computation, then either merges the incoming partial results into its local state or displays the final state. Depending on the type of computation asked of the cluster, it can display different types of charts, such as bar charts, scatterplots or density plots.

Algorithm 3.1 shows how the client actor can initiate the computation on the cluster by sending the `BeginComputation` message to the **master** and handles the incoming `PartialResult` and `CompleteResult` messages by updating the display.

Algorithm 3.1 Client Actor

```
▷ S the computation state
procedure StartComputation(O)
    ▷ O a work-object to execute
    S ← 0
    master ! BeginComputation(O)
    Update display
end procedure
procedure HandlePartialResult(R_p)
    ▷ R_p a partial result
    S ← S ⊕ R_p
    Update display
end procedure
procedure HandleCompleteResult(R_c)
    ▷ R_c the complete result
    S ← R_c
    Update display
end procedure
```

3.6.2 Worker

The worker actor is responsible for calculating the value of r over by iterating over the records on a single tablet and reporting on the state of the computation to the master as appropriate. It receives `BeginComputation` messages and sends `PartialResult` and `CompleteResult` messages back to the **master**, as shown in algorithm 3.2.

Each worker is completely independent of all the other workers, so that VisReduce can scale near-linearly with the number of workers, up to the number of input tablets. Furthermore, a single node with multiple cores or CPU sockets can run multiple worker actors to increase parallelism and total throughput.

Algorithm 3.2 Worker Actor

procedure HANDLEBEGINCOMPUTATION(O, I_t)
  ▷ O the work-object
  ▷ I_t an input tablet identifier
  ▷ D the data in the input tablet I_t
  ▷ S the computation state
  $S \leftarrow 0$
  for all d in D **do**
    $S \leftarrow S \odot d$
    if we haven't updated the master in a while **then**
      master ! `PartialResult`(I_t, S)
    end if
  end for
  master ! `CompleteResult`(I_t, S)
end procedure

3.6.3 Master Actor

The master actor dispatches work to **worker** actors and reports the results of the processing back to the **client**. Like the **client**, it maintains an internal computation state S. It also maintains a state ΔS which contains the results that have not yet been sent to the **client**. It handles all three messages, as shown in algorithm 3.3, and sends `BeginComputation` messages to worker actors.

Algorithm 3.3 Master Actor: External message handling

▷ $I_\forall$ a list of all input tablet identifiers
▷ I a list of unprocessed input tablet identifiers
▷ W a list of all the workers
▷ P a map of partial results
▷ A a map containing which actor is working on a particular input tablet identifier
▷ S the computation state
▷ ΔS the computation state not yet sent to the **client**

procedure HANDLEBEGINCOMPUTATION(O)
    ▷ O the work-object
    $S \leftarrow \mathbf{0}, \Delta S \leftarrow \mathbf{0}, I \leftarrow I_\forall$
    for all worker in W **do**
        if $I \neq \varnothing$ **then**
            worker ! `BeginComputation`$(O, I.head)$
            $P[I.head] \leftarrow \mathbf{0}, A[I.head] \leftarrow$ **worker**
            $I \leftarrow I.tail$
        end if
    end for
end procedure

procedure HANDLEPARTIALRESULT(I_t, R_p)
    ▷ R_p the partial result
    ▷ I_t an input tablet identifier
    if $P[I_t] \neq \varnothing$ and $A[I_t] =$ **sender then**
        $S \leftarrow S \oplus R_p \oplus -P[I_t], \Delta S \leftarrow \Delta S \oplus R_p \oplus -P[I_t]$ ▷ Merge R_p into S and ΔS
        $P[I_t] \leftarrow R_p$
    end if
end procedure

procedure HANDLECOMPLETERESULT(I_t, R_c)
    ▷ R_c the complete result
    ▷ I_t an input tablet identifier
    $S \leftarrow S \oplus R_c \oplus -P[I_t], \Delta S \leftarrow \Delta S \oplus R_c \oplus -P[I_t]$ ▷ Merge R_c into S and ΔS
    if $P[I_t] \neq \varnothing$ and $A[I_t] =$ **sender then**
        $P[I_t] \leftarrow \varnothing$
        worker ! `BeginComputation`$(O, I.head)$
        $P[I.head] \leftarrow \mathbf{0}$
        $I \leftarrow I.tail$
        if $I = \varnothing$ **then**
            client ! `CompleteResult`(S)
        end if
    end if
end procedure

Internally, a `Tick` message is sent at a regular interval to the **master** to ensure that the **client** gets regular updates. Algorithm 3.4 shows how ticks are handled.

Algorithm 3.4 Master Actor: Tick message handling

```
▷ I a list of unprocessed input split identifiers
▷ ΔS the computation state not yet sent to the client
procedure HANDLETICK
    if I ≠ ∅ then
        client ! PartialResult(ΔS)
        ΔS ← 0
    end if
end procedure
```

3.6.4 Fault tolerance

Algorithm 3.5 Master Actor: Fault handling

```
▷ I a list of unprocessed input split identifiers
▷ W a list of all the workers
▷ P a map of partial results
▷ A a map containing which actor is working on a particular input tablet identifier
▷ S the computation state
▷ ΔS the computation state not yet sent to the client
procedure HANDLEFAULT(I_t)
    ▷ I_t an input tablet identifier which failed
    S ← S ⊕ −P[I_t], ΔS ← ΔS ⊕ −P[I_t]      ▷ Merge the inverse of the partial result
    W ← W − A[I_t]                          ▷ Remove the worker from the pool
    I ← I + I_t                             ▷ Add the tablet so it gets processed again
    P[I_t] ← 0, A[I_t] ← ∅
    if W ≠ ∅ then
        worker ! BeginComputation(O, I.head)
        P[I.head] ← 0, A[I.head] ← worker
        I ← I.tail
    else
        No workers left, failure
    end if
end procedure
```

In VisReduce, fault detection is done using an implementation of the φ accrual failure detector of Hayashibara *et al.* (2004). This ensures that faults are detected within seconds and can be

recovered from without a significant impact on interactivity. If there is a fault or an exception that occurs during processing of a tablet (either detected by the runtime or a catastrophic node failure), $-s$ is merged back into the computation state, where s was the last sent accumulated state during the processing of the failed tablet.

3.7 Evaluation

VisReduce is designed to ensure that query performance is comparable to other non-incremental systems; it would not be very useful to see incremental results while having to wait significantly longer for any given query, compared to a non-incremental approach.

We evaluate our approach using a dataset of domestic US flights by major carriers from October 1987 until February 2013 and their associated delay information, for a total of 149,598,920 records with 109 columns. The dataset comprises 305 CSV files for a total of 65.74 GB. The size of the data set and its number of records are consistent with the findings of Rowstron *et al.* (2012) for typical analytical workloads on clusters of computers.

The query performance of VisReduce was compared with PostgreSQL, Apache Hive and Cloudera Impala. All systems were benchmarked using queries that spanned one, two and three columns. The one column query counted the number of flights by carrier, which resulted in 32 rows of output across all 149.5 million flights. The two column query calculated simple aggregate statistics (min, max, count and sum) for the arrival delay of flights grouped by carrier, which also resulted in 32 rows of output. Finally, the three column query calculated the same aggregate statistics for the arrival delay but grouped by origin and destination airport pairs, resulting in 8431 pairs of airports and their associated arrival delay information.

Queries were run on a five node cluster used for production analytical workloads at a commercial dating website during times when no other jobs were running. The cluster was comprised of heterogenous nodes equipped with either one or two Intel Xeon processors ranging from 2.13 GHz to 2.66 GHz, memory sizes ranging between 8 and 16 gigabytes and Western Digital hard drives ranging in size from 160 GB to 600 GB spinning at 7200 RPM. All nodes ran Debian 6.0 “Squeeze,” had Cloudera’s CDH 4.2.0 Hadoop distribution installed and used the

vendor-supplied patches for Apache Hive 0.10.0 and Apache Hadoop 2.0. We used Impala 1.0, as it was the most recent version available at the time of writing. PostgreSQL was tested using version 9.1.3 on a desktop computer running Windows 7 equipped with a hard drive with a rotational speed of 7200 RPM as well as a laptop computer running PostgreSQL 9.1.9 and Ubuntu 10.04.4 LTS equipped with a 120 GB Intel 330 series solid state drive.

The data was loaded directly from the set of CSV files in the case of PostgreSQL, generating a large table with 109 columns. For Apache Hive, the data was loaded from CSV file using CSV SerDe and put into a text table as well as a table encoded in record columnar format, also known as RCFile (He *et al.* (2011)). As Impala supports an efficient column-oriented file format called Parquet natively, we also copied the data from the text table into its preferred Parquet format. For VisReduce, data was imported from CSV and written as its native columnar format, each tablet being the columnar representation of an input CSV file.

To minimize the effect of disk caching, the operating system's read cache was flushed between runs of Apache Hive, Impala and VisReduce; this was done to ensure that the data for each query was loaded from disk. Neither Impala nor VisReduce keep data resident in memory, but both benefit from the operating system cache in the case where data from a previous run is re-read on the same node. As the PostgreSQL database size was larger than the available memory, this step was not deemed necessary for PostgreSQL. Queries on Impala and VisReduce were run 20 times each, five times for Apache Hive and PostgreSQL on SSD and three times for PostgreSQL on a standard hard drive. As is common for benchmarks on the JVM, queries for VisReduce were ran several times before the actual timing runs as a warm up phase as to avoid JIT compilation during benchmarking.

3.8 Results

The performance of VisReduce is significantly better than row-oriented databases, as shown in Figure 3.6. As column-oriented databases only transfer the columns required to answer a query, they have much higher effective I/O utilization than row-oriented approaches for large aggregation-oriented workloads on a subset of columns. Column compression further increases

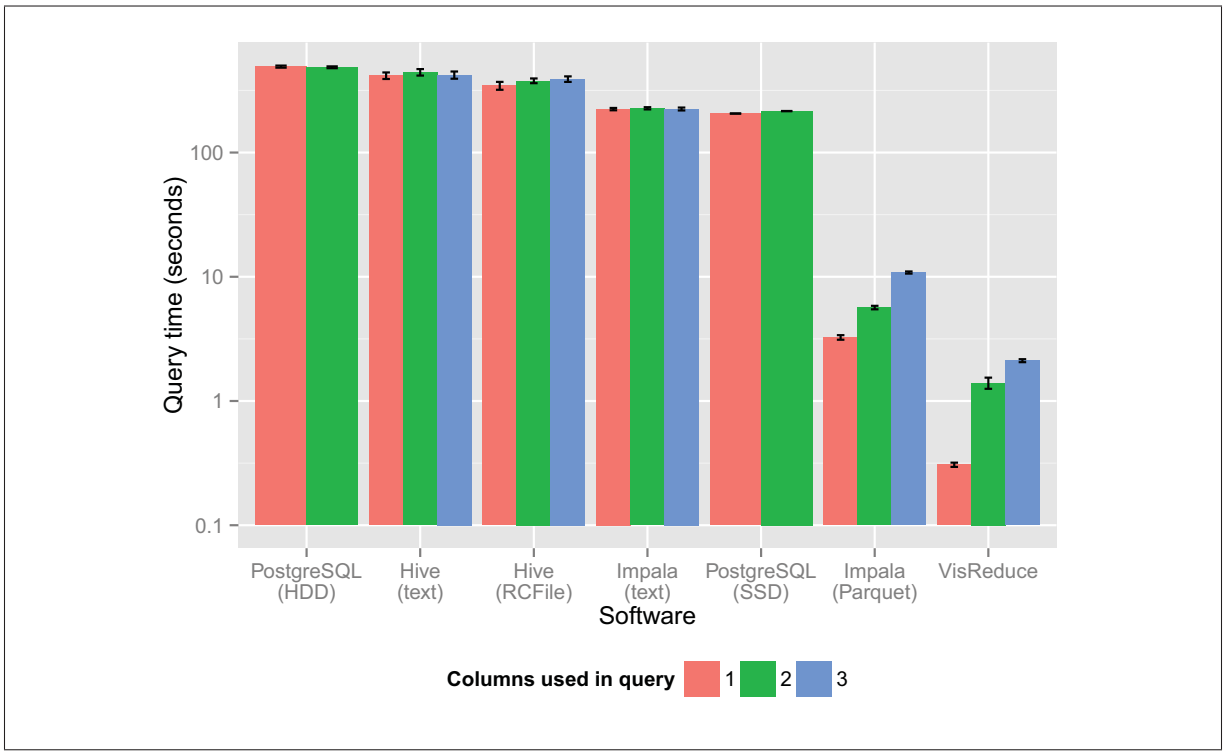

Figure 3.6 Log_{10} plot of mean query completion time by query type. Error bars represent a 95% confidence interval for the mean. Queries on three columns for PostgreSQL were aborted after twenty minutes.

the performance advantage in the case of columns containing highly redundant data, such as the carrier name column in our dataset, which has a very low cardinality compared to the number of records. Finally, as the data in VisReduce is static, there is no need for locks, row versions or other forms of concurrency control, as would be the case in a general purpose database where data can be written at any time, such as PostgreSQL.

In the case of Apache Hive, there is a significant per-job overhead due to the query being translated into a MapReduce job before being deployed onto the cluster. Once the job has been deployed, it requires several MapReduce iterations, with each iteration incurring fixed start up time costs and the need to write to disk between each iteration. While very general — for example, Hive can join arbitrarily sized tables, which neither Impala nor VisReduce can do — there is a significant performance cost to this generality, which makes Hive unsuitable for exploratory visual analytics if the data can be processed by faster systems. At the time of testing, Hive lacked stable support for the more efficient Parquet file format.

As for Cloudera Impala, it is possible that its query planning phase limited its performance relative to VisReduce; the current implementation of VisReduce has no query optimizer and naively processes all tablets across the cluster. Furthermore, Impala has a pluggable storage architecture and supports multiple input formats, while the current implementation of VisReduce only supports its own native column store format.

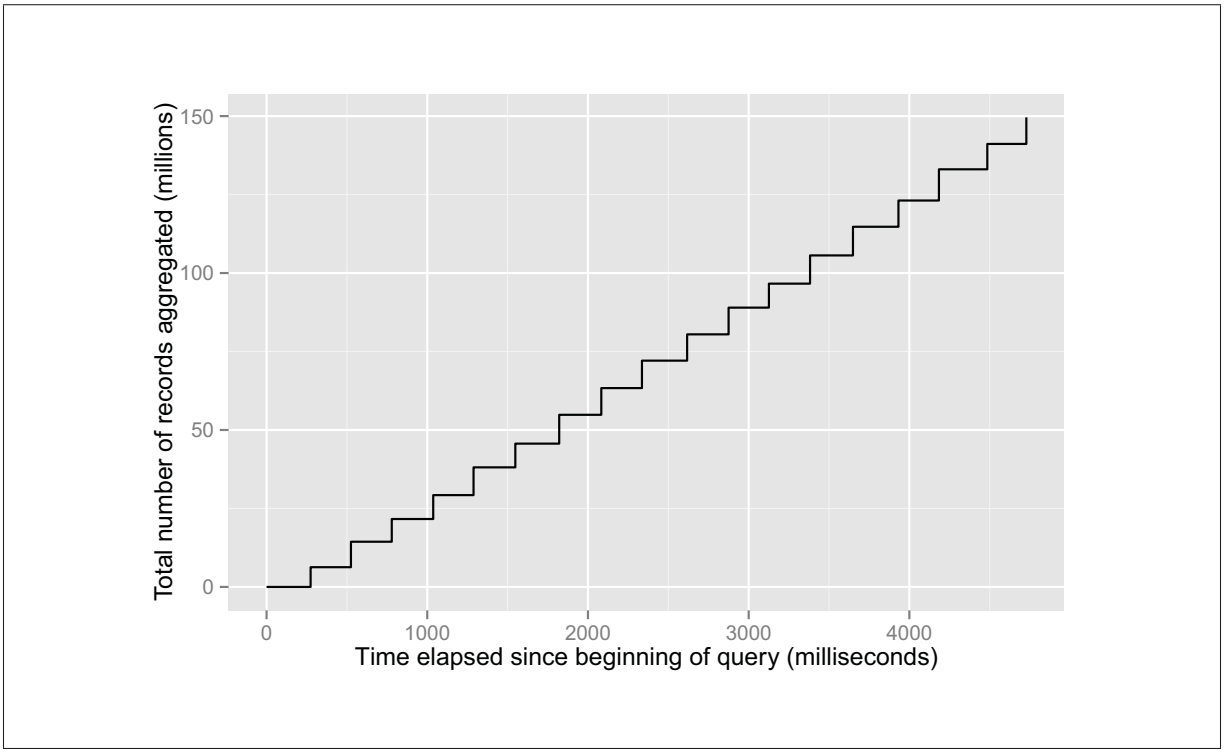

Figure 3.7 Incrementally updated output with VisReduce. The number of records aggregated and sent to the client since query start time using a cluster with a single worker node.

For the bar charts displayed by the client, our VisReduce implementation attempts to send JSON-formatted data every 250 milliseconds to the browser via WebSocket, which is then turned into an animated chart using JavaScript. Figure 3.7 shows the number of records that have been sent to the client for visualisation as a query progresses. We experimented a little with changing this parameter, but it seemed a reasonable compromise between perceived end-user latency and the capabilities of current web browsers to ingest data at a fast rate while animating many SVG elements without any perceived choppiness.

Increasing the rate at which data is sent to the browser also has another unfortunate tradeoff; as the data displayed to the user rapidly converges at the beginning of the computation, showing a visualisation right after the user clicks to start a job means that the user will see a quickly updating and "jumpy" visualisation. Our first iteration on VisReduce simply iterated through tablets in chronological order, which caused the bars in the resulting bar chart to rapidly shift for several seconds as new carriers that did not operate in the year 1987 were introduced into the bar chart and pushed the other bars around while the vertical axis changed its scale to accomodate the fluctuating bars. Shuffling the tablet processing order greatly reduced this shifting. We believe that pre-populating bars with cardinality information gathered from the column store metadata would make the resulting visualisation more aesthetically pleasing, as it would prevent new bars from being introduced.

3.9 Limitations

As we mostly focused on the technical aspects of computing the underlying data for an information visualisation in an incremental fashion, most of the human interface aspects were ignored. For example, adding error bars as the query processes more information seems like an obvious improvement, which has been explored by Fisher (2011); Fisher *et al.* (2012).

Another limitation is the fact that programming a VisReduce job is not as simple as writing a SQL query. The endurance of SQL as a query language shows how user friendly and useful it is to answer a wide range of queries. However, in many visualisation systems, such as Tableau (Stolte *et al.* (2002a)), SQL is merely an implementation detail that is hidden from the user. We believe that it would be possible to provide built-in VisReduce jobs that compute aggregates in an online fashion and offer a more familiar interface just as Apache Hive provides a SQL-like abstraction on top of MapReduce.

VisReduce is also not as general as MapReduce, which can handle arbitrarily sized outputs and enormous input data sizes that would simply be too large to visualize in an interactive fashion; this is by design. In VisReduce, we trade generality for performance and quick feedback.

VisReduce is simply not a good match for batch processing or processing of arbitrary data, just as common implementations of MapReduce are not a good match for interactive processing.

We also do not currently address algorithms that require multiple passes over the input data. For example, it is not possible to compute the standard deviation in a single pass as it requires knowing the mean of the input data, which is unknown at the start of a job. Algorithms that depend on a global ordering, such as computing the exact median of the data, are also impossible to express in a single pass.

3.10 Conclusion and Future Directions

VisReduce is a novel approach for interactive visualization of large data sets, that is scalable, distributed, achieves low latency, returns incremental feedback to the user multiple times per second, and was found to be significantly faster than three competing readily available solutions.

The main drawback with VisReduce is that it currently has no way of computing confidence bounds on the partial results it displays to the user over the course of a query. As mentioned in the previous section, adding estimation of error bounds of partial aggregates would be helpful for analysts to determine if they should stop a query or wait for its completion. Jermaine *et al.* (2006) and Joshi and Jermaine (2008) suggest ways of doing so on relational databases and, while VisReduce isn't a relational database, similar approaches could be used to provide online estimates of error.

An additional direction for future work would be to modify VisReduce to allow pre-loading all data in memory in the case of smaller data sets, as is done by Shenker *et al.* (2012). Further work is also needed to evaluate VisReduce with much larger data sets and cluster sizes to identify potential performance bottlenecks.

CONCLUSION

Visualization of databases by casual business users is still something that is not done often. We looked into solving that problem by developing the GPLOM, which allows automatic visualization of multidimensional multivariate data that is stored in databases and found that users prefer it and it allows exploring datasets faster than Tableau, a commercially available product.

Our contributions with GPLOM are (1) a novel interactive technique for data exploration of multidimensional multivariate datasets that is accessible to casual users; (2) a demonstration of textual search to find quickly values of interest in a composite plot matrix; (3) a novel linking technique called "bendy highlights" that links the various charts; (4) an improvement over Emerson *et al.* (2013)'s technique that allows adding interactive elements by only displaying half of the matrix; (5) an experimental comparison with Tableau, a popular commercial software product, that shows GPLOM being significantly faster in certain cases.

Still, further work is required on the GPLOM prototype in order to explore ways of making it friendlier and less intimidating than the current approach, such as presenting the matrix upside down. There are also performance issues that can happen with large data sets, due to the large number of charts that need to be computed at any given point in time.

In order to address these performance issues, we developed a prototype of an incremental visualization system called VisReduce. By using a modified MapReduce-style approach, it provides incremental visualization of datasets by sending partial aggregates to the client. We also compared its performance with several approaches and found that VisReduce has several performance advantages due to its usage of colunmar storage and simple programming model.

Our contributions with VisReduce are (1) a novel approach for interactive visualization of large datasets that is scalable, distributed, achieves low latency and returns incremental feedback to the user multiple times per second; (2) a comparison with three other readily available solutions — PostgreSQL, Apache Hive and Cloudera Impala — that shows VisReduce is faster for all queries that were tested.

Extending VisReduce to add error bounds for calculations is a logical next step for VisReduce, as well as packaging it with built-in operators that compute frequently done operations — such as the average, min, max, etc. — and exploring computing other types of visualisations. It would also be interesting to run user studies on how actual business users and data analysts perceive incremental visualisation is perceived to confirm if the incremental approach has benefits over non-incremental approaches.

Also, modifying the GPLOM prototype so that it uses VisReduce for incremental calculation seems like an obvious improvement. Unfortunately, current browser technology (circa 2013) does not allow for smooth interpolation of large numbers of SVG elements and would require GPLOM to be reimplemented as an application that uses GPU rendering for fast and smooth animations.

We believe that both of these techniques offer improvements over the current state of the art and, when combined together, plant the seeds for a new visualization tool that significantly outperforms currently available commercial visual analytics tools.

ANNEX I

DETAILS OF THE METHODOLOGY USED FOR THE GPLOM EVALUATION

1 Methodology

As mentioned in section 2.5, the experiment involved a mixed design with 12 participants:

2 types of questions
$\times$ 4 criteria counts (0 through 3)
$\times$ 2 technique-dataset pairs (GPLOM and Tableau 7.0)
$\times$ 12 users
= 192 trials

All participants were handed a pre-questionnaire and a post-questionnaire.

2 Statistical analysis

Three variables were analyzed using R (R Core Team (2013)): the time required by participants to formulate an answer using the visualization tool, their error rate as well as their subjective preferences.

2.1 Time to answer

The time to answer was analyzed using the nparLD R package by Noguchi *et al.* (2012). Because participants were not exposed to each possible combination of dataset and software, the data was separated by dataset for analysis; each participant used each dataset only once, with either Tableau or GPLOM.

Variables used in the analysis are *method* (either GPLOM or Tableau), *questionType* (either data value lookup or trend analysis) and *criteria* (number of criteria, zero to three, inclusive). The non-parametric ANOVA-type statistic (ATS) was used for analysis, due to the presence

of outliers, non-normality and heteroscedasticity between the various levels of the *criteria* variable, following the suggestions of Erceg-Hurn and Mirosevich (2008).

Table-A I-1 Analysis of variance for the time to answer

Cars				
	Statistic	df	p-value	sig
method	23.0686710	1.000000	1.563171e-06	****
questionType	0.5930081	1.000000	4.412583e-01	
criteria	12.8925133	2.593155	1.435815e-07	****
method × questionType	4.3133304	1.000000	3.781486e-02	*
criteria × questionType	3.3022601	2.037353	3.592508e-02	*
method × criteria	0.8958475	2.593155	4.303616e-01	
method × questionType × criteria	0.4142911	2.037353	6.645895e-01	
OnTime				
method	0.831071886	1.000000	3.619628e-01	
questionType	0.002258413	1.000000	9.620966e-01	
criteria	36.442713095	2.079264	4.152101e-17	****
method × questionType	2.409977165	1.000000	1.205641e-01	
criteria × questionType	3.287611907	2.330942	3.023219e-02	*
method × criteria	1.090446622	2.079264	3.378095e-01	
method × questionType × criteria	0.581364532	2.330942	5.846854e-01	

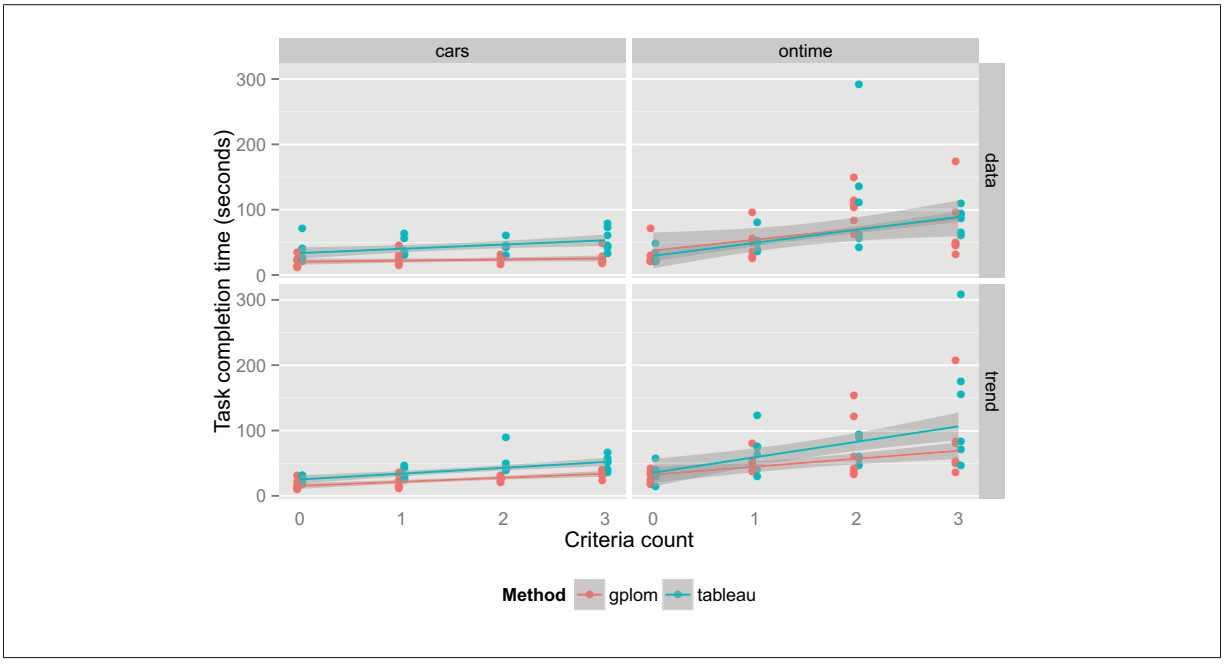

Figure 1.1 Faceted view of the time to answer

While the *method* variable was not significant for the OnTime dataset, the low number of participants for each method (six per method) and the larger variance for both techniques due to the higher complexity of the OnTime dataset[1] might hide a difference between Tableau and GPLOM. GPLOM also had a computational disadvantage compared to Tableau, due to the cost of generating the numerous charts over 494218 data points. It took several seconds to generate charts in GPLOM[2], due to each chart requiring a separate query to the database. Comparatively, it took roughly one second to do the same in Tableau. As each drill down requires a full page load and regenerating the charts in GPLOM, this meant that GPLOM had an approximate performance penalty of 12-15 seconds over Tableau in the case of a question with three criteria, assuming no errors being made by the participant in selecting filters (median performance of questions with three criteria on OnTime for GPLOM was 52 seconds). As mentioned in section 2.5.2, there was no correction for this factor.

Furthermore, one question asked of the participants was found to be unusually hard to answer in both GPLOM and Tableau. The question was "Pour quel groupe d'heures de départ est-ce que Delta (DL) a eu le plus de vols le jour de Noël (25)," translating to "For which departure time group did Delta (DL) have the most flights on Christmas (25)?" To answer the question, participants had to infer in GPLOM that the count aggregation for any given variable returns the number of flights. In Tableau, participants had to use the special variable called "Number of records."

Finally, one participant was found to consistently underperform by a wide margin on the On-Time dataset using GPLOM, as shown in figure 1.2. Such a pattern did not occur with Tableau, neither on the OnTime dataset nor the Cars dataset.

[1] As a rough comparison point, the dataset explanation sheet given to participants for OnTime took two pages, while the one for Cars did not fill a single page.

[2] An instrumented version of GPLOM later showed that the median time to reload the page and generate charts for the OnTime dataset was 5.663 seconds (min = 3.035 s, max = 7.962 s). This measurement was not taken during the experiment. For the much smaller Cars dataset, this median time was 1.695 seconds (min = 1.48 s, max = 3.146 s).

Correcting for all of these factors — removing questions with a criteria count of 2[3], removing participant 12 and applying a 3 second correction per criteria for GPLOM — makes the *method* variable attain weak significance level ($p = 0.0667$) and a study with a larger number of participants would be required to determine whether or not such an effect holds. Figures 1.3 and 1.4 show the ECDFs of the unadjusted data and the adjusted data, respectively.

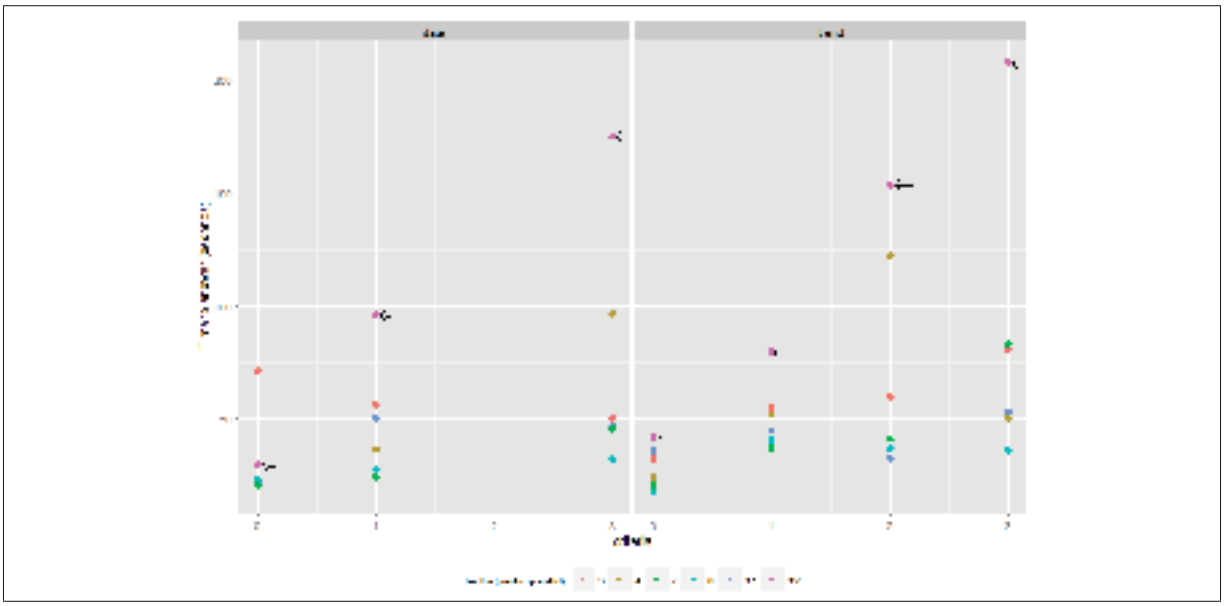

Figure 1.2 Performance of participant 12 on the OnTime dataset

[3]Both questions with two criteria were removed, as to keep the experiment balanced, otherwise the criteria count of 2 would not have both the *data* and *trend* levels.

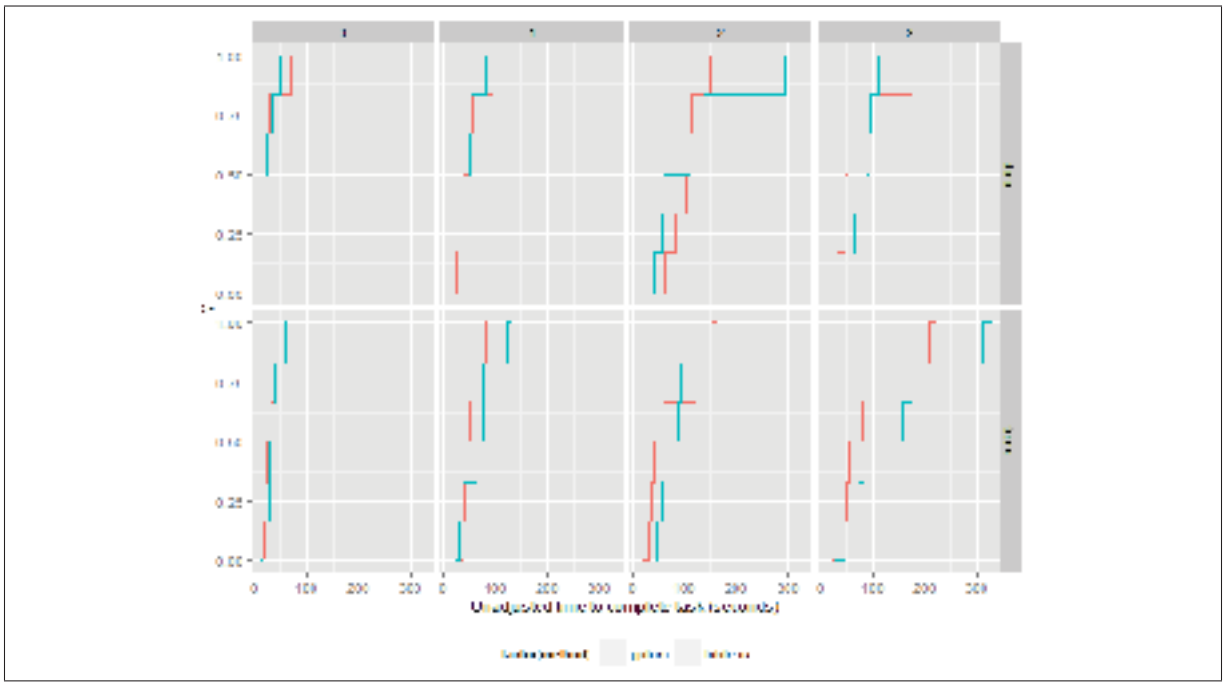

Figure 1.3 Unadjusted ECDF of participant time to answer for the OnTime dataset

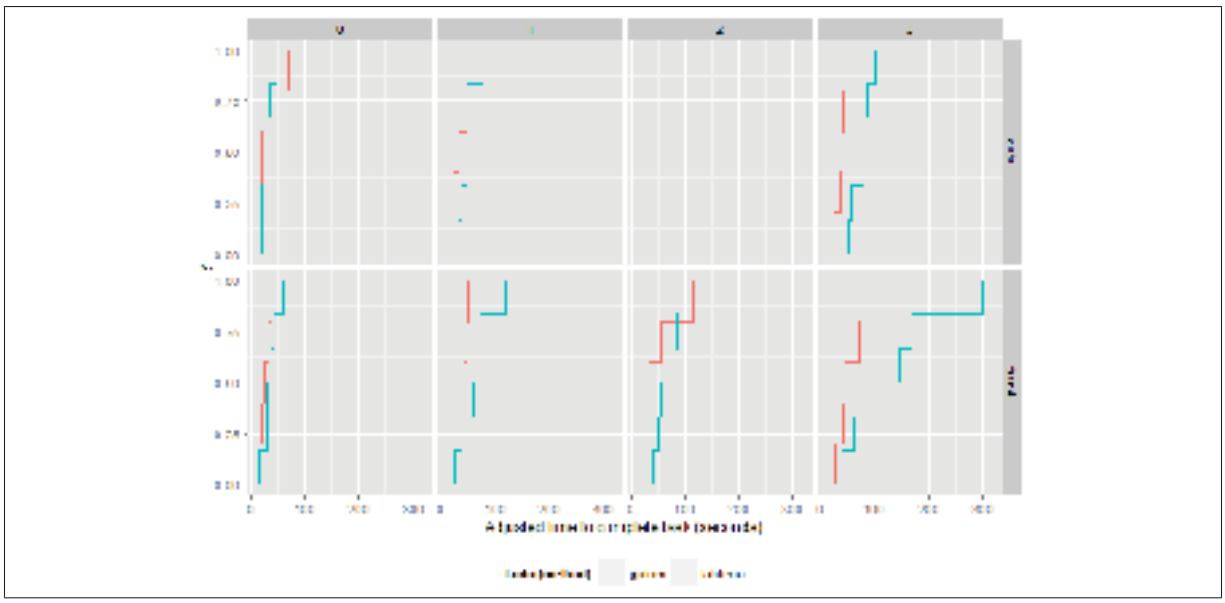

Figure 1.4 Adjusted ECDF of participant time to answer for the OnTime dataset

2.2 Error rate

Error rates were evaluated between GPLOM and Tableau using a logistic regression using R's `glm` with the binomial family. Table I-2 shows the resulting analysis of deviance table.

Interactions between variables were due to certain questions having higher error rates than others. GPLOM had a lower error rate than Tableau, but this was not significant.

Table-A I-2 Analysis of deviance table for the error rate

	Df	Deviance	Resid. Df	Resid. Dev	Pr(>Chi)	sig
NULL			191	182.34		
method	1	1.7217	190	180.62	0.1894755	
questionType	1	0.3176	189	180.30	0.5730409	
dataset	1	6.0891	188	174.21	0.0136016	*
criteria	1	0.5921	187	173.62	0.4416001	
method × questionType	1	0.2437	186	173.38	0.6215398	
method × dataset	1	1.9697	185	171.41	0.1604814	
questionType × dataset	1	0.7545	184	170.65	0.3850422	
method × criteria	1	12.5770	183	158.08	0.0003905	***
questionType × criteria	1	0.0357	182	158.04	0.8500568	
dataset × criteria	1	0.0228	181	158.02	0.8798699	
method × questionType × dataset	1	8.3081	180	149.71	0.0039468	**
method × questionType × criteria	1	0.0845	179	149.63	0.7713250	
method × dataset × criteria	1	4.1948	178	145.43	0.0405481	*
questionType × dataset × criteria	1	0.1442	177	145.29	0.7041066	
method × questionType × dataset × criteria	1	0.0000	176	145.29	0.9998463	

2.3 User preferences

In a post-questionnaire, users had to give a rating to both Tableau and GPLOM on a scale of 1 (not at all) to 5 (very) on nine different aspects.

User preferences between GPLOM and Tableau were evaluated by using a series of Wilcoxon signed rank tests. As the Wilcoxon signed rank test is unable to give an exact p-value in the case of ties, the exactRankTests R package by Hothorn and Hornik (2006) was used, which computes the exact p-value using the Shift-Algorithm.

To control the familywise error rate, the p-values were corrected using the Bonferroni correction.

Table-A I-3 User preferences between GPLOM and Tableau

Question	p	Corrected p	sig	Mean rating	
				GPLOM	Tableau
Is intuitive?	0.04492	0.40428		3.88	2.92
Easy to learn?	0.3906	1		3.92	3.50
Was able to do everything?	0.3594	1		4.42	4.08
Comfortable?	0.08594	0.77346		4.25	3.42
Easy to use?	0.1016	0.9144		4.33	3.58
Fast?	0.0009766	0.0087894	**	4.92	3.00
Satisfied?	0.125	1		4.08	3.50
Fluid interface?	0.003906	0.035154	*	4.50	3.00
Like?	0.08594	0.77346		4.08	3.42

BIBLIOGRAPHY

Abadi, Daniel, Samuel Madden, and Miguel Ferreira. 2006. "Integrating compression and execution in column-oriented database systems". In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*. p. 671–682. ACM.

Agarwal, Sameer, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. 2013. "BlinkDB: queries with bounded errors and bounded response times on very large data". In *Proceedings of the 8th ACM European Conference on Computer Systems*. p. 29–42. ACM.

Anscombe, Francis J. 1973. "Graphs in statistical analysis". *The American Statistician*, vol. 27, n° 1, p. 17–21.

Bederson, Benjamin B. and James D. Hollan. 1994. "Pad++: a zooming graphical interface for exploring alternate interface physics". In *Proceedings of the 7th annual ACM symposium on User interface software and technology*. (New York, NY, USA 1994), p. 17–26. ACM.

Bendix, Fabian, Robert Kosara, and Helwig Hauser. 2005. "Parallel sets: Visual analysis of categorical data". In *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on*. p. 133–140. IEEE.

Bertin, Jacques, 1967. *Sémiologie graphique: Les diagrammes, Les réseaux, Les cartes*. Paris : Éditions Gauthier-Villars.

Böse, Joos-Hendrik, Artur Andrzejak, and Mikael Högqvist. 2010. "Beyond online aggregation: parallel and incremental data mining with online Map-Reduce". In *Proceedings of the 2010 Workshop on Massive Data Analytics on the Cloud*. (New York, NY, USA 2010), p. 3:1–3:6. ACM.

Bostock, Michael, Vadim Ogievetsky, and Jeffrey Heer. 2011. "D^3 Data-Driven Documents". *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 17, n° 12, p. 2301–2309.

Carpendale, M. S. T. and Catherine Montagnese. 2001. "A framework for unifying presentation space". In *Proceedings of the 14th annual ACM symposium on User interface software and technology*. (New York, NY, USA 2001), p. 61–70. ACM.

Chang, Fay, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. June 2008. "Bigtable: A Distributed Storage System for Structured Data". *ACM Trans. Comput. Syst.*, vol. 26, n° 2, p. 4:1–4:26.

Chaudhuri, Surajit and Umeshwar Dayal. 1997. "An overview of data warehousing and OLAP technology". *ACM Sigmod record*, vol. 26, n° 1, p. 65–74.

Chaudhuri, Surajit, Umeshwar Dayal, and Vivek Narasayya. August 2011. "An overview of business intelligence technology". *Commun. ACM*, vol. 54, n° 8, p. 88–98.

Chen, C. july-aug. 2005. "Top 10 unsolved information visualization problems". *Computer Graphics and Applications, IEEE*, vol. 25, n° 4, p. 12 - 16.

Chernoff, Herman. June 1973. "The Use of Faces to Represent Points in K-Dimensional Space Graphically". *Journal of the American Statistical Association*, vol. 68, n° 342, p. 361–368.

Chi, Ed Huai-hsin, Phillip Barry, John T. Riedl, and Joseph A. Konstan. 1997. "A Spreadsheet Approach to Information Visualization". In *Proc. IEEE Symposium on Information Visualization (InfoVis)*. p. 17–24.

Claessen, Jarry H. T. and Jarke J. van Wijk. 2011. "Flexible Linked Axes for Multivariate Data Visualization". *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 17, n° 12, p. 2310–2316.

Collins, C. and S. Carpendale. nov.-dec. 2007. "VisLink: Revealing Relationships Amongst Visualizations". *Visualization and Computer Graphics, IEEE Transactions on*, vol. 13, n° 6, p. 1192 -1199.

Condie, Tyson, Neil Conway, Peter Alvaro, Joseph M Hellerstein, Khaled Elmeleegy, and Russell Sears. 2010. "MapReduce online". In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*. p. 21–21.

Dean, Jeffrey, Sanjay Ghemawat, and Google Inc. 2004. "MapReduce: simplified data processing on large clusters". In *In OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation*. USENIX Association.

Elmqvist, Niklas, Pierre Dragicevic, and Jean-Daniel Fekete. 2008. "Rolling the Dice: Multidimensional Visual Exploration using Scatterplot Matrix Navigation". *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 14, n° 6, p. 1141–1148.

Emerson, John W. and Walton A. Green. 2012. *gpairs: The Generalized Pairs Plot*. <http://CRAN.R-project.org/package=gpairs>. R package version 1.1.

Emerson, John W., Walton A. Green, Barret Schloerke, Jason Crowley, Dianne Cook, Heike Hofmann, and Hadley Wickham. 2013. "The Generalized Pairs Plot". *Journal of Computational and Graphical Statistics*, vol. 22, n° 1, p. 79–91.

Erceg-Hurn, D and V Mirosevich. 2008. "Modern robust statistical methods". *American Psychologist*, vol. 63, n° 7, p. 591–601.

Fisher, D. oct. 2011. "Incremental, approximate database queries and uncertainty for exploratory visualization". In *Large Data Analysis and Visualization (LDAV), 2011 IEEE Symposium on*. p. 73 -80.

Fisher, Danyel, Igor Igor Popov, Steven Drucker, and Monica Schraefel. May 2012. " Trust Me, I'm Partially Right: Incremental Visualization Lets Analysts Explore Large Datasets Faster ". *CHI 2012*.

Grammel, L., M. Tory, and M. Storey. nov.-dec. 2010. " How Information Visualization Novices Construct Visualizations ". *Visualization and Computer Graphics, IEEE Transactions on*, vol. 16, n° 6, p. 943 -952.

Grinstein, Georges, Marjan Trutschl, and Urška Cvek. 2001. " High-Dimensional Visualizations ". In *Proc. International Workshop on Visual Data Mining*. p. 7–19.

Hartigan, John A. 1975. " Printer Graphics for Clustering ". *Journal of Statistical Computation and Simulation*, vol. 4, n° 3, p. 187–213.

Hayashibara, Naohiro, Xavier Defago, Rami Yared, and Takuya Katayama. 2004. " The φ accrual failure detector ". In *Reliable Distributed Systems, 2004. Proceedings of the 23rd IEEE International Symposium on*. p. 66–78. IEEE.

He, Yongqiang, Rubao Lee, Yin Huai, Zheng Shao, Namit Jain, Xiaodong Zhang, and Zhiwei Xu. 2011. " RCFile: A fast and space-efficient data placement structure in MapReduce-based warehouse systems ". In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*. p. 1199–1208. IEEE.

Heer, J. and G.G. Robertson. nov.-dec. 2007. " Animated Transitions in Statistical Data Graphics ". *Visualization and Computer Graphics, IEEE Transactions on*, vol. 13, n° 6, p. 1240 -1247.

Heer, Jeffrey and Ben Shneiderman. February 2012. " Interactive Dynamics for Visual Analysis ". *Queue*, vol. 10, n° 2, p. 30:30–30:55.

Heer, Jeffrey, Jock D. Mackinlay, Chris Stolte, and Maneesh Agrawala. 2008. " Graphical Histories for Visualization: Supporting Analysis, Communication, and Evaluation ". *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 14, n° 6, p. 1189–1196.

Hellerstein, Joseph M., Peter J. Haas, and Helen J. Wang. 1997. " Online aggregation ". In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*. (New York, NY, USA 1997), p. 171–182. ACM.

Hewitt, Carl, Peter Bishop, and Richard Steiger. 1973. " A universal modular ACTOR formalism for artificial intelligence ". In *Proceedings of the 3rd international joint conference on Artificial intelligence*. (San Francisco, CA, USA 1973), p. 235–245. Morgan Kaufmann Publishers Inc.

Holten, Danny and Jarke J. van Wijk. 2010. " Evaluation of Cluster Identification Performance for Different PCP Variants ". In *Proceedings of Eurographics/IEEE-VGTC Symposium on Visualization (EuroVis)*.

Hothorn, Torsten and Kurt Hornik. 2006. “exactRankTests: exact distributions for rank and permutation tests”.

Igarashi, Takeo and John F. Hughes. 2001. “A suggestive interface for 3D drawing”. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*. (New York, NY, USA 2001), p. 173–181. ACM.

Im, Jean-François, Michael J. McGuffin, and Rock Leung. 2013a. “GPLOM: The Generalized Plot Matrix for Visualizing Multidimensional Multivariate Data”. *IEEE Transactions on Visualization and Computer Graphics (TVCG) (Proceedings of InfoVis)*, vol. 19, n° 12.

Im, Jean-François, Félix Giguère Villegas, and Michael J. McGuffin. 2013b. “VisReduce: Fast and responsive incremental information visualization of large datasets”. Proceedings of the IEEE Big Data Visualization Workshop 2013.

Inselberg, Alfred. 1985. “The Plane with Parallel Coordinates”. *Visual Computer*, vol. 1, p. 69–91.

Jermaine, Christopher, Alin Dobra, Subramanian Arumugam, Shantanu Joshi, and Abhijit Pol. 2006. “The Sort-Merge-Shrink Join”. *ACM Transactions on Database Systems (TODS)*, vol. 31, n° 4, p. 1382–1416.

Johnson, Chris. 2004. “Top scientific visualization research problems”. *IEEE Computer Graphics and Applications (CG&A)*, vol. 24.

Joshi, Shantanu and Christopher Jermaine. 2008. “Materialized sample views for database approximation”. *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, n° 3, p. 337–351.

Keim, Daniel A. 2002. “Information Visualization and Visual Data Mining”. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 8, n° 1, p. 1–8.

Kleiner, Beat and John A. Hartigan. June 1981. “Representing Points in Many Dimensions by Trees and Castles”. *Journal of the American Statistical Association*, vol. 76, n° 374, p. 260–269.

LeBlanc, Jeffrey, Matthew O. Ward, and Norman Wittels. 1990. “Exploring N-Dimensional Databases”. In *Proceedings of IEEE Visualization (VIS)*. p. 230–237.

Leinweber, David J. 2007. “Stupid data miner tricks: overfitting the S&P 500”. *The Journal of Investing*, vol. 16, n° 1, p. 15–22.

Leung, Y. K. and M. D. Apperley. June 1994. “A review and taxonomy of distortion-oriented presentation techniques”. *ACM Trans. Comput.-Hum. Interact.*, vol. 1, n° 2, p. 126–160.

Liu, Zhicheng, Biye Jiang, and Jeffrey Heer. 2013. “imMens: Real-time visual querying of big data”. In *Proceedings of EuroVis 2013*.

Mackinlay, J. D., P. Hanrahan, and C. Stolte. 2007. " Show Me: Automatic Presentation for Visual Analysis ". *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 13, n° 6, p. 1137–1144.

Mackinlay, Jock D. April 1986. " Automating the Design of Graphical Presentations of Relational Information ". *ACM Transactions on Graphics (TOG)*, vol. 5, n° 2, p. 110–141.

Melnik, Sergey, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, and Theo Vassilakis. September 2010. " Dremel: interactive analysis of web-scale datasets ". *Proc. VLDB Endow.*, vol. 3, n° 1-2, p. 330–339.

Mihalisin, T., J. Timlin, and J. Schwegler. 1991. " Visualization and Analysis of Multi-variate Data: A Technique for All Fields ". In *Proceedings of IEEE Visualization (VIS)*. p. 171–178.

Munzner, Tamara, François Guimbretière, Serdar Tasiran, Li Zhang, and Yunhong Zhou. 2003. " TreeJuxtaposer: scalable tree comparison using Focus+Context with guaranteed visibility ". *ACM Transactions on Graphics (TOG)*, vol. 22, n° 3, p. 453–462.

Noguchi, Kimihiro, Yulia R Gel, Edgar Brunner, and Frank Konietschke. 2012. " nparLD: An R Software Package for the Nonparametric Analysis of Longitudinal Data in Factorial Experiments ". *Journal of Statistical Software*, vol. 50, n° 12, p. 1–23.

North, Chris and Ben Shneiderman. 2000. " Snap-Together Visualization: A User Interface for Coordinating Visualizations via Relational Schemata ". In *Proceedings of Advanced Visual Interfaces (AVI)*. p. 128–135.

Pavlo, Andrew, Erik Paulson, Alexander Rasin, Daniel J Abadi, David J DeWitt, Samuel Madden, and Michael Stonebraker. 2009. " A comparison of approaches to large-scale data analysis ". In *Proceedings of the 35th SIGMOD international conference on Management of data*. p. 165–178. ACM.

Peng, Wei, Matthew O. Ward, and Elke A. Rundensteiner. 2004. " Clutter reduction in multidimensional data visualization using dimension reordering ". In *Proc. IEEE Symposium on Information Visualization (InfoVis)*. p. 89–96.

Pickett, Ronald M. and Georges G. Grinstein. 1988. " Iconographic Displays for Visualizing Multidimensional Data ". In *Proceedings of IEEE Conference on Systems, Man, and Cybernetics*. p. 514–519.

Pirolli, Peter and Ramana Rao. 1996. " Table lens as a tool for making sense of data ". In *Proceedings of the workshop on Advanced visual interfaces*. (New York, NY, USA 1996), p. 67–80. ACM.

Power, Daniel J. 2007. " A Brief History of Decision Support Systems ". http://dssresources.com/history/dsshistory.html.

Qu, Huamin, Wing-Yi Chan, Anbang Xu, Kai-Lun Chung, Kai-Hon Lau, and Ping Guo. 2007. “Visual Analysis of the Air Pollution Problem in Hong Kong”. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 13, n° 6, p. 1408–1415.

R Core Team. 2013. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. <http://www.R-project.org/>.

Rao, Ramana and Stuart K. Card. 1994. “The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information”. In *Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence*. (New York, NY, USA 1994), p. 318–322. ACM.

Rao, Ramana and Stuart K. Card. 1995. “Exploring large tables with the table lens”. In *Conference companion on Human factors in computing systems*. (New York, NY, USA 1995), p. 403–404. ACM.

Roberts, Jonathan C. 2007. “State of the Art: Coordinated & Multiple Views in Exploratory Visualization”. In *Coordinated and Multiple Views in Exploratory Visualization (CMV)*. p. 61–71.

Rowstron, Antony, Dushyanth Narayanan, Austin Donnelly, Greg O’Shea, and Andrew Douglas. 2012. “Nobody ever got fired for using Hadoop on a cluster”. In *Proceedings of the 1st International Workshop on Hot Topics in Cloud Data Processing*. p. 2. ACM.

Segaran, Toby and Jeff Hammerbacher, 2009. *Beautiful data: the stories behind elegant data solutions*. O’reilly.

Shenker, Scott, Ion Stoica, Matei Zaharia, Reynold Xin, Josh Rosen, and Michael J Franklin. 2012. “Shark: SQL and Rich Analytics at Scale”.

Shneiderman, Ben. 1994. “Dynamic queries for visual information seeking”. *Software, IEEE*, vol. 11, n° 6, p. 70–77.

Shneiderman, Ben and Martin Wattenberg. 2001. “Ordered treemap layouts”. In *Proceedings of the IEEE Symposium on Information Visualization 2001*.

Shrinivasan, Y.B. and J.J. van Wijk. sept.-oct. 2009. “Supporting Exploration Awareness in Information Visualization”. *Computer Graphics and Applications, IEEE*, vol. 29, n° 5, p. 34 -43.

Shrinivasan, Yedendra B. and Jarke J. van Wijk. 2008. “Supporting the Analytical Reasoning Process in Information Visualization”. In *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI)*. p. 1237–1246.

Spenke, Michael, Christian Beilken, and Thomas Berlage. 1996. “FOCUS: the interactive table for product comparison and selection”. In *Proceedings of ACM Symposium on User Interface Software and Technology*. p. 41–50.

Steed, Chad A., J. Edward Swan II, T. J. Jankun-Kelly, and Patrick J. Fitzpatrick. 2009. "Guided Analysis of Hurricane Trends Using Statistical Processes Integrated with Interactive Parallel Coordinates". In *Proc. IEEE Symposium on Visual Analytics Science and Technology (VAST)*. p. 19–26.

Steele, Julie and Noah Iliinsky, 2010. *Beautiful visualization*. O'Reilly Media, Inc.

Steinberger, Markus, Manuela Waldner, Marc Streit, Alexander Lex, and Dieter Schmalstieg. 2011. "Context-Preserving Visual Links". *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 17, n° 12, p. 2249–2258.

Stolte, C., D. Tang, and P. Hanrahan. april-june 2003. "Multiscale visualization using data cubes". *Visualization and Computer Graphics, IEEE Transactions on*, vol. 9, n° 2, p. 176 - 187.

Stolte, Chris, Diane Tang, and Pat Hanrahan. 2002a. "Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases". *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 8, n° 1, p. 52–65.

Stolte, Chris, Diane Tang, and Pat Hanrahan. 2002b. "Query, analysis, and visualization of hierarchically structured data using Polaris". In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. (New York, NY, USA 2002), p. 112–122. ACM.

Tenev, T. and R. Rao. oct. 1997. "Managing multiple focal levels in Table Lens". In *Information Visualization, 1997. Proceedings., IEEE Symposium on*. p. 59 -63.

Theus, Martin. 2003. "Interactive data visualization using mondrian". *Journal of Statistical Software*, vol. 7, n° 11, p. 1–9.

Thomas, James J and Kristin A Cook, 2005. *Illuminating the path: The research and development agenda for visual analytics*. IEEE Computer Society Press.

Tory, M. and T. Möller. 0-0 2004a. "Rethinking Visualization: A High-Level Taxonomy". In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*. p. 151 -158.

Tory, M. and T. Möller. jan.-feb. 2004b. "Human factors in visualization research". *Visualization and Computer Graphics, IEEE Transactions on*, vol. 10, n° 1, p. 72 -84.

van Wijk, Jarke J. and Robert van Liere. 1993. "HyperSlice: Visualization of scalar functions of many variables". In *Proceedings of IEEE Visualization (VIS)*. p. 119–125.

Viau, Christophe and Michael J. McGuffin. 2012. "ConnectedCharts: Explicit Visualization of Relationships between Data Graphics". *Computer Graphics Forum (Proceedings of EuroVis 2012)*, vol. 31, n° 3, p. 1285–1294.

Viau, Christophe, Michael J. McGuffin, Yves Chiricota, and Igor Jurisica. 2010. "The FlowVizMenu and Parallel Scatterplot Matrix: Hybrid Multidimensional Visualizations for Network Exploration". *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 16, n° 6, p. 1100–1108.

Vliegen, R., J.J. van Wijk, and E.-J. van der Linden. sept.-oct. 2006. "Visualizing Business Data with Generalized Treemaps". *Visualization and Computer Graphics, IEEE Transactions on*, vol. 12, n° 5, p. 789 -796.

Wang Baldonado, Michelle Q., Allison Woodruff, and Allan Kuchinsky. 2000. "Guidelines for Using Multiple Views in Information Visualization". In *Proceedings of Advanced Visual Interfaces (AVI)*. p. 110–119.

Ward, Matthew O. 2002. "A taxonomy of glyph placement strategies for multidimensional data visualization". *Information Visualization*, vol. 1, p. 194–210.

Ware, Colin. 2004. "Information visualization: Perception for design".

Wegman, Edward J. 1990. "Hyperdimensional Data Analysis Using Parallel Coordinates". *J. of the American Statistical Association*, vol. 85, n° 411, p. 664–675.

Wickham, H. and H. Hofmann. dec. 2011. "Product Plots". *Visualization and Computer Graphics, IEEE Transactions on*, vol. 17, n° 12, p. 2223 -2230.

Wickham, Hadley, 2009. *ggplot2: elegant graphics for data analysis*. Springer New York.

Wilkinson, Leland, Anushka Anand, and Robert Grossman. 2005. "Graph-Theoretic Scagnostics". In *Proc. IEEE Symposium on Information Visualization (InfoVis)*.

Wittenbrink, Craig M., Alex T. Pang, and Suresh K. Lodha. September 1996. "Glyphs for Visualizing Uncertainty in Vector Fields". *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 2, n° 3, p. 266–279.

Wong, Pak Chung and R. Daniel Bergeron. 1997. "30 Years of Multidimensional Multivariate Visualization". Chapter 1 (pp. 3–33) of Gregory M. Nielson, Hans Hagen, and Heinrich Müller, editors, Scientific Visualization: Overviews, Methodologies, and Techniques, IEEE Computer Society.

Yuan, Xiaoru, Peihong Guo, He Xiao, Hong Zhou, and Huamin Qu. 2009. "Scattering Points in Parallel Coordinates". *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 15, n° 6, p. 1001–1008.

Zaharia, Matei, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. 2010. "Spark: cluster computing with working sets". In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*. p. 10–10.

Zhang, Junliang and Gary Marchionini. 2004. "Coupling Browse and Search in Highly Interactive User Interfaces: A Study of the Relation Browser++". In *Proc. ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL)*. p. 384–384.