ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MANUSCRIPT-BASED THESIS PRESENTED TO
ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
Ph.D.

BY
Christophe VIAU

HYBRID VISUALIZATIONS FOR DATA EXPLORATION

MONTREAL, JULY 17, 2012

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS:

Mr. Michael J. McGuffin, Thesis director
Department of Software and IT Engineering, École de Technologie Supérieure

Mr. Jacques de Guise, Committee president
Department of Automated Production Engineering, École de Technologie Supérieure

Mr. Christopher M. Collins, External examiner
Department of Computer Science, University of Ontario Institute of Technology

Mr. Eric Paquette, 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 MAY 16, 2012

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGEMENTS

I must first thank my supervisor Michael McGuffin, who guided me in an area completely new for me to a useful and thorough knowledge of key concepts in Information visualization and Human computer interaction. His deep knowledge and broad vision of a multi-disciplinary field allowed me to target with great precision my research interests and the subject of my articles. His leadership has been exemplary and well beyond the usual responsibilities incumbent on a supervisor. Grants, internships, conference attendance, meeting with leading researchers, experience of article reviews, lectures and especially the involvement in the writing of articles and in the preparation of conferences are a few examples of his generosity and sense of responsibility.

These articles could not have been possible without valuable collaborations with my supervisor but also with other teachers, students and researchers from different fields. I thank the collaborators who have greatly helped to develop the articles forming a large part of this thesis. The fruitful discussions with Yves Chiricota, the great generosity of Igor Jurisica and his team, working closely and intensely with Guy Melançon in the exploration of theoretical models are examples of combining a rigorous commitment to a surprising creativity.

The continued involvement of SAP, its financial support and the quality of guidance of its team during internships have greatly helped shape the subject of my thesis. Special thanks go to Michael McAllister and Rock Leung for managing the SAP Academic Research Center (ARC) Fellowship, and Ed Prinsen, Christophe Favart, Huma Zaidi, John Armitage and several other key players on the internships side.

I also thank the jury members, who helped me bring this thesis to the desired level of quality, and the team of the École de technologie supérieure for their support.

I have saved my most important thanks for last. My family was always by my side with good advices and support. Above all, I sincerely thank Ennio and Alexandra, without whom my life would have no color.

HYBRID VISUALIZATIONS FOR DATA EXPLORATION

Christophe VIAU

ABSTRACT

Information Visualization (Infovis) graphically encodes information to help a user explore a data set visually and interactively. This graphical encoding can take the form of widespread visualizations such as bar charts and scatterplots. Multiple visualizations can share the same functional space to form complete tools for visual exploration or for communicating information. There is multiple ways of combining these visualizations. The assembly of multiple visualizations can give some complex assemblies sometimes called hybrid visualizations.

A hybrid visualization is the result of assembling multiple simpler visualizations. For example, NodeTrix (Henry *et al.*, 2007a) is composed of a node-link diagram and an adjacency matrix, and MatLink (Henry and Fekete, 2007a) adds arc links to an adjacency matrix. This integration of multiple visualizations can be a way to combine their advantages into a coherent structure. The integration can be achieved, for example, through color coding, or through explicit linking (such as with arrows), or through interaction (such as when different visualizations respond to the manipulation of others). Recent literature contains several examples of new hybrid visualizations, most often to deal with complex datasets where the user can benefit from multiple, complementary visual encodings of the same data. However, to date, there is almost no theory or framework to help researchers understand and characterize existing hybrids or design new ones.

This thesis advances the state of the art in hybrid visualizations in two ways: first, by developing a framework that defines and characterizes hybrid visualizations to help better identify, describe and design them, and second, by demonstrating a variety of novel hybrids. The hybrid visualizations we explored cover a wide range of possibilities. Two of the most general and widely used data types in Infovis, multidimensional multivariate data and graph (i.e., network) data, are each the subject of a chapter in the thesis, with novel hybrid visualization techniques presented for each. A wide range of possibilities for integration is also presented using a pipeline model.

After some preliminary material, chapter 2 of the thesis presents a conceptual framework that defines and characterizes hybrid visualizations. This framework was itself derived from experience designing the hybrid visualizations presented in the subsequent chapters. A hybrid visualization is described as a graphical encoding using other visualizations as building blocks. We present a pipeline to illustrate the assembly of a visualization, starting from the generation of basic shapes or glyphs, then placed on a layout, embellished by adding other graphical elements, then sent to some view transform operators and assembled on the same space. Simple charts can be described with this pipeline as well as more complex assembly and new hybrids are described.

Chapter 3 presents ConnectedCharts, an example of a hybrid assembled on the assembly level of the pipeline, made of multiple multidimensional and multivariate charts explicitly connected by lines or curves showing the relationship between their elements. A user interface enables the interactive assembly of ConnectedCharts, including a wide range of previously-published hybrid visualizations, as well as novel hybrid arrangements. ConnectedCharts serve as an illustration of the conceptual framework in chapter 2, by exploring possible connections between different graphics depending on the relationship of their encoded data types.

Chapter 4 presents another user interface, this time for graph exploration, that incorporates several highly integrated hybrid visualizations. A Parallel Scatter Plot Matrix (P-SPLOM) is presented that constitutes a fusion of a Scatter Plot Matrix (SPLOM) and a Parallel Coordinates Plot (PCP). A radial menu called the FlowVizMenu enables the modification of a visualization integrated at the center of the menu. This menu is also used to select the dimensions for configuring a third hybrid based on an Attribute-Driven Layout (ADL) that combines a node-link diagram and a scatterplot.

The characterization of hybrid visualizations offered by the conceptual framework, as well as the illustration of the framework by innovative hybrid visualizations, are the main contributions of this thesis to the Infovis community.

Keywords: information visualization, hybrid visualization, Infovis, graph

VISUALISATIONS HYBRIDES POUR L'EXPLORATION DE DONNÉES

Christophe VIAU

RÉSUMÉ

L'Infovis encode graphiquement de l'information pour aider un utilisateur à explorer visuellement et interactivement un ensemble de données. Cet encodage graphique peut prendre la forme de visualisations largement répandues, comme les diagrammes à barres et les diagrammes à nuage de points. Plusieurs visualisations peuvent partager le même espace fonctionnel pour former des outils complets servant à explorer un ensemble de données ou à communiquer de l'information. Il existe de nombreuses façons de combiner ces visualisations, pouvant donner des assemblages complexes parfois appelées visualisations hybrides.

Une visualisation hybride est le résultat de l'assemblage de plusieurs types de visualisations. Par exemple, NodeTrix (Henry *et al.*, 2007a) intègre un diagramme noeuds-liens (node-link diagram) et une matrice d'adjacence (adjacency matrix) et MatLink (Henry and Fekete, 2007a) ajoute des liens en arcs à une matrice. Cet intégration de visualisations peut être une façon d'allier les avantages de plusieurs visualisations différentes dans un ensemble cohérent et solidement lié. Cette liaison peut se faire par des codes de couleur, par des liens explicites comme des flèches, par une interaction coordonnées où les visualisations réagissent à la manipulation d'une autre. Nous avons étudiés ces différentes stratégies d'assemblage de deux façons: en proposant une bonne variété de visualisations hybrides et en élaborant une définition et une caractérisation des visualisations hybrides permettant de mieux les identifier, les décrire et les concevoir. Les prototypes d'hybrides que nous décrivons couvrent une large gamme de possibilités. Les données les plus utilisées en Infovis, soit les données multidimensionnelles multivariées et les données de graphes sont représentées, chacunes faisant l'objet d'un chapitre entier. Une large palette de possibilités d'intégration est aussi représentée, du simple code de couleur partagé entre deux graphiques jusqu'à la fusion de visualisations coordonnées par la couleur, les formes, l'animation et l'interactivité.

Nous présentons en premier lieu une définition et une caractérisation des visualisations hybrides découlant des expérimentations des chapitres suivants. Une visualisation hybride est décrite comme un encodage graphique utilisant d'autres visualisations comme matériau servant à l'assemblage. Nous présentons une sorte de diagramme d'assemblage que nous appelons un pipeline pour illustrer l'assemblage de visualisations, à partir de la génération de formes de base ou de glyphes, qui sont ensuite disposés selon un patron, embellis par l'ajout d'autres éléments graphiques, puis transformé pour être assemblé dans la même vue. Des graphiques simples peuvent être décrits par ce pipeline d'assemblage de visualisations, ainsi que des combinaisons plus complexes et de nouveaux hybrides.

Le chapitre suivant donne un bon exemple d'intégration de bas niveau. Les ConnectedCharts sont des graphiques multidimensionnels et multivariés explicitement connectés par des lignes montrant bien la relation entre leurs éléments. Un prototype permet d'assembler toutes sortes

de visualisation hybrides trouvées dans la littérature ainsi que de nouveaux agencements. Il sert d'illustration à un cadre conceptuel explorant les connections possibles entre différents graphiques selon la parenté de leur types de données encodées.

Une autre interface, cette fois pour la visualisations de données de graphes, présente plusieurs visualisations hybrides fortement intégrées. Le P-SPLOM provient de la fusion d'une matrice de diagrammes à nuage de points (SPLOM) et d'une série de diagrammes à coordonnées parallèles. Un menu radial nommé FlowVizMenu permet de faire varier une visualisation qui y est intégrée. Ce menu sert à choisir les dimensions à utiliser pour une troisième visualisation hybride: le Attribute-Driven Layout fusionnant un diagramme noeuds-liens à un diagramme à nuage de points.

La définition et la caractérisation des visualisations hybrides ainsi que l'illustration de ce cadre théorique par la conception d'hybrides novateurs sont les contributions principales de cette thèse au domaine de l'Infovis.

Mots-clés: visualisation de l'information, visualization hybride, Infovis, graphe

CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Page

GLOSSARY

adjacency matrix Graph visualization showing each nodes on the rows and columns of a matrix with the cells indicating the presence or absence of a link between the nodes on this specific row and column. . 10

automated presentation system Computer program optimizing the display of presentation using a set of rules for graphical encoding and some evaluation criteria. 9, 17

glyph A symbol, such as a stylized figure or arrow on a public sign, that imparts information nonverbally. 16, 18, 28, 85

graphical encoding Translating data to graphics by mapping data variables to graphical attributes. 5, 6, 85

hybrid Composite; formed or composed of heterogeneous elements. VII

hybrid visualization A visualization resulting from the assembly of other visualizations. VII, 2, 11, 16, 68, 85, 89

multidimensional multivariate A multidimensional dataset refers to the dimensionality of the independent variables and multivariate of the dependent variables. VII, 2, 39, 89

node-link diagram Visualization typically representing a graph data structures by encoding data elements to a shape (a node) and the relations between elements by lines (links). 10, 12, 60, 61, 67, 89, 93

pipeline Diagram illustrating a process as a flow going through pipes. 16–18, 85

LIST OF ABBREVIATIONS

ADL Attribute-Driven Layout. VIII

ARGOI Attribute Relation Graph of Interest (Claessen and van Wijk, 2011). 54

BI Business Intelligence. 11

DA Dimensional Anchor (Hoffman *et al.*, 1999). 87

FLINA Flexible Linked Axis (Claessen and van Wijk, 2011). 86

HCI Human Computer Interaction. 64

Infovis Information Visualization. VII, IX, X, 1, 5

P-SPLOM Parallel Scatter Plot Matrix. VIII

PCP Parallel Coordinates Plot. VIII, 11, 30, 61

Scivis Scientific Visualization. 5

SPLOM Scatter Plot Matrix. VIII, 61

SPPC Scattering Points in Parallel Coordinates (Yuan *et al.*, 2009). 32

INTRODUCTION

Infovis

Infovis has been described as: “the use of computer-supported, interactive, visual representations of abstract data to amplify cognition”(Card *et al.*, 1999). Finding the right representation makes it easier to visually extract useful information from abstract data, like numbers and percentages. A trend in the data is easier to detect by seeing a line going up than by reading a series of numbers. Some statistics can be used to detect and quantify some characteristics of this trend without using visualization. But the line in a line chart also shows some details, some changes of slope, the shape of the whole curve, some peaks, all at once, without losing the relationship between the different parts and the whole. The human perceptual system is excellent at detecting patterns, trends, visual groupings, and anomalies in patterns (Thomas and Cook, 2005).

Visualization helps “make sense” of data. It gives a shape to abstract data to assist in “solving problems”(Purchase *et al.*, 2008). Exploring data in a visual way helps to turn a series of numbers into actionable insights. Visualization is also used to communicate these insights.

Choosing the right visualization for effective exploration and communication can be difficult. Some tasks or some datasets need a combination of multiple visualizations, or even require to design custom ones. For example, a complex dataset in aerial traffic routing, can be composed of geographic coordinates, a network of routes between airports, planes velocity, weather, etc. In this case, we could use multiple visualizations, like a map for watching the traffic in real-time, a network to resolve transit issues, a line chart to monitor the temperature, etc. But if these visual tools have to be used in coordination, they must be visually integrated, for example being on the same screen. Some visualization can also be integrated further. For example a network showing the trajectory between airports can be overlaid on a map to place these airports in a geographical context, the color of the map can represent the temperature, etc. Multiple uncoordinated views can thus be *combined* in a single visualization built from mostly the same elements.

We will call hybrid visualizations the result of combining multiple preexisting visualizations, for example overlaying a network on a map, drawing a histogram at the axes of a scatterplot or nesting some bar charts into a table. Some guidelines can help in choosing the right visualization for some specific tasks (Few, 2004). But combining visualization is a design task harder to generalize and needs a better conceptual framework for the design of new hybrid visualizations. This thesis addresses this situation by proposing a more systematic framework for thinking about hybrids (chapter 2) and by exploring new hybrid visualizations to demonstrate its value (chapters 3 and 4).

Scope of this research

One of the goals of this thesis is to cover two large categories of datasets: *multidimensional multivariate* data, and *graph* (or network) data. An example of a multivariate multidimensional dataset might consist of measurable quantities such as price, profit and weight of various products in various stores. In this example, "product" and "store" can be thought of as *independent variables* and the measurable characteristics like price and weight could be called *dependent variables*. For each given product, for example, we may want to know their price or to calculate some statistics on their price over time, so price depends on product. As do Wong and Bergeron (1997), "we adopt the convention that the term "multidimensional" refers to the dimensionality of the independent variables, while the term "multivariate" refers to the dimensionality of the dependent variables. An example of multidimensional multivariate visualization is described in chapter 3.

Another widely used data structure in Infovis is the *graph*, also called a network. The graph encodes a set of elements and the relationship between these elements, such as a social network with links between individuals of a community or a biological network including possible interactions between proteins. This data structure is typically represented by a node-link diagram, with nodes represented by shapes and links by lines joining these shapes. Hybrid visualizations of graphs are the topic of chapter 4.

By studying hybrid visualizations of these two common data structures, both our theoretical and applied exploration covers a significant portion of the opportunities for innovation in Infovis.

The contributions of this thesis are mainly about designing innovative hybrid visualizations, supported by a conceptual framework offering a definition and some criteria for the characterization of hybrid visualizations.

Thesis organization

This thesis is organized as follows. First, chapter 1 gives an overview of literature to establish additional context and present some of the more important models and significant achievements in Infovis. Then, chapters 2 through 4 present research that is based, respectively, on three papers. The first of these chapters called "Characterizing Hybrid Visualizations" develops a model of hybrid visualizations to help describe and design them. Next, the chapter "ConnectedCharts: A hybrid Visualization by Explicit Linking" presents a user interface connecting different graphics, exploring the possibilities of explicit connections, drawing lines between graphics and between related items in multiple graphics. Next, chapter 4, called "FlowViz-Menu, P-SPLOM and AD-Layout: Hybrid Visualizations for Network Exploration", presents an application of new hybrid visualizations to the field of interactive graph exploration. The content of chapter 3 was published in (Viau and McGuffin, 2012), and of chapter 4 in (Viau *et al.*, 2010), and chapter 2 has been submitted for publication. A discussion and a conclusion then summarize and help bind the content of these three papers.

The research behind the three papers were performed in the reverse order of their presentation in this thesis. Chronologically, graph visualization (motivated by a specific application in bioinformatics) was studied (chapter 4); next, a more generic application motivated the work in chapter 3; and finally a model was developed (chapter 2). So the hybrid visualization model is the result of the previous design experience, thinking about the needs and the possibilities of hybrids while working on it and afterwards. But introducing the model first in this thesis

will go from general to specific, defining further the context of creation of specialized hybrid visualizations.

CHAPTER 1

LITERATURE SURVEY

1.1 Important models in Infovis

A survey of some "must read" books will help define the field of information visualization.

The Infovis field can't be easily summarized by a few seminal books. It is a mixture of multiple research fields, like statistics, human-computer interaction, psychology of perception, graphic design and semiology. But some trends can be identified. For example, we can survey some important contributions to graphical encoding, design and semiology as well has placing Infovis in a historical context.

The history of visualization does not have precise limits. Tufte's books (Tufte, 1983) give a great overview of the history of graphics. The definition of the field itself is often imprecise. We could say that scientific visualization takes care of spatial data, data visualization is more data transformation oriented, visual data mining is more algorithm oriented, information visualization more on presenting high level results and infographics more about information design. But there is no clear agreement on any of these terms. Infovis is associated with statistical graphics as visual tools to make sense of data. Their is a certain consensus on describing Infovis as using non-spatial or abstract data as opposed to Scientific Visualization (Scivis) visualizing spatial data. For example, Voigt (2002) defines it as the "visualization of abstract data. This is data that has no inherent mapping to space." For Tory and Moller (2004), it "involves abstract, nonspatial data", and for Maria Cristina Ferreira de Oliveira (2003), "in information visualization, the graphical models may represent abstract concepts and relationships that do not necessarily have a counterpart in the physical world." So we will define Infovis as the study of visualization resulting from the graphical encoding of abstract data.

A visualization follows a set of graphical encoding rules for mapping data to graphics and organizing the different elements. We will explain what is a graphical encoding throughout this thesis. Let's describe for now Infovis as a research field exploring the graphical encoding

of data, using a visual language, suitable for a task, with the general goals to extract and communicate information.

1.2 Graphical encoding

We can see a visualization as formed by three fundamental ingredients: data, graphical encoding and interaction. This view conforms with Munzner's model for the design of a visualization (Munzner, 2009), and also with Keim's classification of information visualization and visual data mining techniques (Keim, 2002). We can see the same elements as a dynamic process in a chain of operations starting from the data, through visualization operations with the goal of satisfying a user task with a need for interaction, perception and cognition. One good model for this chain of operations from data to user is Chi and Riedl (1998) operator interaction framework. Another inspiration, closely related to Chi and Riedl's model, is from (Card and Mackinlay, 1997): data types, function for recoding data, recoded data, visual mapping, view transformation and manipulation widgets.

Infovis visually encodes abstract data in an arbitrary way, using a language consisting of signs, "syntactically notational", forming the "sentences of a graphic language", according to Ziemkiewicz and Kosara (2009). Wilkinson is another influential author on visual semiology and his book "The Grammar of Graphics" (Wilkinson, 2010) is the groundwork for a lot of visualization libraries. like Data-Driven Documents (Bostock *et al.*, 2011), Protovis (Bostock and Heer, 2009) and ggplot2 (Wickham, 2009). Wilkinson (2010) developed a complete formalism to describe visualizations as a visual language with semantic rules. We will not get into the "meaning" of a visualization but only into the assembly process starting from the data to get to a suitable shape.

We call this process of translating data to graphics "graphical encoding". According to Gee *et al.* (2005), "Information visualizations attempt to efficiently map data variables onto visual dimensions in order to create graphic representations". So graphical encoding is at the core of Infovis research. Every visualization uses graphical elements with data mapped to visual attributes as the building block of a visual language. The process often follows Bertin's "Semi-

ology of Graphics" (Bertin, 1998) where a graphical element, called a mark, can vary according to different attributes, such as position, size and color, to encode information.

Bertin's terminology of marks and attributes makes an important distinction between "attributes of the plane" and "retinal attributes". Marks are graphical elements, the building blocks of visualizations, like bars, wedges, axes, etc. The attributes of the plane are the x and y components of the position of a mark. Other attributes, like color, orientation, size and texture, are retinal attributes that can be immediately perceived and used to visually group, compare, relate and do other perceptual tasks. For Bertin, the position is a privileged attribute and it must be used to encode the most important aspect of the data. Additional data dimensions can be encoded by retinal attributes as a complement.

Wickham (2009), author of ggplot2, an important R package to develop visualizations, describes this mapping from data to graphics in two steps. First the data variables are associated to their graphical attributes, which he call aesthetics. Then the values are mapped into aesthetic space. For example, the variable Product price can be associate with the x position of the dots of a scatterplot. Then the price values are scaled to map to the pixel space of the scatterplot drawn on a screen. The author uses a model inspired from Wilkinson, who uses the same kind of process to map data to graphics. Engelhardt (2002) uses the terms elementary graphic object and visual attributes. These different terminologies illustrates the same concepts of marks and attributes derived from Bertin.

Graphical encoding can be formalized by an arbitrary set of rules. We can describe a visualization by listing a set of graphical elements, the way they are assembled and the data they encode. For example we can describe a simple bar chart using this arbitrary notation:

- Data (data)
  - Categorical (c)
  - Quantitative (q)
- Graphical elements:
  - Rectangles (bar)

  - Axis length (l)
- Graphical attributes:
  - Position: x, y
  - Size: weight (w), height (h)
- Subset of rules for mapping data to graphics:
  - Quantitative is mapped on bar's height proportionally to data max and to chart height

    bar.h = data.q / data.q.max * axisY.length
  - Juxtaposed bars have equal width sized to fit in the chart width

    bar.w = axisX.length / data.c.size
  - Bars are uniformly distributed on the x axis

    bar.x = data.c.index / data.c.size * axisX.length
  - Bars are aligned to the x axis

    bar.y = axisX - bar.h

This minimal description is enough for a visualization engineer to implement this simple bar chart in a programming language. For example, using D3.js (Bostock *et al.*, 2011), a library inspired by Wilkinson's Grammar of graphics (Wilkinson, 2010) and successor of Protovis (Bostock and Heer, 2009), this bar chart could be described by the same simple mapping to x, y, width and height on as many rectangles as there are values in the dataset.

```
var data = d3.range(10).map(Math.random);

var w = 400,
  h = 200,
  barW = w / data.length,
  barH = function(d, i){return d / d3.max(data) * h;},
  barX = function(d, i){return i / data.length * w;},
  barY = function(d, i){return h - barH(d);};

d3.select("body")
  .append("svg")
  .attr("width", w)
  .attr("height", h)
  .selectAll("rect")
```

```
.data(data)
.enter().append("rect")
.attr("width", barW)
.attr("height", barH)
.attr("x", barX)
.attr("y", barY);
```

1.3 From data to graphics

Graphical encoding is the process of translating data to graphics. To translate data in a meaningful way, the resulting shape must efficiently represent the characteristics of the data.

Many authors, like APT (Mackinlay, 1986), SAGE (Roth and Mattis, 1990), IDES (Goldstein *et al.*, 1994), Polaris (Stolte *et al.*, 2002) and Boz (Casner, 1989), studied data characterization for automating the presentation of graphics, and find some rules for automatically choose the right mapping from data to graphics. One important characteristic of the data is its data type. The term data type can be confusing. (Shneiderman, 1996) uses data type in a broad sense to use as categories. By data type, he means 1-2-3 dimensional data, temporal and multidimensional data and tree and network data. Other authors use the term "data type" with a meaning derived from Stevens (Stevens, 1946) scale types: "nominal", "ordinal", "interval" and "ratio". For example, Zhang (1996) and Bertin take their categories from Stevens, while automated presentation system mainly use an equivalent terminology, like "categorical" and "quantitative". Roth and Mattis (1990) add spatial and temporal coordinates to these data types for a better characterization to suit the rules for automatic presentation. In this thesis, we, will borrow the simple characteristics from APT and SAGE and use the type "categorical" for data elements that can be listed, counted, sometimes sorted (i.e.: apple, orange, banana), and "quantitative" for data elements representing a value, a measure (i.e.: 1, 2, 3).

For Zhang (1996) and Dastani (2002) as well as for the automated presentation systems previously cited, there is a correspondence, or some affinities, between some data types and some graphical attributes. Zhang illustrates this by describing a correspondence between data types and and graphical attributes. For example, a shape can only encode a nominal value because it

can't systematically be ordered in a natural way or measured relatively to another shape wherever a price can be represented by a height because comparing heights and comparing prices are both possible. So Zhang, as well as the automated systems cited above, all define a table of correspondence between data types and some specific visualizations. We will suggest our own table derived from these concepts in the chapter 3.

Data characteristics help to choose the right type of visualization for encoding a dataset efficiently. But combining multiple visualizations is more than simply choosing the right chart. It is more about understanding how a visualization is built and extend this knowledge to the assembly of multiple visualizations.

Recent literature contains several examples of new hybrid visualizations. However, to date there is almost no theory or framework about hybrids to help researchers understand them or design new ones. The work on graphical encoding, for example on automated presentation, has proven effective (Mackinlay *et al.*, 2007), but is of a limited use to describe or generate new combinations of visualizations. Casner (1989) clearly states that: "it is unlikely that BOZ will produce new graphic designs that differ radically from existing designs". APT provides only one strategy to assemble two visualizations: "compose two designs by merging parts that encode the same information" (Mackinlay, 1986). A more flexible model is needed to explore the design space of hybrid visualizations.

1.4 Hybrid visualizations

Many graphics, like bar charts (Fig. 1.1 bottom left), scatterplots (Fig. 1.1 top left), parallel coordinates plot (Fig. 1.1 top right), node-link diagrams ((Fig. 1.1 bottom middle)) and adjacency matrices ((Fig. 1.1 bottom right)), are widely used in Infovis and known for their effective encoding of certain data types for certain tasks. But a dataset is sometimes best represented by multiple different charts. For example, in Business Intelligence (BI), a dashboard with information on the performance of a business can display line graphs to show the trend in profits over time, bar charts to compare the costs of manufacturing different products, and scatterplots to show the correlation between advertising, expenditures and the amount of sales.

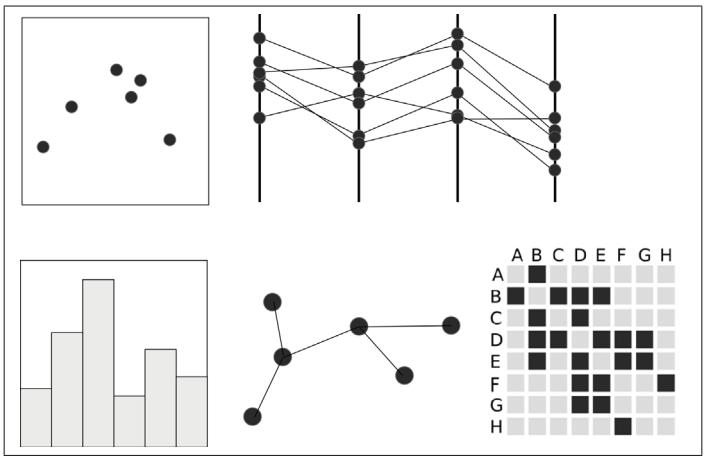


Figure 1.1 Different encoding patterns: scatterplot, parallel coordinates plot, bar chart, node-link diagram, adjacency matrix.

The same dataset can be presented by several types of diagrams to support and cater to different tasks (trend detection, comparison, correlation).

So different part of the same dataset can be encoded by different visualizations, choosing the most suitable for the task and the most consistent with the data characteristics. The same dataset can be displayed using two different graphical encodings (i.e., a pie chart and a line chart for sales price), two graphics can show two different parts of the same dataset (i.e., a bar chart for sales prices and a map for clients location), two different levels of information (i.e., a bar chart showing sales price for each quarter and another one for each weeks), or even two related dataset (i.e., bar chart for sales price and a network for relationship between the clients).

Some visualizations are more suitable than others for certain data characteristics and tasks. A visualization designer may want to combine the advantages of multiple types of visualizations. But their his many different ways of assembling multiple visualizations. One solution is to juxtapose the visualizations on the same space. A problem with juxtaposing charts is the difficulty to make sense of related information across multiple graphical structures. Coordination between views, like brushing and linking, can help visually and conceptually group related elements. But the attention is split between views. Another solution is to bring the different views closer by fusing the views into one. We will explore, particularly in chapter 2, some

strategies to combine visualizations. But we will describe some of them here to give examples of existing hybrids found in the literature and help define what is a hybrid visualization.

A visualization is the result of a graphical encoding. For example, distributing points on a line, like mapping some data as the position of dots on an axis, gives a visualization. A hybrid visualizations is the result of graphical encoding using other visualizations as building blocks. The axis described above can be combined with other axis encoding other dimensions of the data. The corresponding dots on each axis can be linked by a line, giving a basic PCP. So we can see a PCP as formed by smaller visualization units. And we can describe multiple hybrids formed by combining PCPs with other types of visualization.

1.5 Description of existing hybrid visualizations

Visualizations can be combined in many different ways. Some combinations can be loose, for example placing two visualizations in the same space. But it can also be more tight, like the fusion of two charts in a new single coherent visualization. From loose to tight assembly, there is a whole spectrum of possibilities.

A scatterplot and a bar chart can be placed on the same screen. The layout, the relative positioning of these charts, like grouping, alignment and scaling, helps to emphasize a relationship between them. It is a passive strategy acting only on geometric attributes (i.e., size, position, orientation) of the visualization as a whole, not on individual parts. Providing linking by color-coding or with explicit links like arrows, Dynamically updating these attributes, the color highlighting or the arrows, is a way to *integrate* them more tightly, for one to feel more like a part of the other. The bar chart could also be nested into the scatterplot or stitched, connected at its border. The elements of each visualization are thus more tightly connected. The tightest assembly uses sub-components of visualizations as building blocks. Each graphics are dissected into smaller graphical components then reassembled to form a single new visualization sharing some characteristics from both.

A graph (also called a network) could be visualized using an arc diagram or an adjacency matrix. These two views are complementary, each having their own advantages for certain

tasks. One possibility to combine their advantages would be to display them side-by-side, as in MatrixExplorer (Henry and Fekete, 2006), and to coordinate selection across the two views so that selecting in one will highlight corresponding elements in the other. However, we can go even further than this. For example, MatLink (Henry and Fekete, 2007a) stitches the arc diagram to the border of the adjacency matrix (See Figures 1.2 A)).

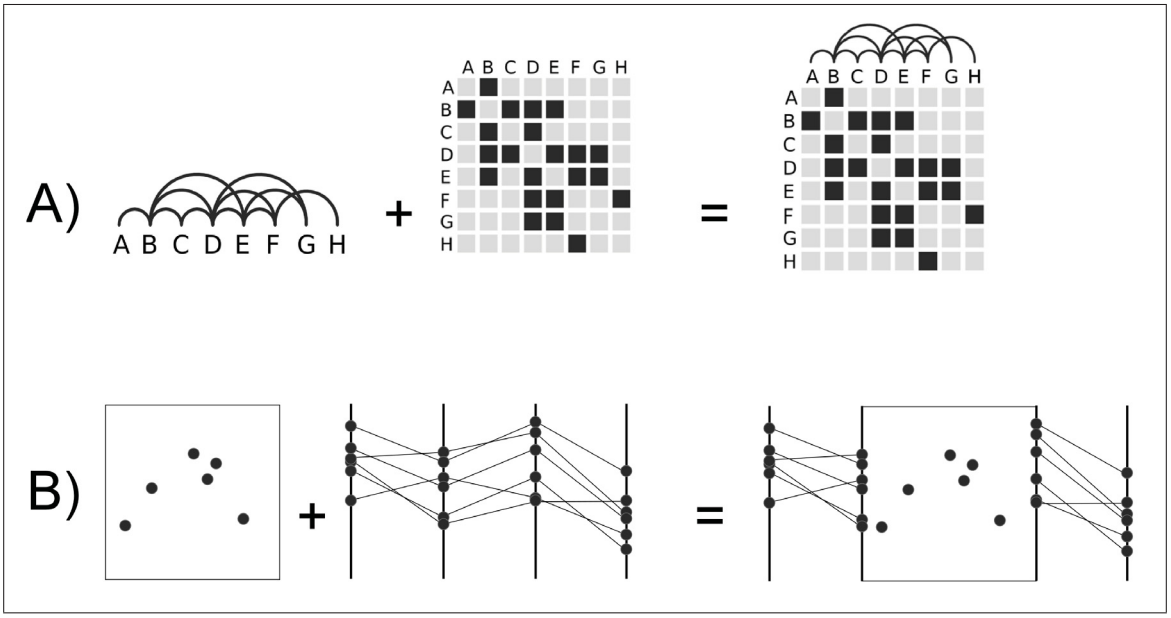


Figure 1.2 Two examples of hybrid visualizations. In A), an arc diagram is integrated with an adjacency matrix as in MatLink (Henry and Fekete, 2007a). In B), a scatterplot is integrated with a PCP, as in SPPC (Yuan *et al.*, 2009) or in FLINA (Claessen and van Wijk, 2011)

Other examples of graph hybrids integrate a node-link diagram to an adjacency matrix or to a scatterplot. NodeTrix (Henry *et al.*, 2007a) transforms some selected nodes to a matrix while preserving links between those matrices, as in a node-link diagram clustered as matrices. Elastic hierarchies (Zhao *et al.*, 2005) allows to collapse any branch of a tree to a matrix in a hierarchical way and to transform back any cell of the matrix to a branch. MatrixZoom (Abello and van Ham, 2004) display the link of a hierarchical tree on the edge of an adjacency matrix. GraphDice (Bezerianos *et al.*, 2010a) and the Attribute-Driven Layout of (Viau *et al.*, 2010) fuse the node of a node-link diagram on the corresponding dot of a scatterplot.

Many hybrids based on a PCP have been proposed in the literature. PCPs are well known in the Infovis community interested in multidimensional visualizations, in great part due to the extensive studies of Inselberg (1985). Vertical axes are arranged in a parallel sequence representing each dimension of the dataset. Each member of the dataset is represented by a polyline joining each of these axes at a height dependent on the value of this member to this dimension. Many PCPs can be juxtaposed, like in the PCP matrix of (Albuquerque *et al.*, 2009) and of (Heinrich *et al.*, 2012). We refer to this one as an example of a non-hybrid, being the assembly of visualizations of the *same* type. The PCP can be juxtaposed to other type of graphics to form a basic type of hybrid, like in (Chung and Zhuo, 2008) where a graph is used to manipulate the graphical attributes of a PCP, or in (Holten and van Wijk, 2010) and (Steed *et al.*, 2009) where scatterplots are added to each axis. The last one also add a histogram to each axis. Collins *et al.* (2009) are presenting a combination of a PCP and a tag cloud. Yuan *et al.* (2009) integrates more tightly a PCP to a scatterplot and (Viau *et al.*, 2010) shows a transition from a PCP to a scatterplot matrix (Claessen and van Wijk, 2011).

Most of these examples of hybrid visualizations are no older than four years. Maybe this trend is too young to have motivated the community to think of a model or even just define hybrid visualizations. Two very recent papers were published when this thesis was nearly finalized. Product Plots (Wickham and Hofmann, 2011) (the first author being the creator of ggplot (Wickham, 2009)), explores some interesting hybrids by combination of multiple graphical encodings. Another even more recent paper, Exploring the Design Space of Composite Visualization (Javed and Elmqvist, 2012), uses an approach similar to our chapter 2, describing some assemblies in similar terms, like juxtaposition, superimposition, overloading and nesting, which illustrates that the Infovis community is in need of better models to understand hybrid visualizations. But the papers on the subject are still rare and very few designers justify their hybrid designs. This is the main motivation behind the work that we present in this thesis.

CHAPTER 2

CHARACTERIZING HYBRID VISUALIZATIONS

Christophe Viau[1], Guy Melançon[2], Michael J. McGuffin[1],
[1]Department of Software and IT Engineering, École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3
[2]Department of Mathematics and Computer Science, Bordeaux I University
351 Cours de la libération, 33405 Talence Cedex, France
Paper submitted to Eurovis2012 in December 2011.

2.1 Abstract

Previous literature has proposed many combinations of visualizations, such as focus and context views, multiple coordinated views, small multiples, and various "hybrid" visualizations that combine features of simpler visualizations. To better understand the ways that visualizations can be combined, we analyze different ways that a dataset can be fed into multiple visualization pipelines, where each pipeline generates a different view or simple visualization. We also analyze how the output of these multiple pipelines can then be combined into a single, integrated visualization, possibly yielding a hybrid visualization. We also review previous examples of hybrid visualization, showing how each of them fit into our framework. Our work formalizes the meaning of "hybrid visualization" and clarifies issues around them, creating opportunities to identify new kinds of hybrids that are more clearly situated with respect to previous work.

2.2 Introduction

The use of multiple, side-by-side views onto a dataset (Wang Baldonado *et al.*, 2000; Roberts, 2007), showing different aspects of the data and often linked through coordination (North and Shneiderman, 2000a), has become a standard approach for designers and researchers. Such multiple views are one way of combining several individual visualizations. The past several

years of research in Infovis have also seen a growing number of more exotic combinations of visualizations that are sometimes called "hybrids" (Zhao *et al.*, 2005; Henry and Fekete, 2007b; Henry *et al.*, 2007a; Yuan *et al.*, 2009).

With a growing number of examples of hybrids, however, it becomes increasingly important to answer the questions: "What is a hybrid, or combination, of visualizations?", and "What kinds of hybrids, or combinations, are possible and may have not yet been discovered?" We offer answers to these questions by distinguishing between 6 different kinds of combinations of visualizations, each of which can be modeled with a variant of the familiar visualization pipeline of operators. The 6 are: side-by-side assembly, overlay assembly, heterogeneous visualizations, nested visualizations, hybrid layouts, and hybrid glyphs. For each kind of combination, we discuss how the pipeline is modified with respect to a simple visualization, and also present some examples of such combinations (including some novel visualizations), and discuss issues surrounding each kind of combination. Finally, we also present examples of new combinations or hybrid visualizations that were directly inspired by taking two simple visualizations and proceeding down the list of possible kinds of combinations, as though through a "checklist", thinking about what each kind of combination would imply.

By formalizing the notion of hybrid visualization, we are able to classify and contrast previous work, and to think more systematically about hybrid visualizations. We believe this will sometimes help designers and researchers to discover new visualizations, as we demonstrate with the novel hybrid visualizations presented later in this paper.

Our contributions are (1) the identification of 6 distinct kinds of combinations of visualizations, (2) examples of how to model these by modifying the standard visualization pipeline, (3) examples of new hybrid visualizations that were inspired by thinking about each kind of combination that might be possible.

2.3 Background

A visualization can be seen as a dynamic process, as a chain of operations starting from the data and passing through operators with the goal of satisfiying a user task. Two good models

for this chain of operations are Chi and Riedl's operator interaction framework (Chi and Riedl, 1998) and the closely related model of Card and Mackinlay (Card and Mackinlay, 1997).

The characterization of data and of graphical encoding were effectively formalized by research on the automatic presentation of visualization and on visual grammar. Many automated presentation systems such as APT (Mackinlay, 1986), SAGE (Roth and Mattis, 1990), IDES (Goldstein *et al.*, 1994), Polaris (Stolte *et al.*, 2002) and Boz (Casner, 1989) are based on an assembly algebra and on a rigid formalism based on rules aiming for the optimal presentation of visualizations. This type of automation has proven effective (Mackinlay *et al.*, 2007),

Designing a good visualization requires a good knowledge of the data characteristics, of the possible graphical encodings and of the user task. Composing multiple visualizations is no different, but at a higher level of graphical encoding. For example, a list of numbers can be encoded by the height of multiple rectangles; these rectangles can be aligned and distributed to form a simple barchart, and these barcharts can be stacked to form small multiples, etc. Graphical encoding has been extensively studied by Wilkinson (Wilkinson, 2010), Bertin (Bertin, 1967), and others (Card *et al.*, 1999) (Ware, 2004) (Dastani, 2002). But the design of hybrid visualizations and how to compose them from existing ones is still a territory to explore.

2.4 Visualization Pipeline

Different versions of the visualization pipeline have been presented in previous literature (Chi and Riedl, 1998) (Card *et al.*, 1999, chapter 1) (Carpendale, 1999, chapter 1). These have largely presented the pipeline as a linear process, with data flowing into a single sequence of transformations, yielding a single visualization. One way we can model the combination of multiple visualizations, such as in a hybrid, is by having the pipeline split apart into two or more branches, and later merge before the final output. We have therefore developed our own variant of a set of pipeline operations that allow us to model both simple visualizations as well as combinations with branching. Figures 2.1 and 2.2 illustrate two simple cases.

Figure 2.1 shows a dataset of four tuples serving as input. The first operator generates glyphs, whose radius depends on the 3rd column of the data, and whose color depends on the 4th

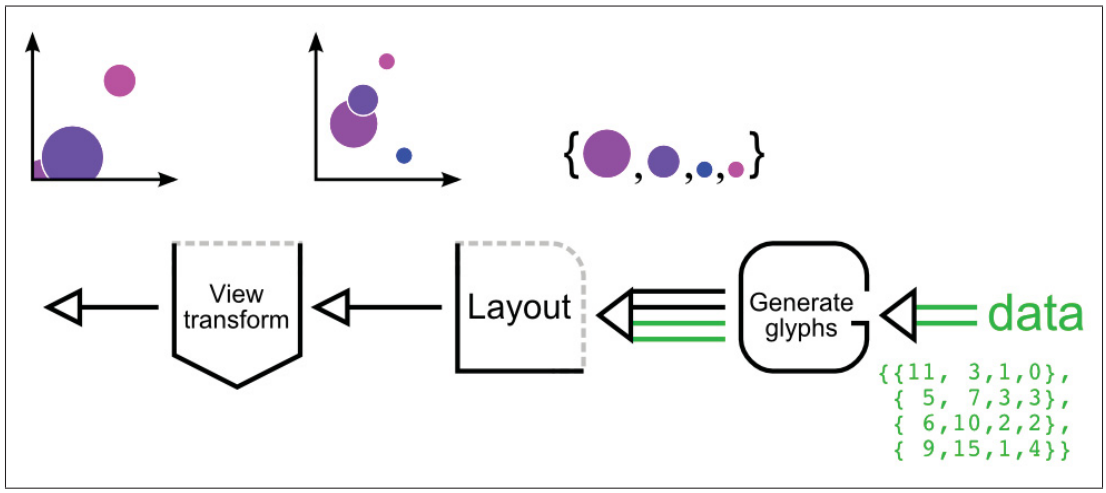


Figure 2.1 A pipeline modeling the mapping of multidimensional data to a scatterplot.

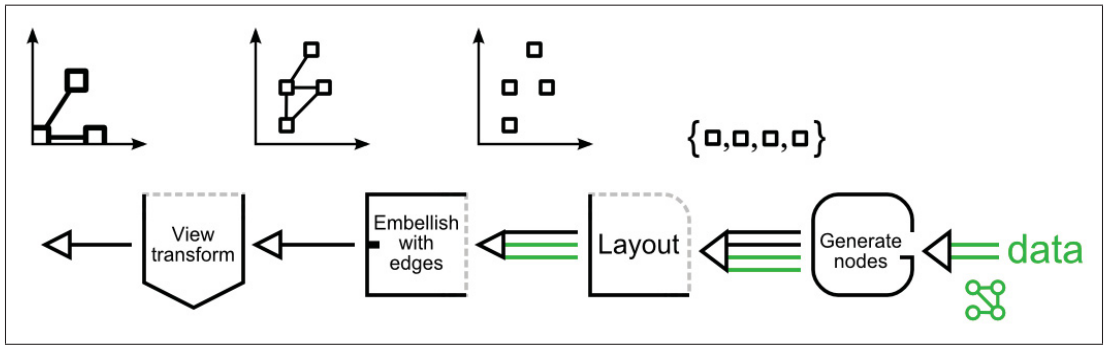


Figure 2.2 A network dataset visualized as a node-link diagram.

column. Next, a layout operator positions the glyphs, using the 1st and 2nd columns for the x and y coordinates, respectively. Finally, a view transform operator scales and translates the view seen by the user.

Figure 2.2 shows a graph (or network) as input data, transformed into a set of node glyphs, which are then positioned in 2D and scaled and translated.

In our pipeline, two kinds of information may flow along the "pipes" connecting operators: data that is stored in memory, and geometric objects. Data may be lists, multidimensional data, graphs (i.e., networks), etc. Geometric objects may be glyphs, or "graphics" (larger objects composed of glyphs) which might be nested within other geometric objects. Data is always shown in our figures using green ink, whereas geometric objects are shown in black or with

other colors. The line segments connecting operators show the kind of information passed from one to the next: two green line segments for data, two black line segments for multiple geometric objects (such as glyphs), and a single black line segment for a single geometric object (such as a graphic). For example, in Figures 2.1 and 2.2, we start with only a dataset at the right end of the pipeline (shown with two green line segments, since the data are plural). Once glyphs have been generated, two black line segments are added, to show that both geometric objects and data flow to the next stage. The input to the view transform operator, however, is only a single geometric object (a single black line segment), i.e. a graphic, because at that point the original dataset is no longer needed.

Our pipelines are constructed using 6 kinds of operators (Figure 2.3).

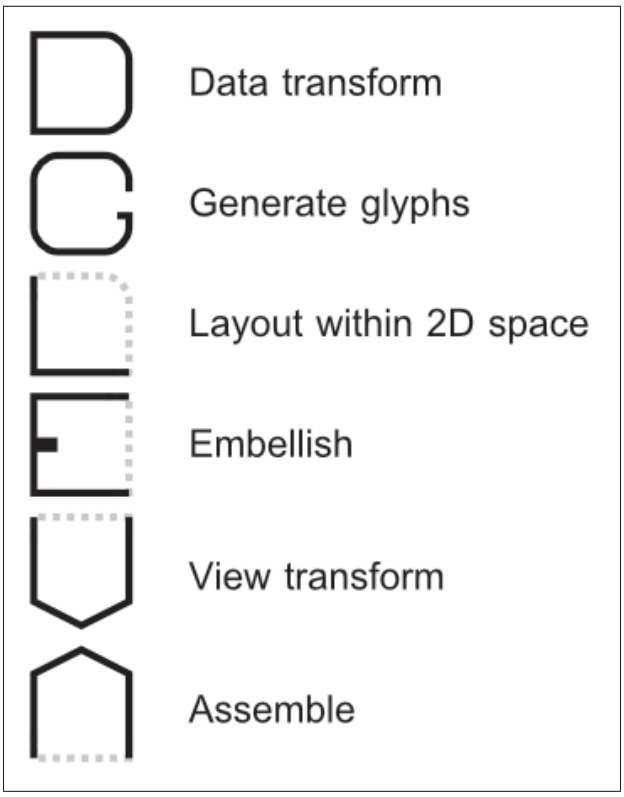


Figure 2.3 Legend listing the 6 kinds of operators used in the pipeline.

Data operators serve to transform, split, or merge streams of data. Glyph generation operators create geometric objects as a function of input data. Layout operators embed geometric objects by positioning them within a 2-dimensional space, yielding a graphic, i.e., a composite

geometric object. Embellish operators are used to add "decorations" to a geometric object. Embellish operators can add line segments or curves to connect the glyphs within a graphic, or add closed curves to indicate subsets of glyphs. View transform operators serve to translate and scale a geometric object. Finally, Assemble operators can combine multiple geometric objects (i.e., graphics) to create side-by-side views, for example.

At a minimum, a visualization pipeline must use at least one Glyph generator operator, one Layout operator, and one View transform operator. Additional operators of any type are possible, as will be seen in the subsequent examples.

We now present several ways to combine visualizations, roughly ordered from simplest to most complex, showing how each can be modeled using the pipeline. Note that the way we model various combinations and hybrids is not always unique; there are sometimes alternative ways of describing a combination that give the same output. However, the methods we present are *useful* perspectives to take for understanding and explaining a given combination, and the pipelines modeling them help to be more precise about the meaning of each kind of combination.

2.5 Methods for Combining Visualizations

2.5.1 Geometric Assembly

2.5.1.1 Side-by-Side Assembly

The simplest way to combine two visualizations is to display them side-by-side. This requires that there be two pipeline streams, either originating at different datasets, or because a single pipeline was somehow split into two streams. Figure 2.4 shows an example. Such side-by-side views may correspond to small multiples (Tufte, 1983), multiple coordinated views (Wang Baldonado *et al.*, 2000; Roberts, 2007), or focus-and-context views (Carpendale, 1999, chapter 2). Figure 2.4, for example, shows a world map beside a zoomed view of Africa.

Notice that there are two View transform operators in Figure 2.4. In side-by-side assemblies, these two operators may be independent, or may be connected somehow. For example, North

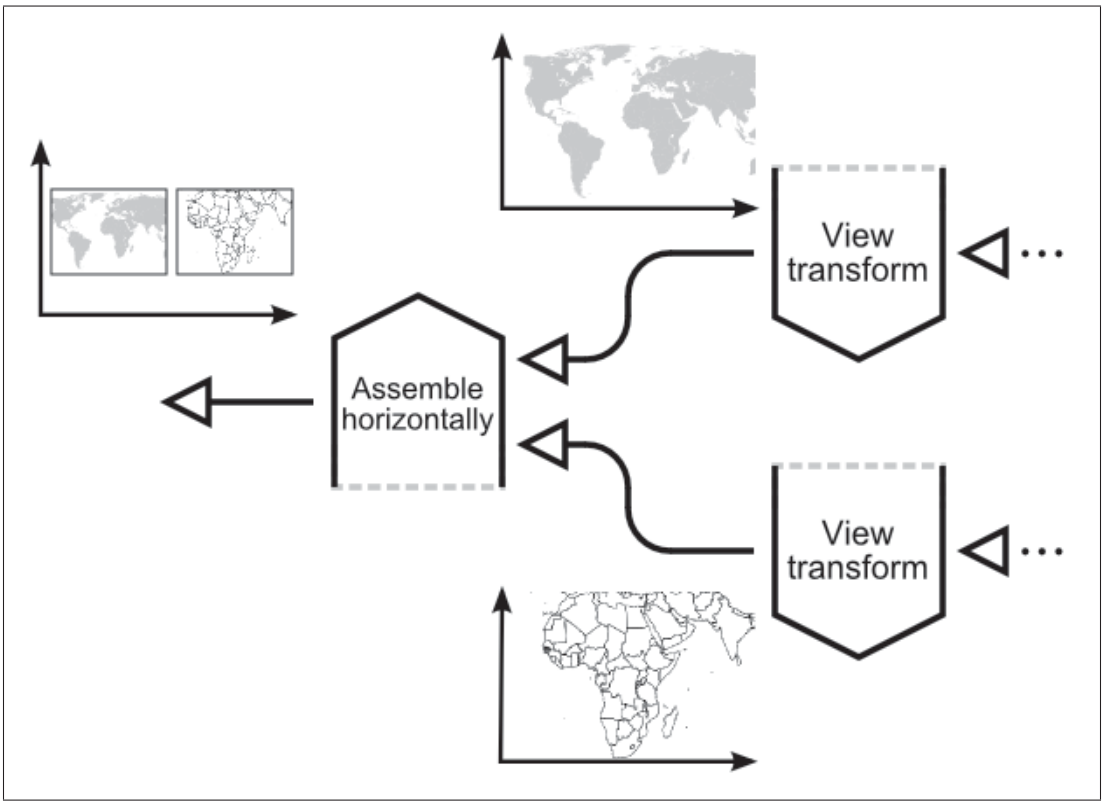


Figure 2.4 Side-by-side assembly of two views of the same data.

(North, 2000) discusses forms of coordinated views where navigating in one view causes a corresponding navigation in the other (navigation ↔ navigation coordination), or where selection in one view causes the other view to navigate to the corresponding element (selection ↔ navigation coordination).

Views may also be constrained to have one or both axes in common. For example, a scatterplot matrix (Hartigan, 1975) can be thought of as an assembly of a large number of individual views (the scatterplots), and in this case any two views that are adjacent typically share a common (horizontal or vertical) axis with the same scale, to ease the comparison of points across the views.

During assembly, it is also possible to draw line segments connecting corresponding elements in the two views, to explicitly link elements. Examples in previous work include (Diaconis and Friedman, 1980; Collins and Carpendale, 2007; Claessen and van Wijk, 2011).

Certain hybrid visualizations involve “stitching” two representations together along an edge, possibly showing the result within a single viewport window. Because the representations are essentially side-by-side, we can model this with the same kind of side-by-side assembly. Figure 2.5 shows MatLink (Henry and Fekete, 2007b) in this way. MatLink combines an adjacency matrix view of a network with an arc diagram (Bertin, 1967; Wattenberg, 2002) of the same network. The fact that the resulting hybrid may be displayed within a single window, rather than two side-by-side windows, is rather incidental, and can be accounted for by requiring that the two View transform operators remain synchronized.

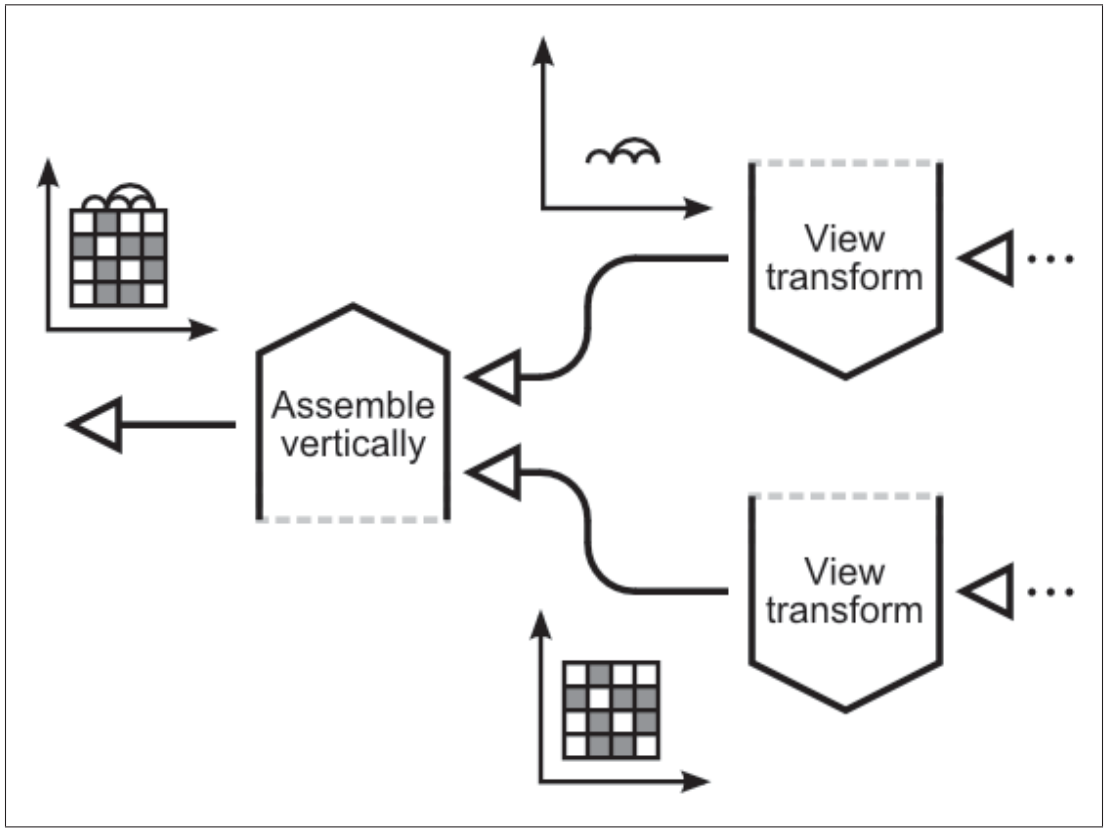


Figure 2.5 MatLink (Henry and Fekete, 2007b) can be thought of as a side-by-side assembly of an adjacency matrix and an arc diagram.

2.5.1.2 Overlaying Assembly

A second way of assembling the output of View transform operators is to render their output in overlapping spaces, compositing them with alpha blending, for example. Macroscope (Lieberman, 1997) is an example of this (Figure 2.6).

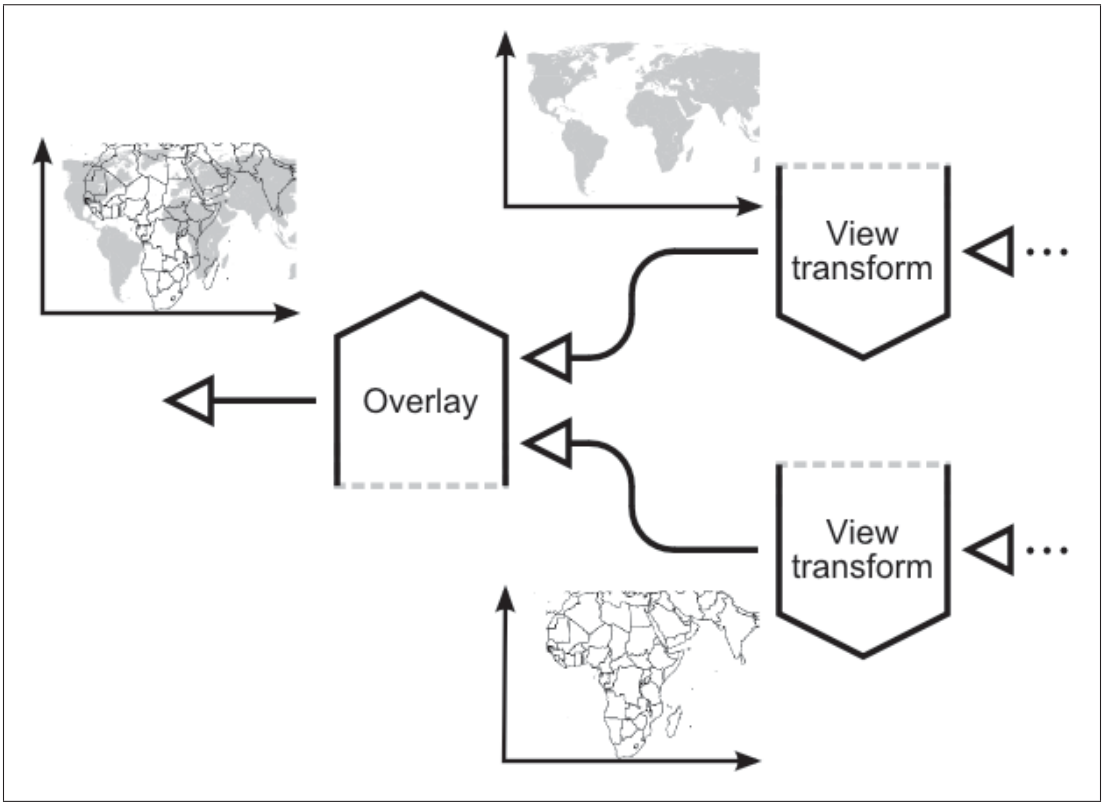


Figure 2.6 Overlaying assembly: A zoomed view of Africa is alpha blended over a world map, producing output similar to Macroscope (Lieberman, 1997).

An additional example of overlaying assembly is when two charts are drawn within the same 2D space. For example, a line chart of CO_2 emissions over time might be overlaid on a line chart of average global temperature over time. These two charts would share the same horizontal axis (time) but have different vertical axes (concentration and temperature, respectively).

2.5.2 Heterogeneous Combinations of Glyphs

Visualizations such as Elastic Hierarchies (Figure 2.7) or NodeTrix (Figure 2.8) do not involve simply overlaying two simpler visualizations. Instead, in both cases, the data is partitioned into

two subsets, each of which is rendered using a different representation (node-link and treemap in the former case, node-link and matrix in the latter) and the two representations are then combined within a common 2D space. In both Elastic Hierarchies and NodeTrix, each node of the dataset is shown only once, unlike the case in Figure 2.4 where each country of Africa appears twice, at different scales.

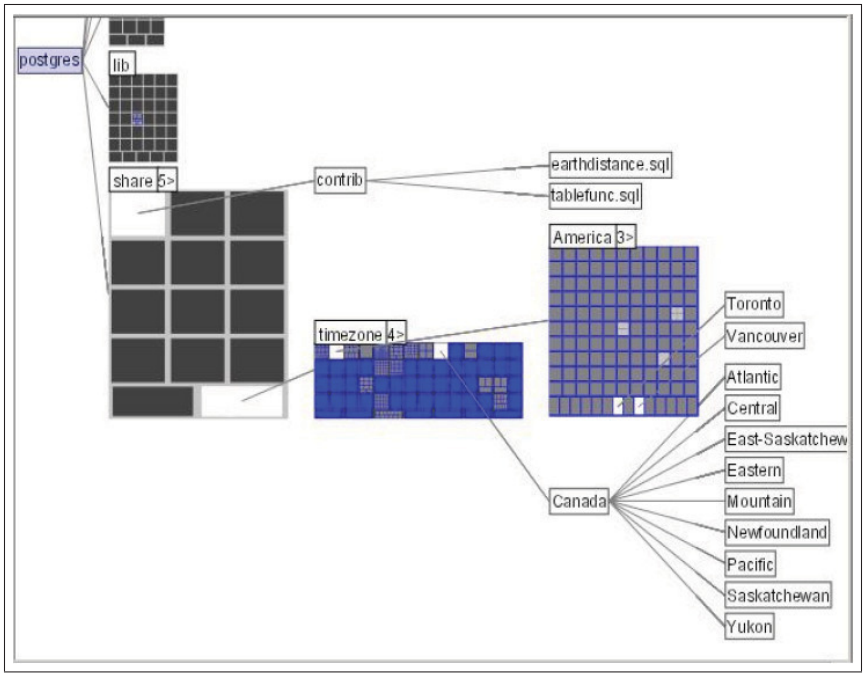


Figure 2.7 Elastic Hierarchies (Zhao *et al.*, 2005) combine node-link and treemap representations of tree data.

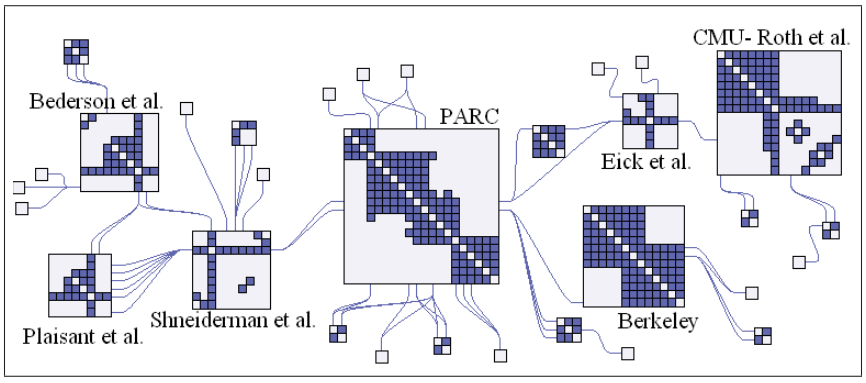


Figure 2.8 NodeTrix (Henry *et al.*, 2007b) combines node-link and adjacency matrix representations of network data.

Elastic Hierarchies and NodeTrix are examples of hybrids that we call *heterogeneous* visualizations. We model them with the pipeline in Figure 2.9, where data is split into two subsets, and the resulting geometric objects are later merged. The essential characteristic of a heterogeneous visualization is the splitting of the data into non-overlapping subsets that are fed into separate branches, each passing through a Glyph generation operator, later to be merged in a common 2D space.

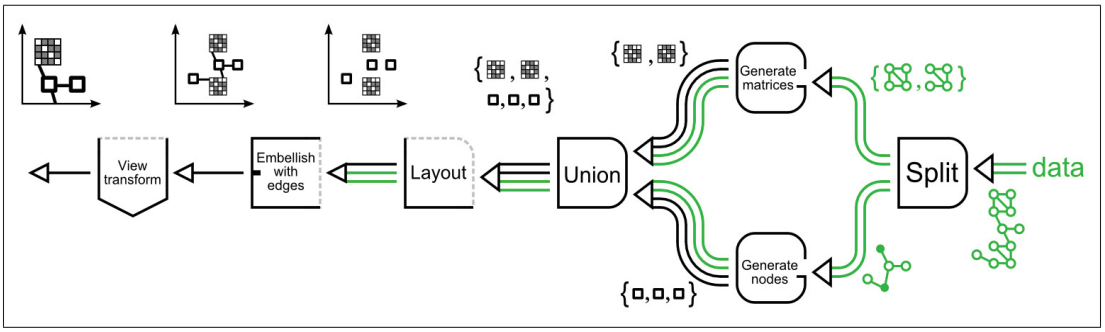


Figure 2.9 Pipeline describing NodeTrix.

2.5.3 Nesting Visualizations

It is also possible to reduce a visualization in size to play the role of a glyph inside a larger visualization. We refer to such combinations as *nested* visualizations. An example is modeled in Figure 2.10. The essential feature of such nesting is that the output of one Layout operator L_2 is fed into another Layout operator L_1, implying that the layout L_2 is applied recursively within the layout of L_1. This process could be extended, in theory, for visualizations having 3 or more layout levels.

Examples of nested visualizations in previous work include Microsoft Pivot[1], which nests photo thumbnails inside barchart bars; TableLens (Rao and Card, 1994), which nests numeric data and simple graphics inside a tabular layout; TopoLayout (Archambault *et al.*, 2007), which nests different graph layouts according to a decomposition of a graph; and barcharts in Tulip (Auber *et al.*, 2012) which display individual nodes inside bars (Figure 2.11).

[1] http://www.microsoft.com/silverlight/pivotviewer/

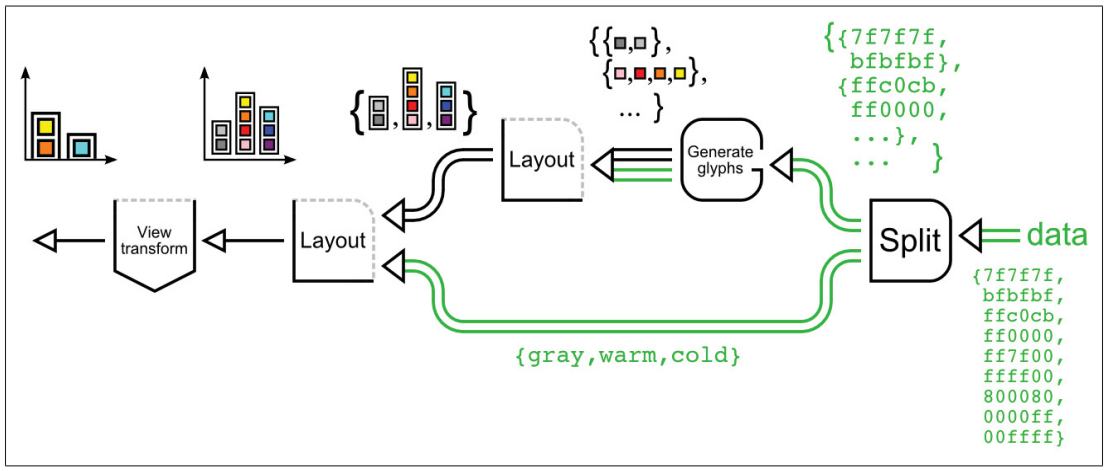


Figure 2.10 A nested visualization: a barchart containing color swatches.

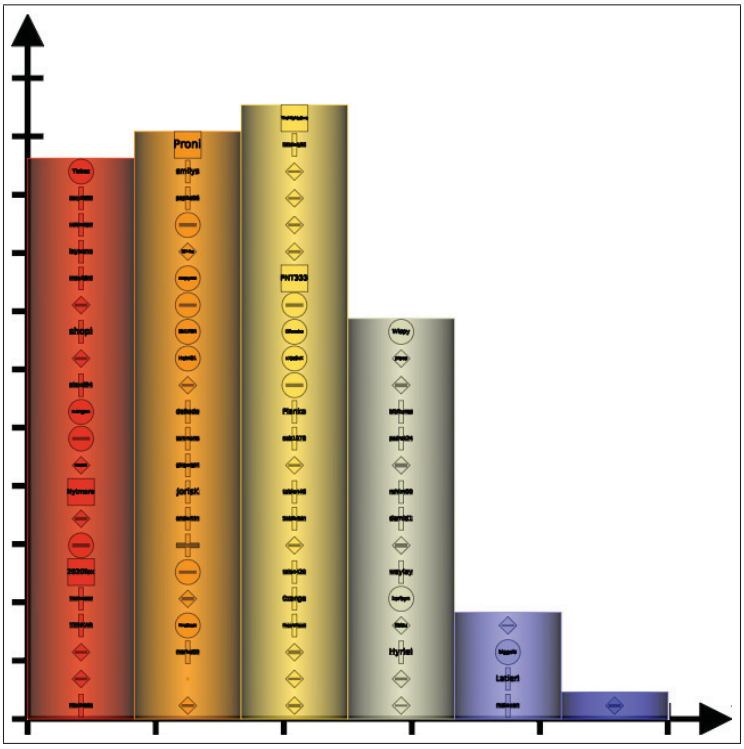


Figure 2.11 A barchart in Tulip (Auber *et al.*, 2012), showing individual graph nodes inside each bar.

During interaction, zooming in or out of a nested visualization may cause the representation of certain elements to change in a scale-appropriate manner. This behavior is usually called *semantic zooming* (Bederson and Hollan, 1994).

In the course of our search for examples of nested visualizations, a novel case occurred to us: the "bullet" glyphs proposed by Stephen Few[2] can be nested inside a Gantt chart, allowing for a richer indication of the progress of each stage of a project (Figure 2.12).

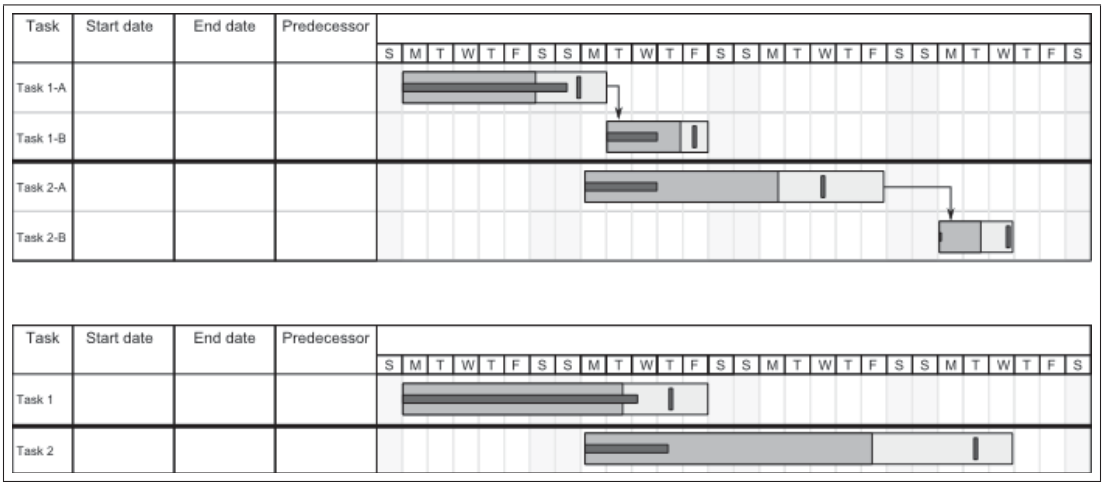


Figure 2.12 A mockup of a Gantt chart whose bars have been replaced with Few's bullet glyphs.

2.5.4 Hybrid Layout Operators

Yet another approach for creating a hybrid visualization is to combine aspects of the layout algorithms of two other visualizations. Given two pre-existing visualizations that use Layout operators L_A and L_B, respectively, the idea is to create a new operator $L' = L_A \oplus L_B$, where $\oplus$ is some kind of combination of operators. A simple example would be an L' that uses the x coordinate of L_A and the y coordinate of L_B. The new Layout operator L' can then be used inside any pipeline, even the simple, linear pipelines of Figures 2.1 and 2.2. Figure 2.13 shows an example where the layouts used in two barcharts are combined to produce a scatterplot's layout.

Some visualization systems allow for a whole range of Layout operators, where the x and y components can be modified independently. These include "graph drawing by axis separation" (Koren and Harel, 2005) and GraphDice (Bezerianos *et al.*, 2010b). For example, with

[2] http://www.perceptualedge.com/articles/misc/Bullet_Graph_Design_Spec.pdf

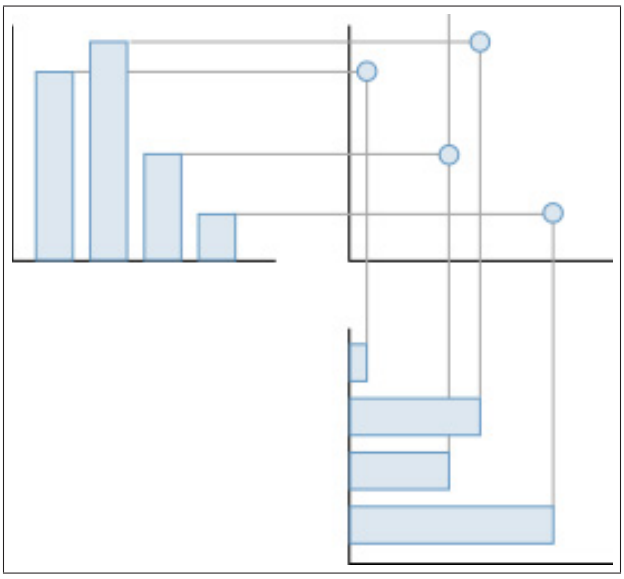


Figure 2.13 The layout of points in a scatterplot can be seen as a combination of the layouts in two barcharts.

GraphDice, a user may choose to layout the nodes of a graph with an x coordinate corresponding to the node's degree, and a y coordinate corresponding to the node's clustering coefficient.

Other combinations of layouts are not so obvious, but can still lead to novel visualizations. We examined two ways of depicting 1-dimensional datasets: the histogram (Figure 2.14, lower left), and a "TableLens-style" (Rao and Card, 1994) chart of records sorted by value (Figure 2.14, upper left). It was not immediately apparent to us if the layouts of these two charts could be combined, since they show glyphs corresponding to different kinds of data elements: the histogram shows rectangle glyphs corresponding to aggregated subsets of data, whereas the TableLens chart shows individual records as line segments. However, observe that, in the TableLens chart, the right extremities of the line segments are the most important portion of the line segments, since they convey the x-coordinate of each record. Notice also that these extremities can be binned (Figure 2.14, upper right) into rectangles of constant width, whose height is proportional to the number of records enclosed. Finally, these rectangles can be translated downward (Figure 2.14, lower right) to align their bottom edges, recreating the histogram's layout. The two charts on the right half of Figure 2.14 appear to be new hybrid charts that show both aggregate-level information (through the height of the rectangles) as well as individual data elements. All four charts in the figure have the same x axis, but the positioning of

glyphs (points and/or rectangles) along y is different between the upper two charts and lower two.

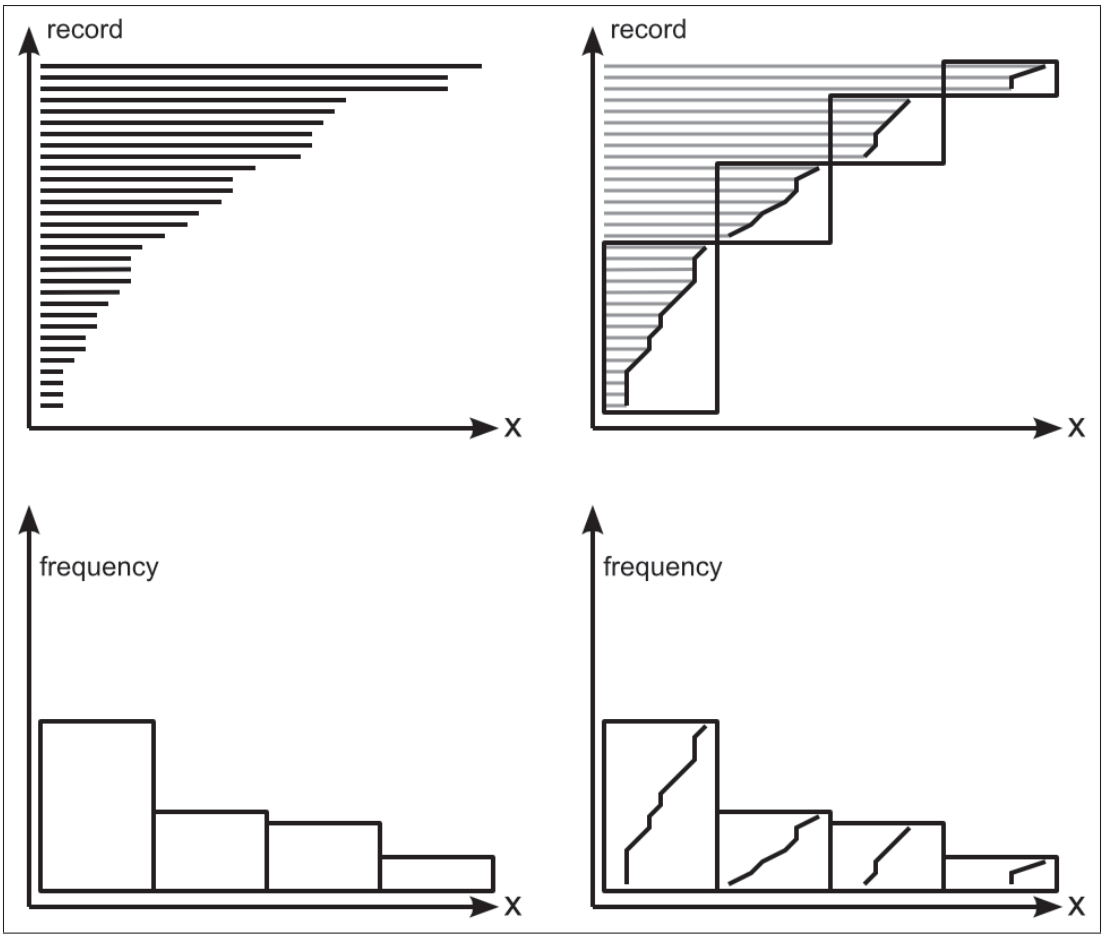


Figure 2.14 Visualizations of a 1-dimensional set of numbers. Upper-left: a TableLens-style depiction (Rao and Card, 1994). Lower-left: histogram of the same data. Upper-right and lower-right: hybrid depictions that mix the layouts of the first two charts.

2.5.5 Hybrid Glyph Generation Operators

The last approach we discuss involves creating a new Glyph generation operator $G' = G_A \oplus G_B$ by combining characteristics of two pre-existing operators G_A and G_B, and then using G' within a pipeline. For example, if G_A maps some data variable x_1 to the color of glyphs, and G_B maps another variables x_2 to the size of glyphs, then a simple combination G' could map x_1, x_2 to color and size, respectively, or may instead map x_1, x_2 to the width and height of the

glyphs. Figure 2.15 shows another example. Such combinations are easy to perform because the attributes of the glyphs are orthogonal.

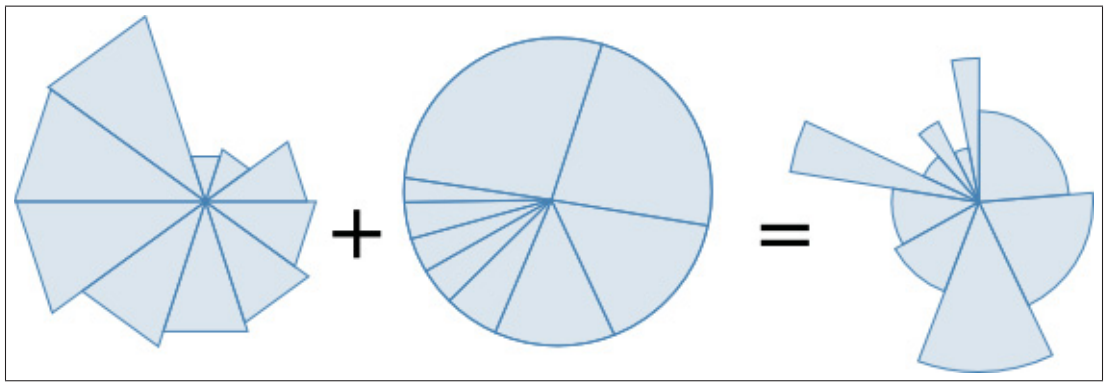


Figure 2.15 A Nightingale diagram varies the radius of its glyphs, whereas a pie chart varies their angles. Combining these yields a new visualization where each glyph's radius and angle is varied.

However, if the glyphs to be combined do not have the same shape or same attributes, combining them may be less obvious and require creativity. The later sections present examples of combinations of the point glyphs in a scatterplot with other points or with the polygonal lines in a parallel coordinates plot, yielding hybrid glyphs.

2.6 Example Design Studies

To demonstrate the value of distinguishing different ways of combining visualizations, we now present the results of two attempts to combine different visualizations. First, we consider ways of combining scatterplots, and then ways of combining scatterplots and PCP (Inselberg, 1985; Wegman, 1990). In each case, rather than proceed in a completely ad hoc manner, the preceding 6 kinds of combinations served as analogies and focal points for brainstorming and formulating questions about how the visualizations might be combined.

2.6.1 Example 1: Combining Scatterplots

In this section, we consider ways of combining two (or more) scatterplots. Such combinations could be useful, for example, when visualizing a 4-dimensional dataset, where each tuple is of the form (a,b,c,d), and each scatterplot serves to show two of the 4 dimensions. We consider

most of the approaches presented in the preceding sections for combining visualizations to see what results can be produced.

Side-by-side and overlaying assembly yield Figure 2.16, A and B. Notice that it is necessary in both cases to link the corresponding points with line segments to make the visualizations intelligible. Interestingly, the combination in Figure 2.16, A, was proposed in (Diaconis and Friedman, 1980), where line segments were also used to link points.

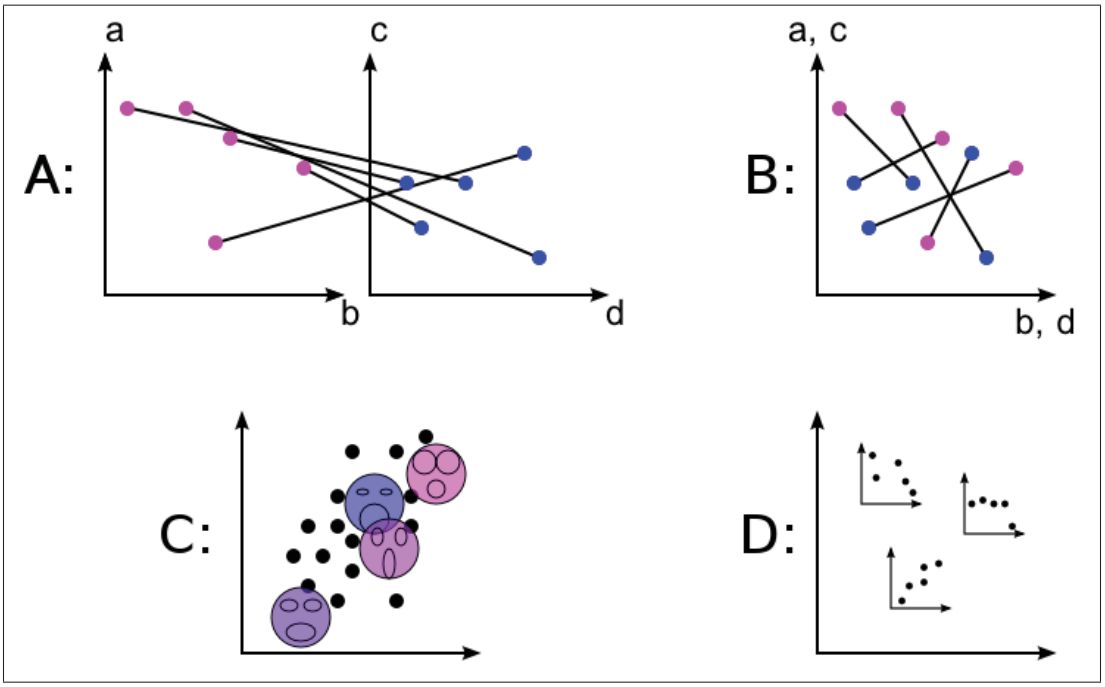


Figure 2.16 Combinations of scatterplots. A: side-by-side assembly, linked with line segments. B: overlaying assembly, linked with line segments. C: a heterogeneous combination. D: nesting.

The combination in Figure 2.16, B, however, is novel to our knowledge, and although it was inspired by thinking about overlaying assembly (section 2.5.1.2), it could also be interpreted as a visualization of hybrid glyphs (section 2.5.5) where pairs of points are merged into line segments. Thus, more than one approach for combining visualizations may yield the same result. Indeed, we do not claim that the approaches for combining visualizations are mutually exclusive, but rather that they provide useful perspectives for describing and thinking about combinations of visualizations.

Figure 2.16, C, shows a heterogeneous combination of Chernoff faces (Chernoff, 1973) and simple points. The Chernoff faces, or other glyphs, would depict variables beyond the x and y coordinates of the scatterplot. This hybrid was inspired by simply asking what a heterogeneous combination of scatterplots would look like. It was not immediately obvious whether this combination has any use. Upon reflection, however, we notice that if all points were displayed as Chernoff faces, this could easily create inter-glyph occlusion. It could indeed be desirable for the user to specify a subset of points to display as glyphs. Such a subset could be specified through direct selection (pointing, or lasso selection), or by manipulating a slider, e.g., to specify that we want all tuples with some variable within some range to be displayed as Chernoff faces.

Figure 2.16, D, nests scatterplots within a scatterplot. This could be useful in a 4-dimensional dataset that has been partitioned into a small number of subsets, such as clusters. Each subset, or cluster, could have a corresponding small scatterplot to show individual tuples along two of the dimensions. These small scatterplots could, in turn, be located within the larger scatterplot by their average value along the two remaining dimensions.

2.6.2 Example 2: Scatterplots + Parallel Coordinates

We now perform a similar exercise, trying to combine scatterplot(s) with PCPs. Side-by-side assembly produces Figure 2.17, A, and again requires line segments to explicitly link elements.

Overlaying assembly produces Figure 2.17, B, which we note is very similar to Scattering Points in Parallel Coordinates (Yuan *et al.*, 2009) (SPPC) (Yuan *et al.*, 2009). SPPC can be thought of as an overlay assembly (section 2.5.1.2), with the additional characteristic that the polygonal lines in the PCP are replaced with curves *passing through* the scatterplot points. Such curving of the polygonal lines could be thought of as the result of merging the layout of the PCP and scatterplots (section 2.5.4), or as a merging of the scatterplot points and PCP polygonal lines (section 2.5.5). As in the previous section, there is more than one way of describing the hybrid combination.

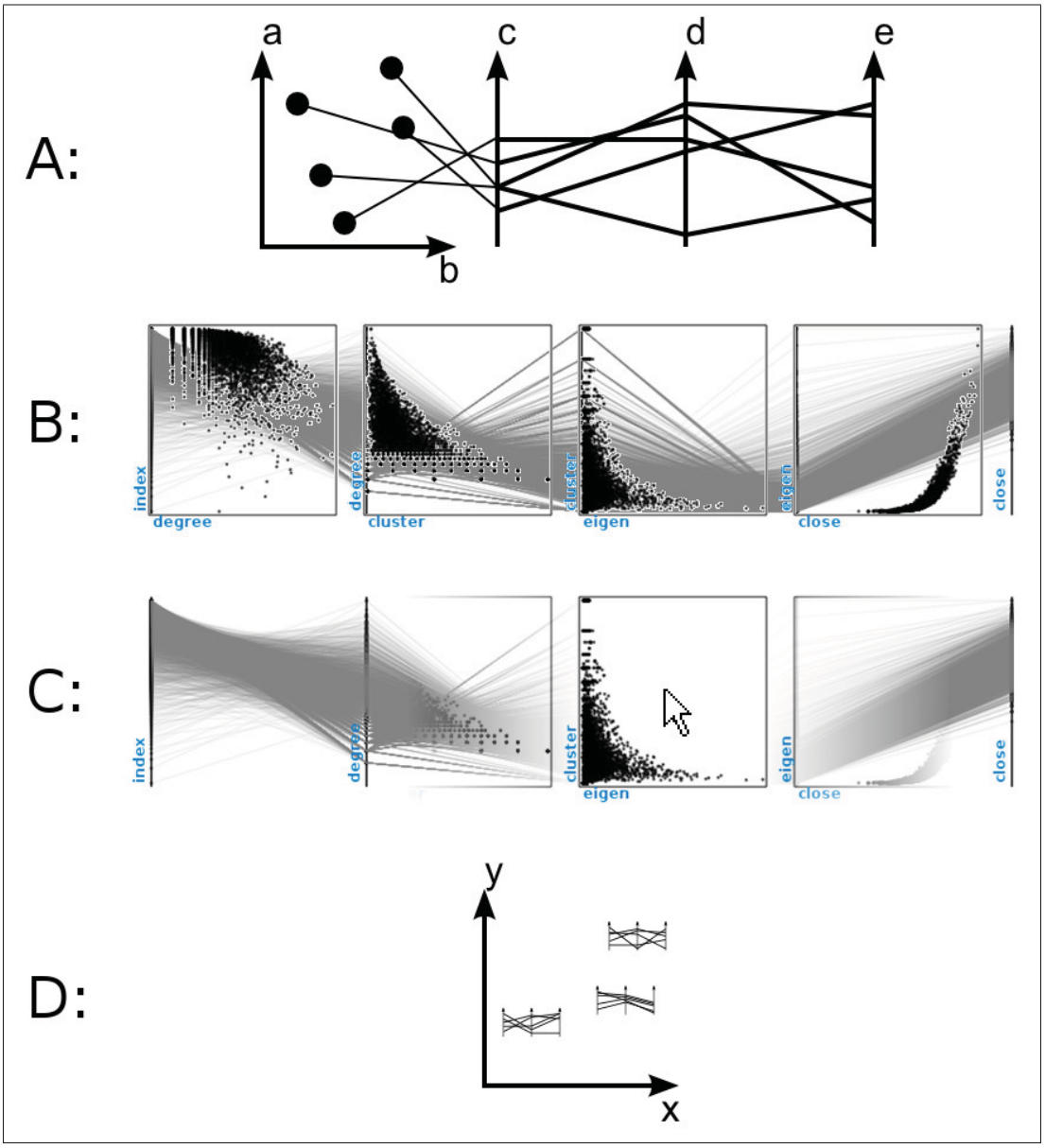


Figure 2.17 Combinations of scatterplots and parallel coordinate plots (PCP). A: side-by-side assembly, linked with line segments. B: overlaying assembly, similar to (Yuan *et al.*, 2009). C: a heterogeneous combination, where the focus (under the mouse cursor) is rendered as a scatterplot, and the context is rendered in PCP form. D: PCPs nested within a scatterplot.

A heterogeneous combination of scatterplots and PCPs is shown in Figure 2.17, C. We imagine that it might be useful for the user to see a "focal" region as a scatterplot, and to see the surrounding "context" as a PCP, since the polygonal lines of the PCP might make it easier to see links to distant axes. Thus, Figure 2.17, C shows a mouse cursor over the focal region, which

might follow the cursor as it moves, like a lens. Notice, however, that in this heterogeneous visualization, rather than having the dataset partitioned into two subsets of tuples, it is divided into two subsets of dimensions, each of which is rendered differently. In other words, if the data is stored in a table (rows corresponding to tuples, columns to dimensions), the table is split into two subsets of columns rather than two subsets of rows. Two of the dimensions are rendered as a scatterplot, and the remaining are shown with a PCP. Of course, it would also be possible to partition the data into subsets of tuples, e.g., showing selected tuples as points and remaining ones as polygonal lines.

Nesting PCPs within a scatterplot produces Figure 2.17, D, which could be used in a way analogous to the nested visualization in Figure 2.16, D.

Finally, during this design study, another hybrid that occured to us is shown in Figure 2.18. This visualization is redundant in that pairs of dimensions are shown both as a scatterplot and a pair of PCP axes. In terms of the approaches we have described for combining visualizations, Figure 2.18 could be seen as multiple visualizations assembled side-by-side (section 2.5.1.1).

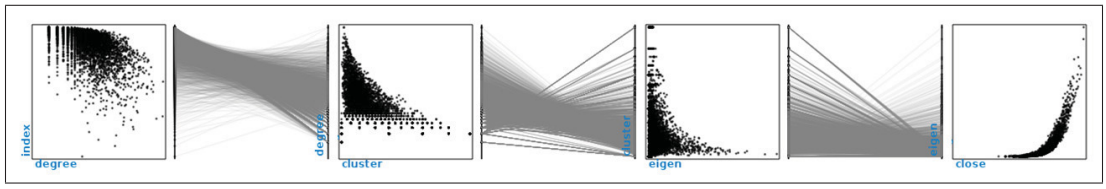


Figure 2.18 Another combination of scatterplots with PCPs.

2.7 Conclusion

Our work distinguishes between 6 kinds of combinations of visualizations: side-by-side assembly, overlay assembly, heterogeneous visualizations, nested visualizations, hybrid layouts, and hybrid glyphs. We have shown how to model each kind of combination as a modification to the visualization pipeline, and positioned previous work with respect to the 6 kinds of combinations. We have also presented a few novel visualizations: bullet glyphs nested within a Gantt chart (Figure 2.12), variants of histograms that display individual records (Figure 2.14), hybrid combinations of scatterplots (Figure 2.16) and of scatterplots and parallel coordinate plots

(Figures 2.17 and 2.18). These novel hybrids were discovered by thinking deliberately about the different kinds of combinations that are possible with existing visualizations, demonstrating the generative design potential from having explicitly distinguished types of combinations.

2.8 Future Directions

Future work could examine different kinds of tasks and interactions for enacting changes to the visualization pipeline, given that the pipeline may split, merge, or pass through multiple Layout operators, requiring extensions to previous work on visualization tasks and interactions. Future studies might also compare different kinds of hybrid visualizations, to establish which ones succeed at combining the advantages of component visualizations, and to what degree.

Future work might also extend our work to 3-dimensional layouts and 3D visualizations. In 3D, certain visualizations have proposed updating pipeline operators automatically in response to camera motion (such as distortion viewing (Carpendale *et al.*, 1997) which updates 3D layout based on camera position; and importance-driven rendering (Viola *et al.*, 2004) which updates opacity based on camera position). In a hybrid 3D visualization, new kinds of automatic responses to camera motion may be warranted, such as changing the representation of subsets of data as the user moves through the 3D space.

CHAPTER 3

CONNECTEDCHARTS: A HYBRID VISUALIZATION BY EXPLICIT LINKING

Christophe Viau[1], Michael J. McGuffin[1],
[1]Department of Software and IT Engineering, École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3
Published in Computer Graphics Forum (Proceedings of EuroVis 2012), vol. 31, n. 3, 2012,
p. 1285-1294

3.1 Abstract

Multidimensional multivariate data can be visualized using many different well-known charts, such as bar charts, stacked bar charts, grouped bar charts, scatterplots, or pivot tables, or also using more advanced high-dimensional techniques such as scatterplot matrices or parallel coordinate plots. These many techniques have different advantages, and users may wish to use several charts or data graphics to understand a dataset from different perspectives. We present ConnectedCharts, a technique for displaying relationships between multiple charts. ConnectedCharts allow for hybrid combinations of bar charts, scatterplots, and parallel coordinates, with curves drawn to show the conceptual links between charts. The charts can be thought of as coordinated views, where linking is achieved not only through interactive brushing, but also with explicitly drawn curves that connect corresponding data tuples or axes. We present a formal description of a design space of many simple charts, and also identify different kinds of connections that can be displayed between related charts. Our prototype implementation demonstrates how the connections between multiple charts can make relationships clearer and can serve to document the history of a user's analytical process, leading to potential applications in visual analytics and dashboard design.

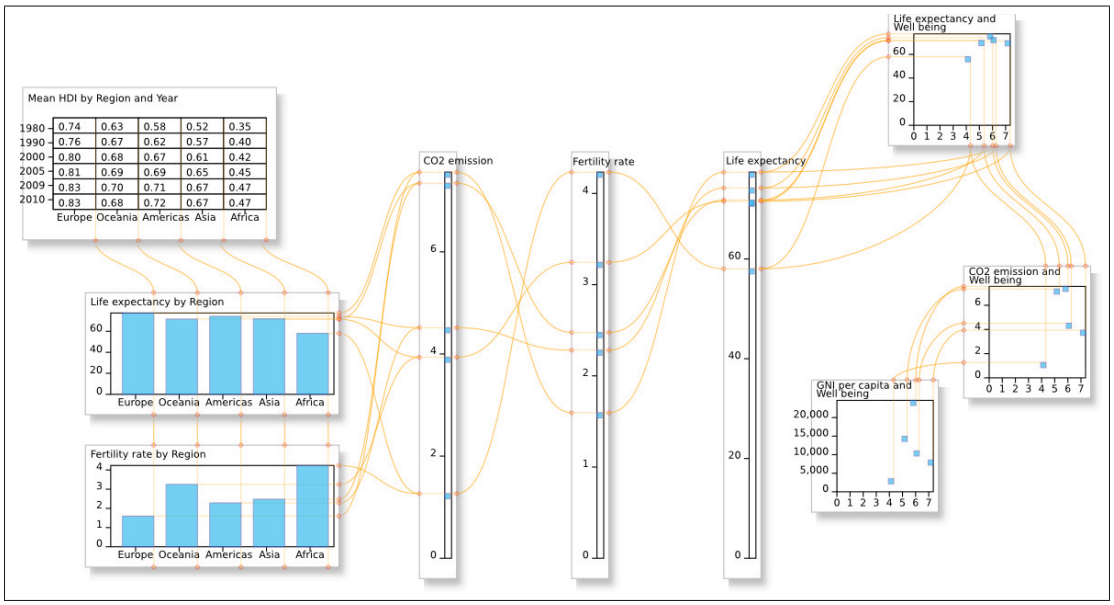


Figure 3.1 A ConnectedChart showing a text table, bar charts, parallel coordinate axes, and scatterplots. (Real data set.)

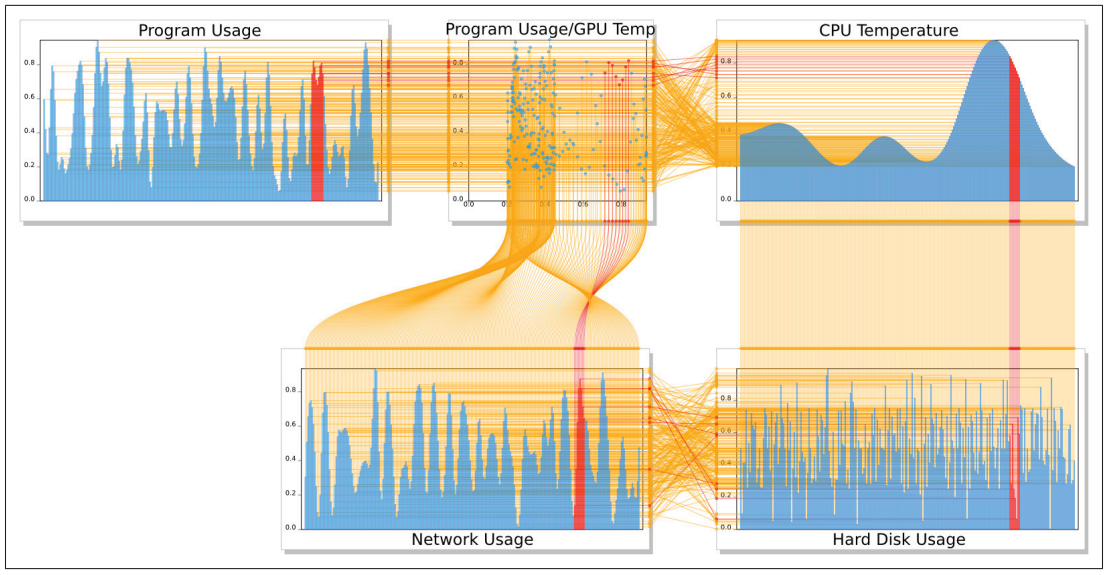


Figure 3.2 A ConnectedChart with four barcharts, and a scatterplot, displaying 5 dependent variables that are functions of time. In this case, GPU temperature is a smooth, slowly varying function of time, hence the connection between the scatterplot's horizontal axis and the barchart below it exhibits an interesting, continuous variation. (Synthetic data set.)

3.2 Introduction

The use of multiple views onto the same data, or related data, is a common approach in visualization. Users are given multiple perspectives of their data, allowing them to benefit from

the advantages of each view, and to compare across views. To help the user understand the relationships between views, some form of *linking* is required. Two common approaches are to use color or drawing line segments or curves to link corresponding data elements. However, if there are many data elements and all links are displayed at once, we either quickly run out of distinguishable colors, or suffer from line clutter. For this reason, linking is often only displayed in response to mouse motion: hovering over an element causes the links between that element and related ones to be displayed (either in the form of a color highlight or line or curve segments). This approach is elegant and can be applied to even large data sets, but it constrains the user to exploring relationships one element at a time, requiring the user to interact with the data, possibly for an extended period of time, to extract useful visual feedback.

We propose a new technique, called ConnectedCharts (Figures 3.1, 3.2), situated in the middle ground between the extremes of linking all elements, and only linking the element under the mouse cursor. In our work, curves are drawn between charts (data graphics), showing the correspondence between data elements (tuples) when possible, or otherwise between axes. Occlusion is avoided within each chart by "anchoring" the curves to the edges or axes of the chart. Also, rather than drawing all possible links, only those that have been created by the user (in the process of creating the charts) are displayed.

We have applied our technique to visualizing multidimensional multivariate data, such as data from a relational database table. Some very flexible visualizations of such data already exist, for example, FLINA (Claessen and van Wijk, 2011), which supports combinations of scatterplots and parallel coordinate plots, and Polaris (Stolte *et al.*, 2002) / Tableau (Mackinlay *et al.*, 2007), which can display multiple charts of the same type within a tabular grid. ConnectedCharts supports scatterplots and PCPs, but unlike FLINA, it also supports bar charts and other 2D charts, and also allows data tuples to be aggregated differently in each chart. Compared to Polaris / Tableau, ConnectedCharts offers more flexibility in that different kinds of charts can be instantiated at once, and can be positioned freely within a 2D space instead of being limited to a grid.

A user may use ConnectedCharts to explore a data set, creating new charts to answer new questions, with connections displayed to the previous charts. In such a scenario, the connections may serve to record and retrace the history of a user's analytical steps. A set of charts and their connections may also be designed and presented to an end-user, for use as a dashboard, in which case the connections elucidate the relationships between the views of the data, even when related charts are not side-by-side.

Although ConnectedCharts can result in many line segments or curves being displayed (Figure 3.2), the same is also true of parallel coordinate plots, where the line segments or curves serve to depict distributions, relationships, correlations, clusters, and inverse correlations. ConnectedCharts can be thought of as a generalization of parallel coordinate plots, that allow the advantages of these connections to be leveraged not just for 1-dimensional axes, but for many kinds of charts (Figure 3.3).

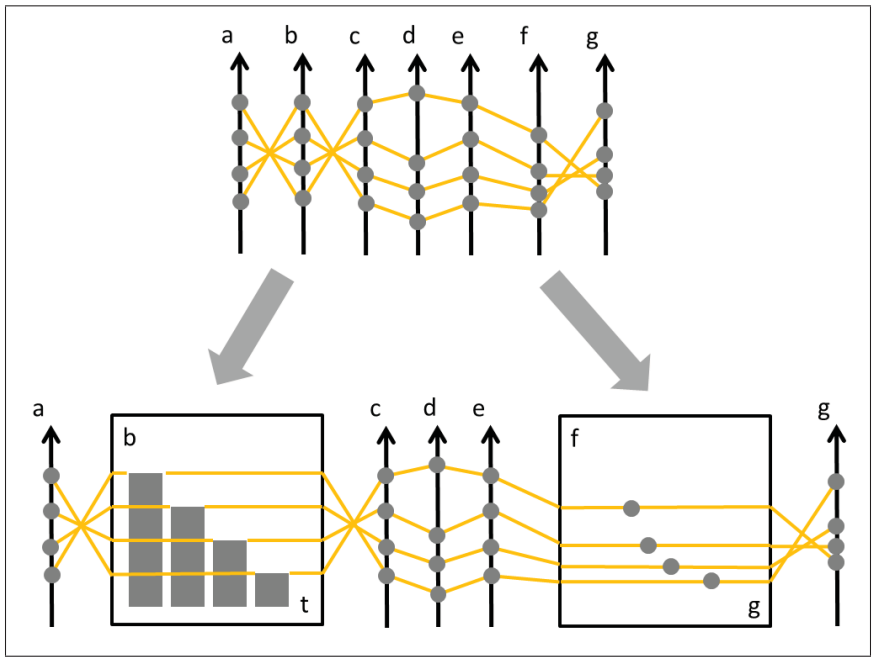


Figure 3.3 One of the potential uses of ConnectedCharts is to enrich parallel coordinate displays by expanding axes interactively. Here, the b axis is converted into a barchart showing time t, and the f axis is converted into a scatterplot against g. ConnectedCharts allow charts to be intermixed with parallel coordinate axes.

Our contributions include (1) a formal description of a design space of charts based on plots of 2D rectangles, general enough to subsume scatterplots, bar charts, Gantt charts, and variants; (2) an analysis of the types of connections that can be displayed between charts; (3) a demonstration of how ConnectedCharts can be used to produce SPLOMs, PCPs, and hybrid combinations of these and other charts; (4) a generalization of the Attribute Relation Graph of Interest (ARGOI) presented in (Claessen and van Wijk, 2011).

3.3 Background

3.3.1 Linking and coordination across views

Many visualizations involve multiple views of data that are somehow *linked* to convey a relationship between the views. Buja et al. (Buja *et al.*, 1991) give several techniques for this: linking with color (e.g., drawing corresponding elements with the same color), linking "by drawing lines connecting [corresponding] points", and linking "over time" with a "smooth animation" from one view to another. Wong and Bergeron (Wong and Bergeron, 1997) point out that linking can also be done by aligning axes in different views, as is done with the scatterplots in a scatterplot matrix (SPLOM) (Hartigan, 1975).

At least some of these linking techniques can be performed interactively. For example, it may not be feasible to draw *all* data elements in colors that distinguish the corresponding subsets of points from each other, nor may it be useful to draw all line segments between corresponding points. However, if the mouse cursor hovers over an element, the corresponding elements could then be indicated with a highlight color or line segments. An example of such interactive linking is "brushing and linking" (Becker and Cleveland, 1987).

Interactive linking of views is also called *coordination* (Wang Baldonado *et al.*, 2000; Roberts, 2007), which is also very common in visualizations. North (North, 2000) presents a software framework for this, and distinguishes 3 types of coordination: selection $\leftrightarrow$ selection (i.e., selection in one view causes a selection in another), selection $\leftrightarrow$ navigation, and navigation $\leftrightarrow$ navigation.

Of particular relevance to the current work is linking done using line segments or curves. Interactive linking across views with line segments dates back at least to work by Ted Nelson in the early 1970s (Nelson, 1999). Line segments between views have been used for meta-visualization of the coordination of views (Weaver, 2005; Tobiasz *et al.*, 2009). There are also several examples of line segments and curves drawn between corresponding data elements in different views. "M and N" plots (Diaconis and Friedman, 1980) are an early example: in a "2 and 2" plot, points in corresponding 2D scatterplots are connected to convey 4-dimensional tuples. Parallel coordinates (Inselberg, 1985; Wegman, 1990) use line segments to connect tuples across multiple axes, and each axis can be thought of as a 1D "view" of the data. VisLink (Collins and Carpendale, 2007) is a general framework for connecting data elements across views. More recent examples of connecting elements across views include (Aris and Shneiderman, 2007; Viau *et al.*, 2010). There is also evidence that connecting elements allows a user to find elements faster than with simple highlighting (Steinberger *et al.*, 2011).

With ConnectedCharts, we can make use of interactive color highlighting, but also display static curves to link together charts. These curves are distinct from previous work in a few ways: first, rather than statically display all possible linking curves between corresponding elements, we only display those that the user has established through their interactions with the charts, avoiding excessive clutter; second, the connecting curves between charts are "anchored" to the axes or edges of charts, avoiding clutter or occlusion within each chart; third, we allow for several kinds of charts to be connected, and identify several kinds of connections that can be shown (see section 3.6).

Note that we do not propose ConnectedCharts as a replacement to brushing and linking. Instead, we see these two approaches as having complementary advantages (Figure 3.4): brushing and linking is flexible and scales well to large data sets, but requires the user to invest time moving a pointing device over the data. ConnectedCharts, however, can reveal relationships at a glance prior to any interaction, and even without reading axis labels, just like how the connective lines in parallel coordinates reveal relationships between their axes.

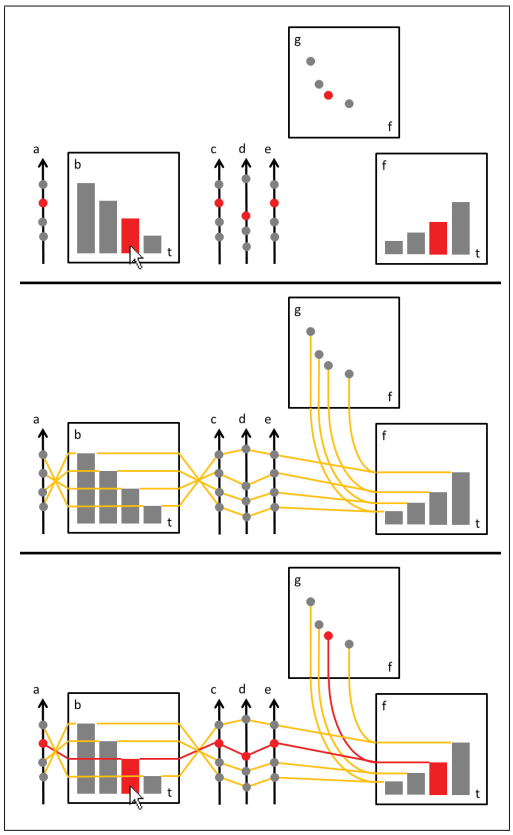


Figure 3.4 Brushing and connections are complementary approaches. Top: brushing without connections. The user must interactively roll over data elements with a pointing device to discover relationships. Notice the parallel coordinate axes in the center, that can be brushed, but that do not display connections, making them less informative than usual parallel coordinates. Middle: Displaying connections between charts and between parallel coordinate axes allows the user to see common variables, correlations (e.g., between variables *c*, *d*, and *e*), and inverse correlations (between *a* and *b*, and between *b* and *c*) without any mouse interaction. Bottom: The advantages of both approaches combined.

3.3.2 Support for history in visualization

Recent work (Heer *et al.*, 2008; Shrinivasan and van Wijk, 2008) has presented support for navigating a user's history of their views of data. The connections in ConnectedCharts can be used to convey and retrace a user's history of analytical steps, and unlike previous work, the views and their (historical) connections are displayed in the same 2D space, rather than in separate viewports. In our prototype, the user can zoom in on a single view or zoom out to see its connections and other views. Furthermore, the user is free to edit connections, meaning

that they may no longer reflect the user's true history; this can be both an advantage and a disadvantage.

3.3.3 Multiple views of multidimensional multivariate data

Dimensional anchors (Hoffman *et al.*, 1999) and FLINA (Claessen and van Wijk, 2011) allow for many combinations and variants of scatterplots and parallel coordinate plots to be displayed of multivariate data. As will be seen, ConnectedCharts also allow for combinations of scatterplots and PCPs. However, ConnectedCharts also supports other kinds of charts (such as bar charts, stacked bar charts, and grouped bar charts) and supports aggregated data. We also generalize the notion of ARGOI presented in (Claessen and van Wijk, 2011).

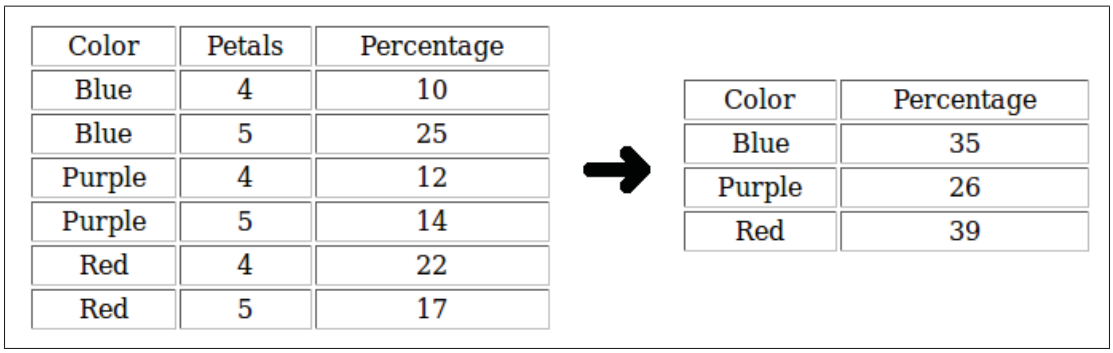
Polaris (Stolte *et al.*, 2002), and its successor Tableau (Mackinlay *et al.*, 2007), allow for multiple charts to be displayed of multidimensional multivariate data, arranged in a grid of rows and columns; the charts are implicitly linked by the alignment of their axes. As presented in the literature, the charts in the grid are always of the same type (e.g., a grid of scatterplots, or of bar charts). ConnectedCharts, in contrast, allows charts of different types to be freely positioned within a 2D space, giving the user more freedom. Nevertheless, grid-like arrangements are still permissible in ConnectedCharts (e.g., Figure 3.12).

Product plots (Wickham and Hofmann, 2011) allow many kinds of charts to be generated based on a few variants of rectangles: *bars*, *spines*, *tiles* and *flucts*. The charts in ConnectedCharts are also based on plots of rectangles. ConnectedCharts does not support all the kinds of charts in product plots, but supports others (such as scatterplots) and could eventually be extended to support all product plots.

3.4 Data model

Many data sets can be thought of as a table where each column is an attribute (or field) and each row is a tuple. Typically, some of the attributes are best thought of as *independent*, while the others are *dependent*. These are often referred to as dimensions and measures, respectively, in the database literature, and correspond to the domains and codomains of a function. For

example, in Figure 3.5(left), color and petals are independent, percentage is dependent, and the data set can be thought of as a function, or mapping, from (color, petals) pairs to percentages.



Color	Petals	Percentage
Blue	4	10
Blue	5	25
Purple	4	12
Purple	5	14
Red	4	22
Red	5	17

Color	Percentage
Blue	35
Purple	26
Red	39

Figure 3.5 An example flower data set. Left: the raw data d : Color $\times$ Petals $\mapsto$ Percentage. Right: the data aggregated over petals, yielding $d^{[Petals]}$: Color $\mapsto$ Percentage.

As another example, consider a table d with 6 columns: 4 independent variables corresponding to product P, region R, year Y, and month M, and 2 dependent variables corresponding to sales S and expenses E. The table can be thought of as a mapping $d : P \times R \times Y \times M \mapsto S \times E$. Written in another way, the sales $s \in S$ and expenses $e \in E$ are a function $(s,e) = d(p,r,y,m)$ of product $p \in P$, region $r \in R$, year $y \in Y$ and month $m \in M$.

It is common to also distinguish between *categorical* attributes (also called nominal, finite, or discrete) such as a set of products, and *quantitative* attributes (also called metric, or continuous) such as real numbers. For example, Mackinlay et al. (Mackinlay *et al.*, 2007) distinguish between quantitative dependent, quantitative independent, and categorical data fields, which they denote as Qd, Qi, and C, respectively. In our work, we note that quantitative independent variables must be discretized to a finite number of values to fit in computer memory, and therefore independent variables can always be thought of as discrete, and therefore "categorical" in some sense. For example, time is normally thought of as quantitative, but must be discretized when used as an independent variable (e.g., reduced to a set of months or days). Thus, for simplicity, we treat all independent variables as categorical, meaning discrete or discretized. On the other hand, the dependent variables, such as sales, are quantitative in the most general case. Hence, we define $C_1, \ldots, C_M$ as the M (categorical) domains, and $Q_1, \ldots, Q_N$ as

the N (quantitative) codomains in the dataset. We can then think of a dataset d as a mapping $d : C_1 \times \ldots \times C_M \mapsto Q_1 \times \ldots \times Q_N$, with $(q_1, \ldots, q_N) = d(c_1, \ldots, c_M)$.

A common operation in database systems is aggregating (also called rolling-up or projecting), which removes one of the independent variables C_i in the dataset and replaces the dependent variables with aggregations (e.g., sums, or averages) over the values of the removed variable. Figure 3.5(right) shows an example of this. In general, aggregating dataset d over the domain C_i can be defined as

$$\begin{aligned} & d^{[C_i]}(c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_M) \\ & = \sum_{c^\star \in C_i} d(c_1, \ldots, c_{i-1}, c^\star, c_{i+1}, \ldots, c_M) \end{aligned}$$

Returning to the earlier example involving sales and expenses, an aggregation over both year and month would be defined as

$$(s, e) = d^{[Y,M]}(p, r) = \sum_{(y,m) \in Y \times M} d(p, r, y, m)$$

In general, any dataset d can be aggregated along any combination of its independent variables, yielding a collection of tables $d, d^{[C_1]}, d^{[C_2]}, d^{[C_1,C_2]}, \ldots, d^{[C_1,\ldots,C_M]}$. Each of these tables can be visualized using various different charts.

3.5 A design space of charts

Before identifying different kinds of connections that can be shown between data graphics or charts, we first define the different kinds of charts to consider. Rather than consider all possible charts or visualizations, we have identified a small number of "ingredients" that are amenable to analysis, but that can be combined to yield a rich design space containing many commonly known charts, as well as a few novel kinds of charts.

We assume that each chart shows all the tuples of a data set d, which may or may not have been aggregated over some independent variables. We further assume that each tuple is represented

graphically as a rectangle in the chart, where each rectangle has a position (x,y), width w, and height h. The chart maps each tuple $(c_1,\ldots,c_M,q_1,\ldots,q_N)$ to a rectangle (x,y,w,h). For example, in a bar chart, each rectangle has $x = c_i$ for some categorical variable c_i (ignoring scaling factors) and $y = 0$, as well as width $w = K$ for some constant K, and height $h = q_j$ for some quantitative variable q_j (again, ignoring scaling factors). We can therefore write that the rectangles are given by $(x,y,w,h) = (c_i,0,K,q_j)$. Converting this to a shorthand notation, we can define bar charts as charts with a mapping of the form $(C,0,K,Q)$ (Figure 3.6, upper left).

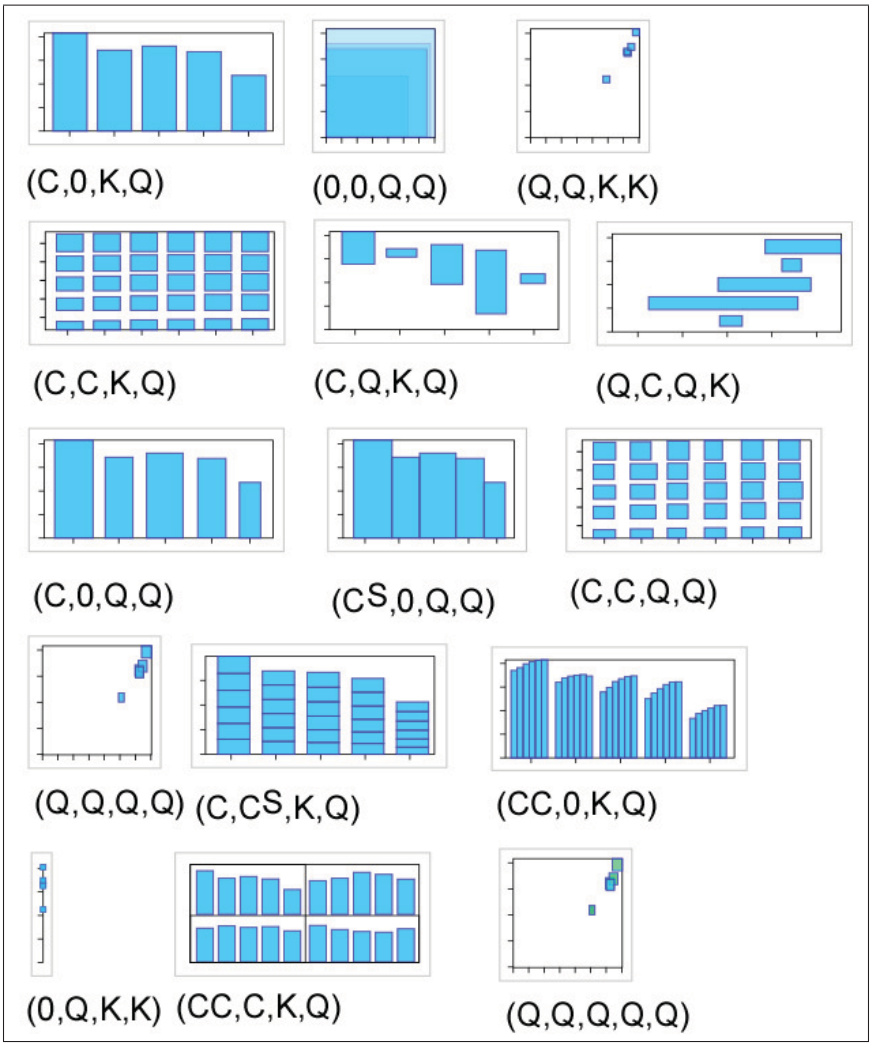


Figure 3.6 Examples of charts in the design space. Well-known charts include bar charts $(C,0,K,Q)$, scatterplots (Q,Q,K,K), Gantt charts (Q,C,Q,K), stacked bar charts (C,C^S,K,Q), grouped bar charts $(CC,0,K,Q)$, parallel bar charts (CC,C,K,Q). Interesting variants include bar charts with variable width $(C,0,Q,Q)$, and scatterplots with rectangular glyphs (Q,Q,Q,Q).

As another example, a scatterplot of two quantitative variables q_i and q_j corresponds to the mapping $(x,y,w,h) = (q_i,q_j,K,K)$, where the "points" in the scatterplot are actually small $K \times K$ squares. In shorthand, this is (Q,Q,K,K) (Figure 3.6, upper right).

The chart (C,C,Q,Q) (Figure 3.6) might be called a "table of 2D bars" with $(x,y,w,h) = (c_i,c_j,q_k,q_l)$. Note that, when we write (C,C,Q,Q), we imply that each C and each Q is distinct (i.e., has a different subscript). In contrast, a chart of the form $(x,y,w,h) = (c_i,c_i,q_j,q_j)$ redundantly encodes c_i and q_j twice, plotting squares along a diagonal. Such a redundant chart is not captured by our shorthand notation, but is not so interesting anyway.

We allow independent variables to be *nested* in the computation of x or y. For example, if the categorical variables are stored as positive integers (that is, $c_i \in C_i = \{1,2,\ldots,|C_i|\}$ for all i), and we wish to plot a grouped bar chart where each group of bars corresponds to a value of C_i, and each bar within a group corresponds to a value of C_j, we might define $x = c_i + \frac{1}{|C_j|}c_j$ (ignoring margins or scaling factors). We then say that C_j is nested within C_i, and denote such a grouped bar chart as $(C_iC_j,0,K,Q_k)$ or simply $(CC,0,K,Q)$. Another example of a chart involving nesting is parallel bar charts (C_iC_j,C_k,K,Q_l) (see (CC,C,K,Q) in Figure 3.6), where C_i and C_k establish the columns and rows, respectively, of a grid, and within each cell of the grid is a $(C_j,0,K,Q_l)$ bar chart. This nesting of variables corresponds directly to the notions of "hierarchical axis" of Mihalisin et al. (Mihalisin *et al.*, 1991), "dimensional stacking" in LeBlanc et al. (LeBlanc *et al.*, 1990), and also to the "cross" operator in the table algebra of Polaris (Stolte *et al.*, 2002).

The last ingredient in our design space allows rectangles to be *stacked* along a categorical variable C_i; we denote this stacking with C_i^S. For example, a stacked bar chart described with (C_1,C_2^S,K,Q_1) has the y of each rectangle equal to the sum of the heights of rectangles whose tuples have smaller values of c_2.

To more precisely define our design space, we note that, in our shorthand notation, each of x and y may be given by any of the following: 0 (zero), Q, C, C^S, or a sequence of one or more C followed by a final Q, C, or C^S. In addition, each of w and h may be given by K (a constant)

or Q. For example, the chart $(C_1C_2Q_1, C_3C_4Q_2, Q_3, Q_4)$ would be a table of columns (C_1), sub-columns (C_2), rows (C_3) and sub-rows (C_4), within which each cell contains a (Q_1 versus Q_2) scatterplot of 2D bar glyphs encoding Q_3 and Q_4 in their width and height, respectively. In theory, it is possible to automatically enumerate the possible charts in this design space, however there are (countably) infinitely many unless some limit is imposed on the nesting depth.

Our design space could be extended in a few straight-forward ways. Each rectangle, in addition to having a position, width, and height, could also be given a variable color α, yielding 5 parameters (x, y, w, h, α). A scatterplot of rectangular glyphs with variable width, height, and color is shown in the lower right corner of Figure 3.6. Another extension would add a text label t to each rectangle, to show a numeric value or a string. For example, the chart $(x, y, w, h, t) = (C_1, C_2, Q_1, K, Q_1)$ would be a table of horizontal bars, where the width and text label of each bar redundantly show the same quantitative value, allowing a user to quickly scan for interesting values (by looking at the bars) and then read precise values using the text labels: this kind of chart might be called a "back bar chart", since the bars appear behind the text labels.

Figure 3.7 shows the charts implemented in our prototype. In addition to the charts based on plotting rectangles from our design space, we added a "text table" chart to enable displaying precise numeric quantities.

3.6 Types of connections

The groundwork laid by the previous sections allows us to now analyze the types of connections that we may want to display between charts. We distinguish two types of connections: those between corresponding *tuples*, and those between corresponding *axes*.

To display connections between corresponding *tuples*, we must have the same tuples in the two charts. This is of course not the case if the data has been aggregated differently in the two charts. For example, in Figure 3.8, the text table and grouped bar chart show some data set $d : Region \times Year \mapsto \ldots$, whereas the charts along the top row show $d^{[Year]}$ (i.e., aggregated

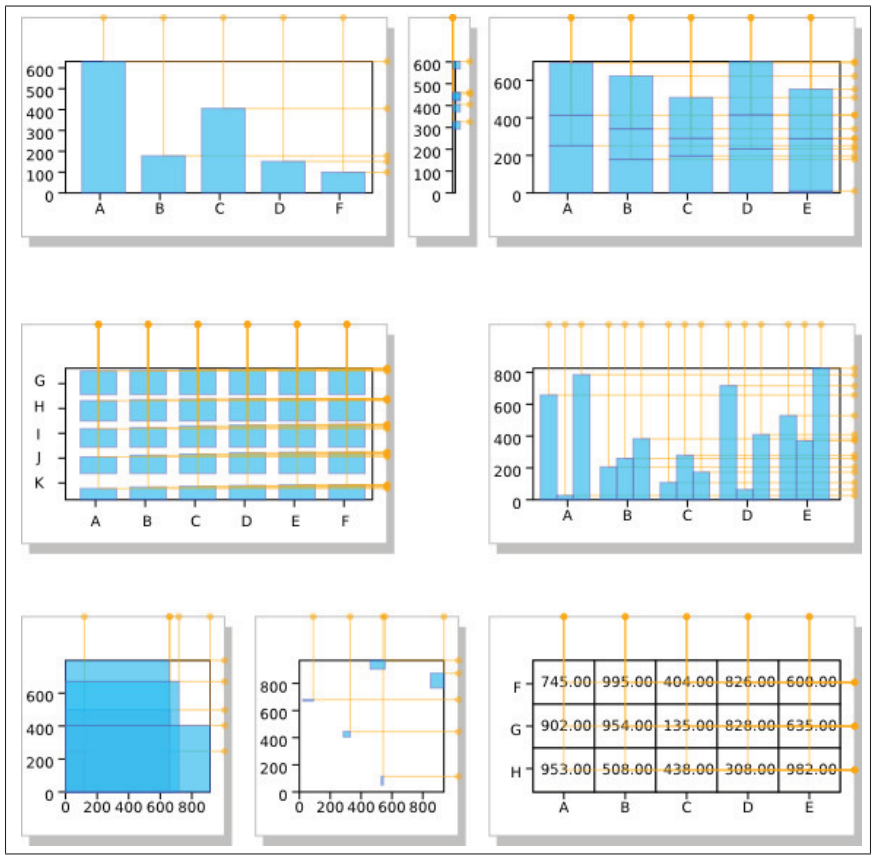


Figure 3.7 Examples of the charts supported by our prototype: bar chart; 1D axis plot; stacked bar chart; table of bars; grouped bar chart; a scatterplot variant where the location of tuples, instead of being shown with points, is shown by the upper-right corner of overlapping rectangles; a scatterplot whose marks are rectangular glyphs with variable width and height; text table. Normal scatterplots are also supported. Normally, the orange lines within each chart are only drawn in the direction of neighboring connected charts; in these examples, we assume a neighboring chart is situated to the north and to the east, so the orange lines extend up and right.

over the Year variable, leaving Region as the only distinguishing categorical variable), and the bar chart in the left bottom corner shows $d^{[Region]}$ (leaving Year as the only distinguishing categorical variable).

The datasets d, $d^{[Year]}$, and $d^{[Region]}$ have 30, 5, and 6 tuples, respectively, and it would not make sense to try to connect the 5 tuples in a chart showing $d^{[Year]}$ to the 6 tuples in some other chart showing $d^{[Region]}$. If, however, the data in the two charts is aggregated the same way, we can connect the tuples in one chart to the tuples in the other. There are two sub-cases to consider:

connecting tuples through a Q axis, and connecting tuples through a C axis. Two examples of connections through Q are shown in the top row of Figure 3.8, where the Q axes are either the same, or different (notice how the connections between different Q axes is similar to the connections in parallel coordinates). To connect through a C axis, if it is the same C axis in both charts, we recommend simply connecting the values of the C axes (see Figure 3.8), since there are presumably multiple tuples for each value of the C axis, and connecting all tuples would only create more connective lines or curves without any benefit.

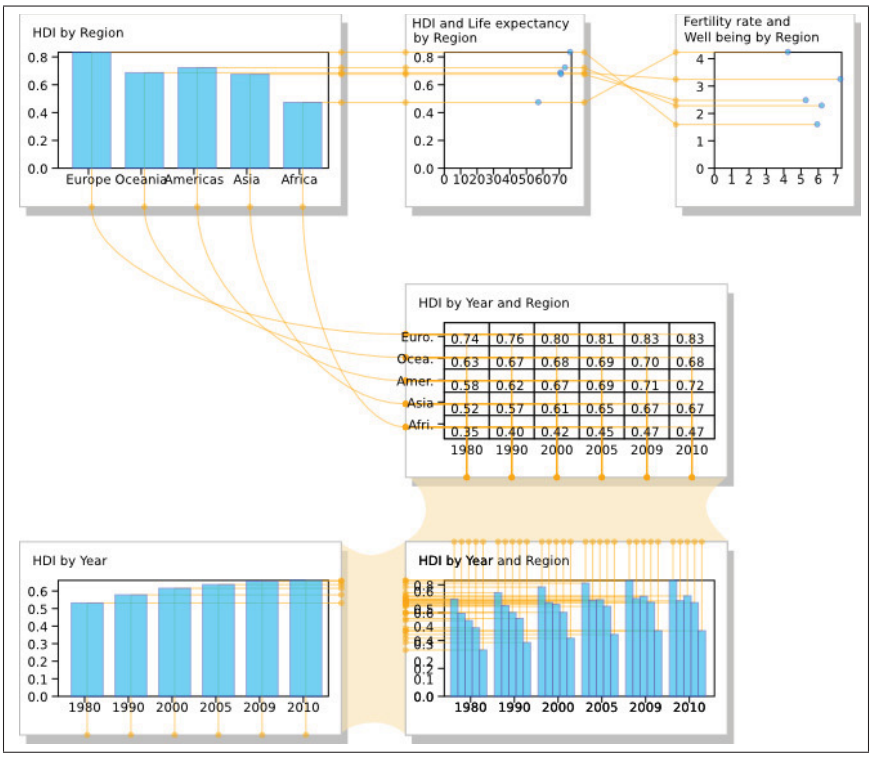


Figure 3.8 Examples of different connections between charts. Top left to top center: tuples are connected across the same Q axis (Mean HDI). Top center to top right: tuples are connected across different Q axes (Mean HDI and Fertility rate). Top left to middle row: values of a C axis (Region) are connected. Middle row to bottom center: an entire C axis is connected to another C axis (Year). Bottom left to bottom center: an entire Q axis is connected to another Q axis (Mean HDI).

If the data in the two charts is not aggregated in the same way, then it is not possible to connect tuples. However, it is still possible to connect axes. Two examples are shown in Figure 3.8, one for a C axis, and another for a Q axis.

Note that we intentionally do not draw connections involving a C variable nested within another C variable. We also do not draw connections involving a Q variable mapped to rectangle width or height, unless the x or y, respectively, of the rectangle is equal to zero. For example, the Q variable in a $(C, 0, K, Q)$ bar chart can be connected to another Q variable in another chart (as shown in Figure 3.8, top left to top center), because the y of rectangles in that chart is equal to zero. However, in a scatterplot of glyphs (Q_1, Q_2, Q_3, Q_4), we do not draw connections from Q_3 or Q_4 to an axis in another chart. We did consider ways of graphically depicting such connections, but in the end decided that they would be too confusing.

3.7 ConnectedCharts prototype

Our prototype is implemented in JavaScript using Data-Driven-Documents (D3) (Bostock *et al.*, 2011), a graphical toolkit that allows data to be bound to graphical elements like SVG shapes. In most of our charts, the graphical "marks" are rectangles, so defining a chart with D3 is done by mapping the data to the rectangle's attributes. Smoothly animated transitions between different kinds of charts, although not currently implemented, would be straight forward to do, since the attributes of the rectangles could simply be animated from one chart to another.

Figure 3.7 shows most of the implemented charts. These charts are positioned within a 2D space that can be zoomed and panned. Each chart can be dragged, cloned, and deleted, and hovering over rectangles in a chart causes related elements to highlight. A popup menu (Figure 3.9) allows the (independent and dependent) variables of the chart to be edited.

Given a set of independent and dependent variables associated with a chart, a small set of rules determine which mappings are appropriate for the chart. The following table lists these rules: the first column is the number of independent variables, the 2nd column is the number of dependent variables, and the third column lists the appropriate chart types. Notice that, in every case, the number of independent variables is equal to the number of Cs appearing in the mapping, and the number of dependent variables is equal to the number of Qs.

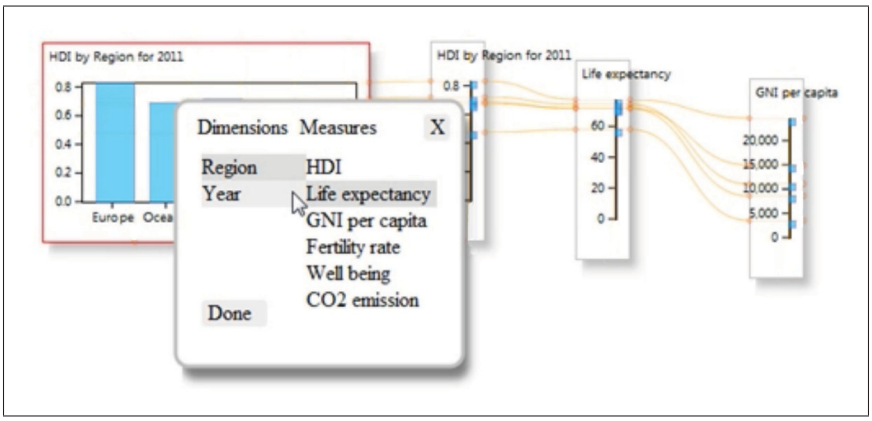


Figure 3.9 A popup menu allows the variables associated with a chart to be changed, which can cause the type of chart to change.

Table 3.1 Rules for mapping independent and dependent variables to a chart.

$*$	1	1D axis $(0,Q,K,K)$
1	1	bar chart $(C,0,K,Q)$
$*$	2	scatterplot (Q,Q,K,K) overlapping rectangles $(0,0,Q,Q)$
1	2	bar chart whose bars have variable width and height $(C,0,Q,Q)$
$*$	3	scatterplot of glyphs with variable height (Q,Q,K,Q)
$*$	4	scatterplot of glyphs with variable width and height (Q,Q,Q,Q)
2	1	stacked bar chart (C,C^S,K,Q) grouped bar chart $(CC,0,K,Q)$ table of bars (C,C,K,Q) text table $(x,y,t) = (C,C,Q)$

As an example, if there is 1 independent variable and 2 dependent variables associated with the chart, then the rules for (*,2) and (1,2) would apply, allowing for a scatterplot, overlapping rectangles, or a bar chart whose bars have variable width and height. Selecting the chart and hitting the spacebar allows the user to cycle through these 3 chart types.

A typical workflow could start with a single chart, which could be cloned with a shift-drag. Then, in the cloned chart, the user could choose new variables for each axis with the right-click

menu, switch chart types with the spacebar, then connect the charts with a drag and drop from one chart's axis to the other.

3.7.1 Data

The dataset shown in Figures 3.1, 3.8, 3.10, 3.11, and 3.12 is based on the Human Development Report by the United Nations Development Programme (UNDP) (http://hdr.undp.org/en/reports/global/hdr2011/download/). In this dataset, countries were grouped by Region to form a first independent variable (C), and Year was used as a second independent variable. Dependent variables (Q) include HDI (Human Development Index), GNI (Gross National Income) per capita, Life expectancy, and Fertility rate.

3.7.2 Examples

Figure 3.10 illustrates one way that ConnectedCharts can be used to successively analyze a dataset, and how the connections reveal the history of the user's analytical process. The configuration in Figure 3.10 starts with a text table of Human Development Index (HDI) for each region over a number of years. The table of bars makes it easier to perceive some overall patterns. This table is then "split", so to speak, into two stacked bar charts, one for each combination of one categorical variable "slicing" the other. Finally, ordinary bar charts are produced (by aggregating), to emphasize the totals, for easy comparison and detection of trends. Each step of the process is recorded in the form of the connections, making it easy to follow the same exploratory path.

Figure 3.11 illustrates another analytical process. Finally, Figure 3.12 shows that entire PCPs and SPLOMs can be instantiated, by appropriately connecting together the relevant charts. Furthermore, many variations and mixtures of these can be created, incorporating bar charts and their variants, as the user sees fit.

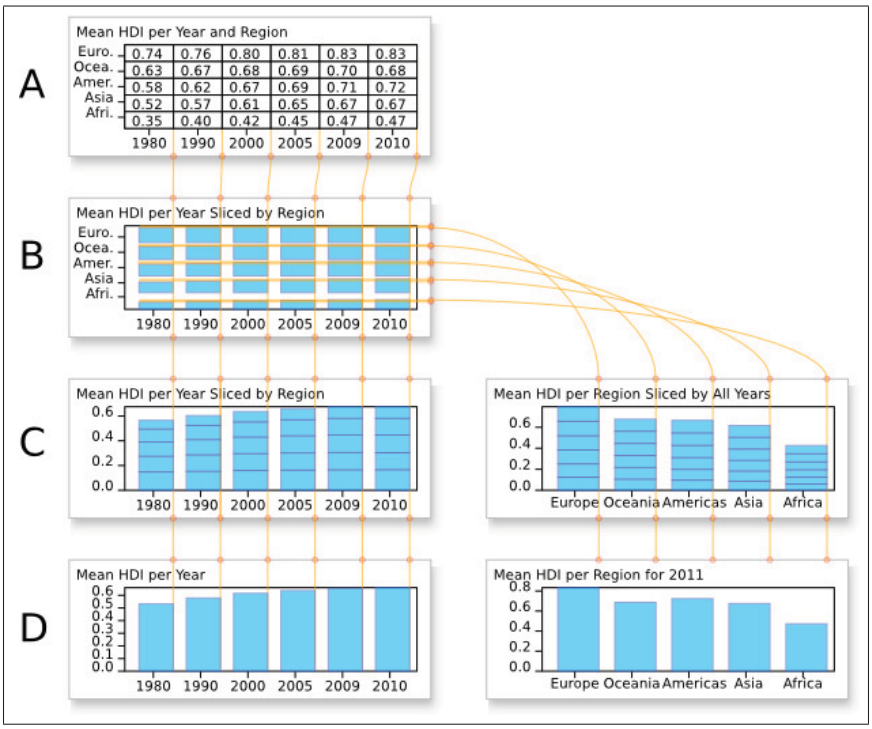


Figure 3.10 A: HDI for each year and region. B: The same data, shown with parallel bars instead of numbers, to enable quick, visual inspection. C: The same bars stacked along years, and along regions. D: Equivalent bar charts, aggregated by year, and by region.

3.8 Generalizing ARGOIs

Claessen and van Wijk (Claessen and van Wijk, 2011) introduced the notion of an Attribute Relation Graph of Interest (Claessen and van Wijk, 2011) (ARGOI), where each node is a variable, and each edge is a pair of variables (each edge corresponding to a scatterplot or pair of parallel coordinate axes). Our ConnectedCharts can be modelled by a graph analogous to the ARGOI, but in our case the graph is a hypergraph (i.e., a graph with hyperedges, each of which can be incident on many nodes), since in our case each chart would be a hyperedge, and each chart may display multiple variables. We call this graph the hyper-ARGOI.

The dual of the hyper-ARGOI would also be useful for modeling ConnectedCharts: in this case, each node would be a chart, and each hyperedge would be a variable that is possibly displayed in many charts.

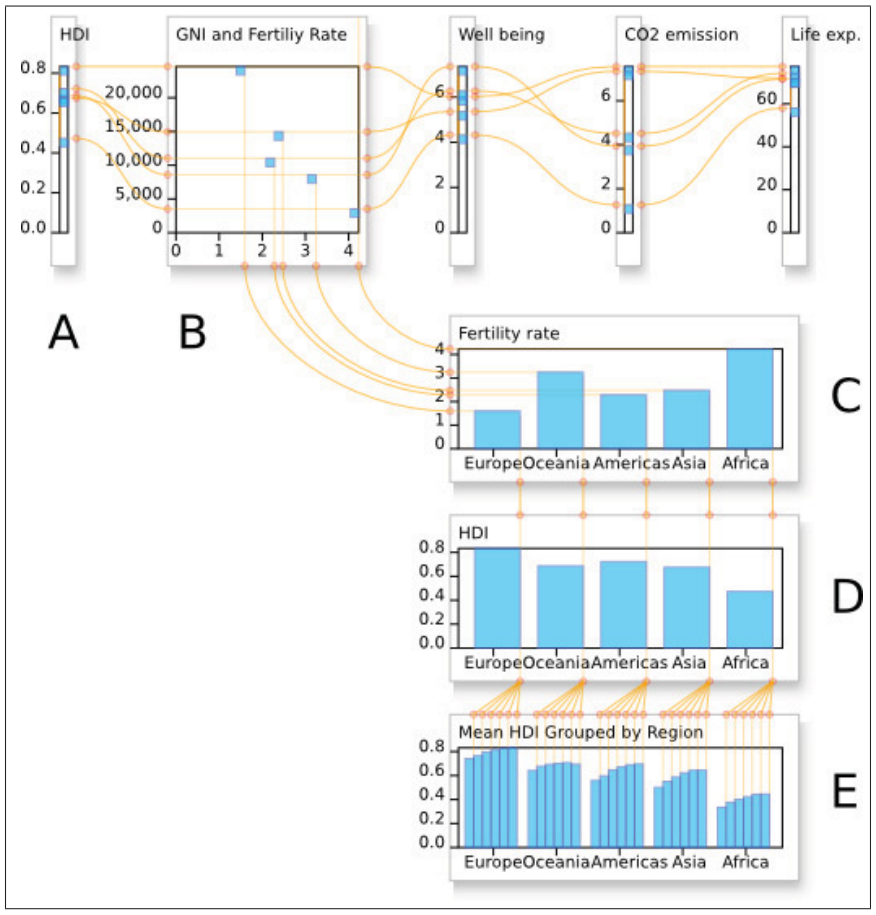


Figure 3.11 The top row of charts show a parallel-coordinates-style comparison of variables. Connections between A and B show HDI versus GNI. B: a scatterplot of GNI (vertically) and fertility rate (horizontally) reveals a tuple with low GNI and high fertility. C: connections between B and C reveal the tuple to be Africa, which has the highest fertility rate. D: Africa also has the lowest HDI. E: In fact, Africa's HDI has been the lowest over all years.

Finally, a third way to model a ConnectedCharts visualization would be with a bipartite graph: one set of nodes for variables, another set of nodes for charts, and edges between a variable and a chart when the former appears in the latter. Notice that all three structures, the hyper-ARGOI, its dual, and the bipartite graph, encode the same information and are isomorphic.

Note that none of these graphs correspond directly to the connections actually *displayed* by ConnectedCharts, since ConnectedCharts only displays the connections established by the user through interaction, ensuring that excessive clutter can be avoided.

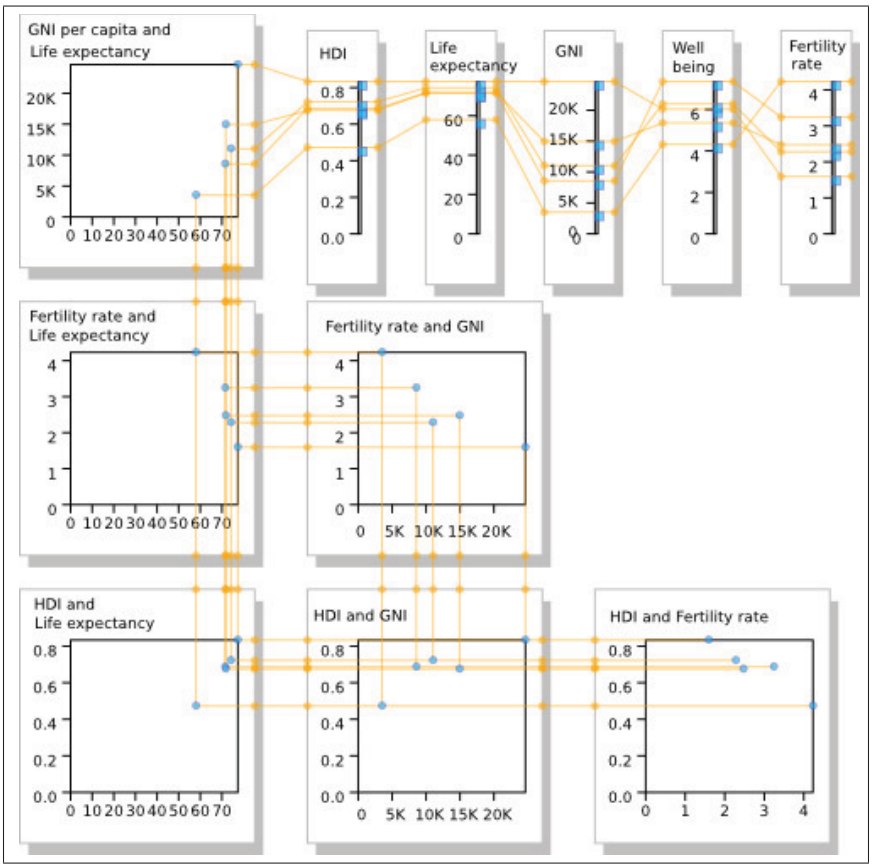


Figure 3.12 Both a scatterplot matrix and parallel coordinates plot can be instantiated as a ConnectedChart, as can a hybrid mixture of them, shown here.

3.9 Conclusions and future directions

We have presented a technique for linking together charts using line segments and curves that can be applied to many different kinds of charts, including scatterplots, bar charts, and variants. We have also shown how multiple charts may be assembled to create parallel coordinate plots, scatterplot matrices, and mixtures of these with each other or other types of charts. ConnectedCharts allows for tuples in different charts to be aggregated to different levels, and for charts to positioned freely in a 2D space. Because the ConnectedCharts technique does not require that all possible connections be displayed, the connections that are displayed can be a compromise between showing all possible links and showing only links in response to brushing.

We have also presented a formal description of a design space of charts based on plotting rectangles, and shown how this design space encompasses many useful kinds of charts, includ-

ing scatterplots of rectangular glyphs and Gantt charts. We have also distinguished different kinds of connections (section 3.6) that may be displayed between charts, and pointed out that Claessen and van Wijk's (Claessen and van Wijk, 2011) notion of ARGOI can be generalized to a hyper-ARGOI.

For future work, we are interested in implementing improvements to the user interface of our prototype. For example, a popup menu based on the FlowVizMenu (Viau *et al.*, 2010) might allow for a gestural-style of interaction with the charts, to clone and modify them. Macros might also be implemented to allow the user to quickly instantiate entire PCPs or SPLOMs without having to construct them one chart at a time. Techniques for enhancing parallel coordinates, or reducing clutter, such as edge bundling, might be used to similarly enhance ConnectedCharts.

A more ambitious project would be to automatically instantiate a set of ConnectedCharts, based on some arbitrary dataset that has been read in by the system: which charts would be most informative to the user, and with which connections between them?

CHAPTER 4

FLOWVIZMENU, P-SPLOM AND AD-LAYOUT: HYBRID VISUALIZATIONS FOR NETWORK EXPLORATION

Christophe Viau[1], Michael J. McGuffin[1], Yves Chiricota[2], Igor Jurisica[3],
[1]Department of Software and IT Engineering, École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3
[2]Department of Mathematics and Computer Science, Université du Québec à Chicoutimi,
555 boulevard de l'Université, Chicoutimi, Québec, Canada G7H 2B1
[3]Ontario Cancer Institute, Toronto Medical Discovery Tower,
101 College Street, Toronto, Ontario M5G 1L7, Toronto, Canada M5G 1L7
Published in IEEE Transactions on Visualization and Computer Graphics (Proceedings of InfoVis 2010), vol. 16, n. 6, November/December 2010, p. 1100-1108

4.1 Abstract

A standard approach for visualizing multivariate networks is to use one or more multidimensional views (for example, scatterplots) for selecting nodes by various metrics, possibly coordinated with a node-link view of the network. In this paper, we present three novel approaches for achieving a tighter integration of these views through hybrid techniques for multidimensional visualization, graph selection and layout. First, we present the FlowVizMenu, a radial menu containing a scatterplot that can be popped up transiently and manipulated with rapid, fluid gestures to select and modify the axes of its scatterplot. Second, the FlowVizMenu can be used to steer an attribute-driven layout of the network, causing certain nodes of a node-link diagram to move toward their corresponding positions in a scatterplot while others can be positioned manually or by force-directed layout. Third, we describe a novel hybrid approach that combines a scatterplot matrix (SPLOM) and parallel coordinates called the Parallel Scatterplot Matrix (P-SPLOM), which can be used to visualize and select features within the network. We also describe a novel arrangement of scatterplots called the Scatterplot Staircase (SPLOS) that requires less space than a traditional scatterplot matrix. Initial user feedback is reported.

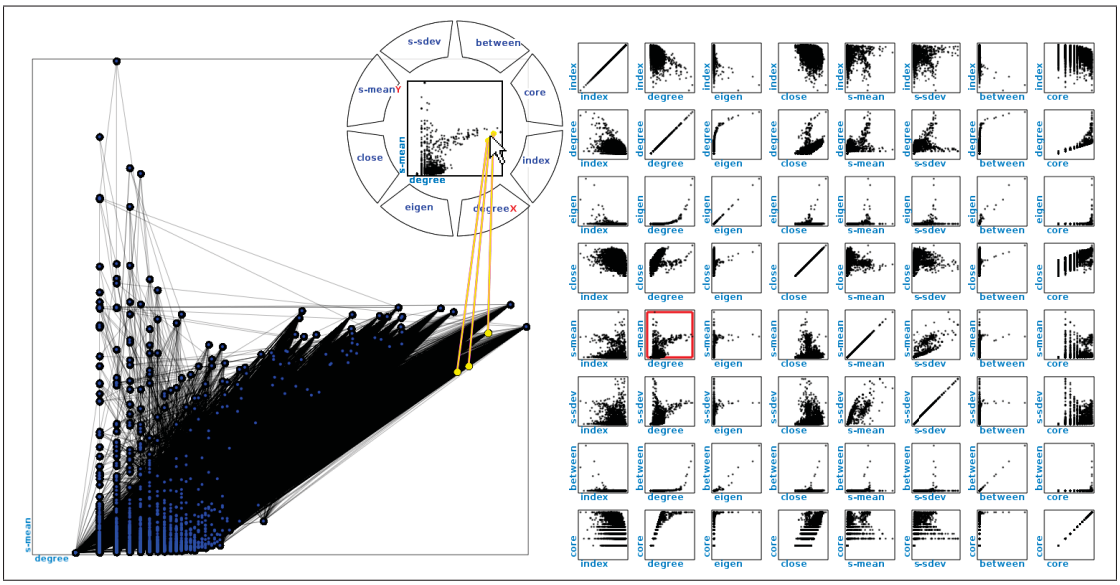


Figure 4.1 The FLowVizMenu controls the AD-layout and the selected SPLOM is highlighted in the P-SPLOM.

4.2 Introduction

The most common approach for visualizing a network is a node-link diagram, for which there are many layout algorithms (di Battista *et al.*, 1998). Unfortunately, some networks are so complicated that it may be impossible to give them a layout that makes the most important nodes, their connections, and their immediate neighbors clearly visible. For example, in our own work with biological networks, we often have individual nodes with over 300 neighbors; just laying out these neighbors in a clear way is challenging. In addition, a single, automatically generated layout may not be appropriate for all situations, leading to users often manually adjusting and repositioning certain nodes in the node-link diagram. When the nodes of interest to the user are not immediately visible, some indirect means of searching and selecting them is required. For example, a spreadsheet interface (such as that in (Brown *et al.*, 2009; Bezerianos *et al.*, 2010a)) might list all nodes and allow them to be sorted by name, degree, clustering coefficient, etc. and selected. Another possibility is to have a scatterplot, of clustering coefficient versus degree for example, within which the user may select nodes (Chiricota *et al.*, 2004).

Taking this idea further, for any given network, we could compute various metrics associated with each node (such as degree, clustering coefficient, betweenness centrality, etc.) and we

may also have various attributes associated with each node (protein name, biological function, cellular localization, etc.) Such multidimensional data motivates the use of standard multidimensional visualization techniques, in particular: Scatter Plot Matrix (SPLOM) (Hartigan, 1975) and Parallel Coordinates Plot (PCP) (d'Ocagne, 1885; Inselberg, 1985; Wegman, 1990). Recent research by Bezerianos *et al.* (2010a) takes such an approach, using a SPLOM to visualize a multivariate graph. In their system, the SPLOM serves as a kind of overview of the data, and is coordinated with a single zoomed-in scatterplot that serves as the focus. Links between nodes are drawn on top of the scatterplot to reveal the graph's structure. A ScatterDice-style interface (Elmqvist *et al.*, 2008) allows for transitions of dimensions through 3D rotation.

Our work further explores the design space of network visualizations that incorporate multidimensional visualization techniques. We have developed a novel combination of a node-link diagram, a SPLOM, and PCPs (Figures 4.1 and 4.2). The layout of our node-link diagram can be modified manually, or through force-directed layout, or attribute-driven layout, or a mixture of these. Our SPLOM and PCPs are combined into a Parallel Scatterplot Matrix (P-SPLOM) that affords fluid transition between scatterplots, 3D PCPs, and normal (2D) PCPs using 3D rotation. We also present a novel popup widget that enables rapid, fluid gestures to select within scatterplots and modify the layout of the node-link diagram. Together, these techniques provide a greater variety, and a tighter integration, of visualizations of the network than has been previously possible.

Our contributions are (1) a novel popup widget for manipulating multidimensional visualizations called the FlowVizMenu; (2) a technique for mixing attribute-driven layout with force-directed and manual layout of nodes within a node-link diagram; (3) a novel integration of SPLOMs, PCPs, and 3D PCPs called the P-SPLOM; (4) an investigation of the tradeoffs of different orderings of the axes within a P-SPLOM; and (5) a novel arrangement of scatterplots called the Scatterplot Staircase (SPLOS). We also report initial user feedback.

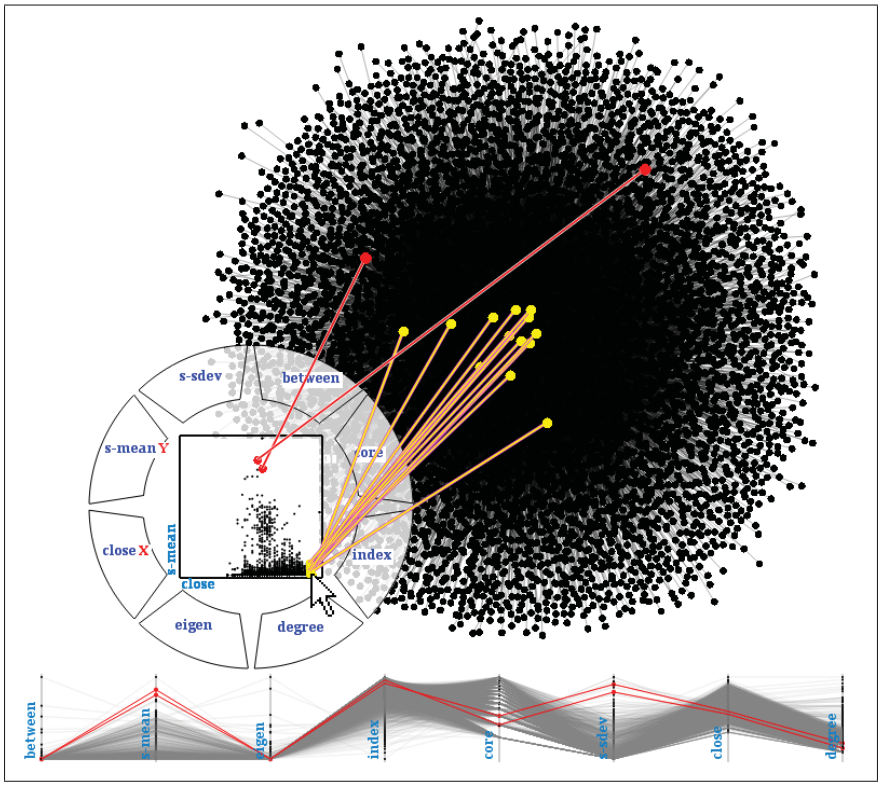


Figure 4.2 Selecting (in red) and brushing (in yellow) within the FlowVizMenu's scatterplot shows linking with the node-link diagram. Along the bottom of this screenshot, the P-SPLOM has been converted to a single row PCP, and selected nodes are shown as red polylines.

4.3 Related Work

Recent work on multivariate graph visualization includes (Wattenberg, 2006; Aris and Shneiderman, 2007; Bezerianos *et al.*, 2010a). Of these, the most similar to our current work is GraphDice (Bezerianos *et al.*, 2010a), in which there are two main views: the SPLOM, and a single zoomed-in scatterplot. The nodes in the scatterplot are always positioned according to the selected dimensions, such as degree, centrality, etc. — this could be called attribute-driven layout. Two special dimensions are the *x* and *y* positions of the nodes in a force-directed layout. Selecting these *x* and *y* positions as the dimensions for the scatterplot results in a "scatterplot" showing the force-directed layout of the graph. Our work differs from GraphDice in several respects: we allow mixing of manual, force-directed, and attribute-driven layout; we allow the SPLOM to be rearranged in many different ways, as discussed in Section 4.7; our SPLOM

can also be rotated in 3D to yield 3D parallel coordinate plots or normal (2D) PCPs. Our FlowVizMenu also differs from the ControlMenu used in (Bezerianos *et al.*, 2010a) to rotate dimensions, as discussed in Section 4.5.

Our system uses parallel coordinate plots (Inselberg, 1985; Wegman, 1990). 3D variants of PCPs have been proposed before, including (Fanea *et al.*, 2005; Johansson *et al.*, 2006), and, most relevant to our work, (Falkman, 2001; Rübel and et al, 2006). The "Cube" of Falkman (2001), and 3D parallel coordinates of Rübel and et al (2006), are both essentially sequences of parallel planes, with each plane containing a scatterplot. This can be equated with taking normal 2D PCPs and replacing each axis with a scatterplot on a plane. The links connecting corresponding points on consecutive planes could be compared to the links between planes in VisLink (Collins and Carpendale, 2007). We will refer to this 3D variant of PCPs as simply 3D PCPs. The P-SPLOM we present in this work is a novel unification of SPLOMs, PCPs, and 3D PCPs, achieved through a simple 3D rotation of the scatterplots.

Hybrid multidimensional visualizations have also been proposed before, including combinations of PCPs and scatterplots (Qu *et al.*, 2007; Yuan *et al.*, 2009; Steed *et al.*, 2009; Holten and van Wijk, 2010) (see also (Xu *et al.*, 2007) for a similar idea) and parallel coordinates displayed within a matrix (Albuquerque *et al.*, 2009). Our P-SPLOM contributes one additional hybrid multidimensional visualization, with tradeoffs compared to previous work. For example, the scatterplot-PCP hybrid of (Holten and van Wijk, 2010) displays both the points of scatterplots and the polylines of a PCP at the same time, but at an increased cost in screen space. On the other hand, our P-SPLOM allows users to only view scatterplots, or only view a PCP, or view a 3D PCP showing both the points within each (partially rotated) scatterplot and the polylines of the PCP.

4.4 Metrics used

In principle, any number of attributes (protein name, biological function, cellular localization, etc.) associated with the nodes could be used as dimensions in our system. Our system does not require the network to have any attributes, however, as several metrics are defined and com-

puted for each node from the network structure itself. These include: `index`: a unique integer identifying the node (although this dimension often has little meaning, its use within a scatterplot can help de-occlude points to reveal a distribution); `degree`: the number of neighbors of the node; `between`, `close`, `eigen`: the betweenness, closeness, and eigenvector centralities, respectively[1]; `cluster`: the clustering coefficient; `core`: the shell associated with the *k*-core decomposition of the network (Wuchty and Almaas, 2005); `s-mean` and `s-sdev`: the average and standard deviation, respectively, of the *strength* (as defined in (Auber *et al.*, 2003)) of the edges adjacent on the node.

4.5 The FlowVizMenu

The Human Computer Interaction (HCI) community has proposed several popup widgets that afford a gestural style of interaction (Callahan *et al.*, 1988; Kurtenbach and Buxton, 1993; Pook *et al.*, 2000; Guimbretière and Winograd, 2000; Kurtenbach *et al.*, 1999), some of which have been applied to interactive visualization (McGuffin and Jurisica, 2009). Popup widgets have several advantages: they require no screen space when not in use; they eliminate the need to travel back and forth between a work area and menu bars or panels of widgets located in the periphery; and they can also be invoked by holding down a button, keyboard key, or stylus tip, resulting in kinesthetic feedback that helps avoid mode errors (Sellen *et al.*, 1992) and helps to integrate the selection of arguments into a single phrase or gesture (Buxton, 1986).

The FlowVizMenu[2] is a novel popup widget that contains a multidimensional visualization, and allows dimensions to be selected through outward and inward motions. Figure 4.3 shows the variant we implemented, which displays a single scatterplot at a time. The menu is popped up with a keyboard key, after which the pointing device may brush over the scatterplot (Figure 4.2). The user may also select dimensions for the scatterplot by stroking outward or inward, for *x* and *y* respectively. This allows the user to quickly switch between, and compare, scatterplots. The outward-inward motions were inspired by those in the original FlowMenu (Guimbretière and Winograd, 2000). These quick, continuous gestures have been found to

[1] http://en.wikipedia.org/wiki/Centrality

[2] We initially considered calling this widget the MatchWheel, because MatchWheel + HotBox form a chiasm with MatchBox + HotWheel, two well known toy car brands.

yield good selection performance (Guimbretière, 2003). When the pointer crosses over the name of a dimension, a smooth transition occurs, showing the new dimension rotate into the old dimension's place. The FlowVizMenu's rotation transition is similar to that in (Elmqvist *et al.*, 2008; Bezerianos *et al.*, 2010a) except that we use an orthographic projection instead of a perspective projection, to maintain the position of the points along the unchanging axis. Furthermore, when the pointer crosses over the name of a dimension, the user may "scrub" to control the progression of the transition. This scrubbing functionality is comparable to the way a Control Menu (Pook *et al.*, 2000) works, in that it uses the mouse drag to control a continuous parameter. Thus, the FlowVizMenu can be seen as a hybrid between a FlowMenu and a Control Menu that furthermore incorporates a visualization. Teoh *et al.* (2004) also proposed a radial layout of small visualizations to be used for navigation, however their radial layout is not within a popup widget.

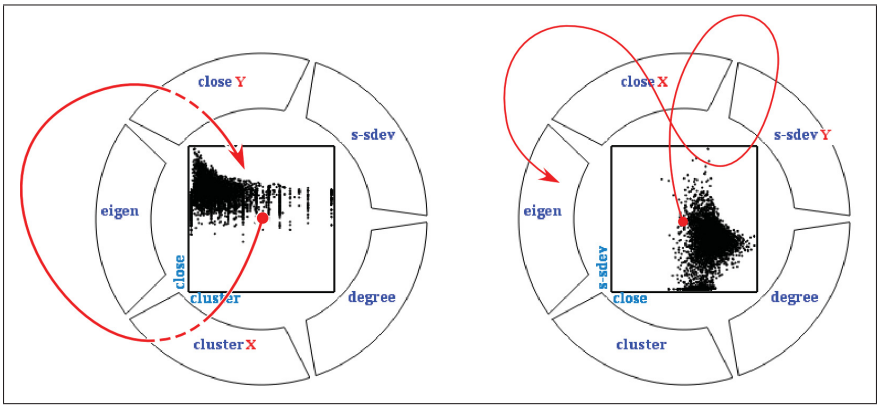


Figure 4.3 Our implemented FlowVizMenu. *Left*: the `cluster` × `close` scatterplot was selected by stroking outward through the dimension for the horizontal axis and then inward through the dimension for the vertical axis. As the user strokes over the dimension names (dashed parts of the red arrow), the scatterplot rotates away from the previously selected axis and towards the new one. The user may "scrub" over the dimension name to slow down or replay this rotation. *Right*: Drawing a figure-8 gesture allows the user to rotate between two different scatterplots, in this case toggling between `s-sdev` and `eigen` on the vertical axis while maintaining `close` on the horizontal axis. This allows for comparison of the two scatterplots within a relatively small space.

We can also compare our implemented FlowVizMenu with the control menu used in GraphDice (Bezerianos *et al.*, 2010a). Both are used to transition between dimensions in a scatterplot, and both allow for scrubbing. However, because GraphDice's control menu only makes use of outward motions, it contains 2 copies of every dimension: one for the scatterplot's horizontal axis, and another for the vertical axis. Furthermore, the menu items in GraphDice's control menu only cover half a circle. The end result is that the dimensions in our FlowVizMenu each cover an angle that is four times larger, enabling easier and faster selection. Our FlowVizMenu also differs in that it *contains* a visualization, whereas GraphDice's control menu is used to control an underlying visualization. As shown in Figure 4.2, this means the user can perform brushing and linking for coordination with other views without leaving the FlowVizMenu.

In our FlowVizMenu, each repeated outward-inward motion will normally replace the two dimensions in the scatterplot with new dimensions. Hence, a repeated figure-8 motion (Figure 4.3, right) cycles between two scatterplots. However, it also occurred to us that repeated outward-inward motion could be useful for *accumulating* dimensions. In our implementation, holding down the shift key during motions causes the dimensions to be accumulated along the axes using principle component analysis (PCA). For example, holding down Shift, and moving out through dimension A, in through B, out through C, and in through D, will cause the system to compute a PCA projection from A $\times$ C to the horizontal axis, and a separate PCA projection from B $\times$ D to the vertical axis.

Another way that dimensions could be "accumulated" during repeated motions is in building up a PCP. We designed (but did not implement) this idea, as shown in Figure 4.4, where the accumulated dimensions result in a 1D histogram, then a 2D scatterplot, then a 3- (or more) axis PCP. Yet another design (also not implemented) is shown in Figure 4.5.

We hypothesized that because our implemented FlowVizMenu contains only a single scatterplot, it might be usable on a device with a small screen, and useful even without any additional views of the data. We have prototyped a FlowVizMenu with JavaScript that can be used on Apple's iPhone and iPod touch devices (Figure 4.6). The size of the menu items and the surrounding space were chosen to reduce finger occlusion and to be able to move around the

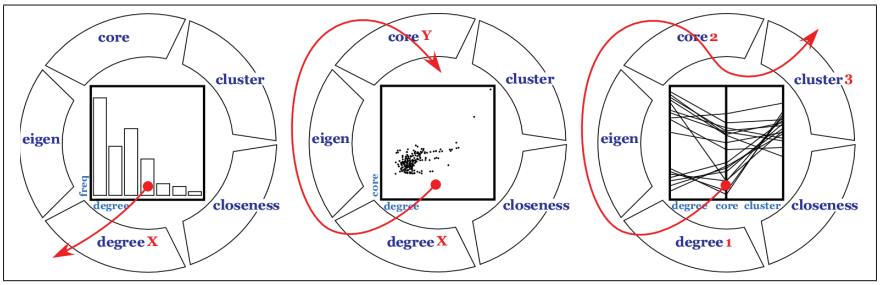


Figure 4.4 Mock-up of alternative design: selecting a single dimension causes a histogram to be displayed (*Left*), selecting a second dimension causes a scatterplot to be displayed (*Middle*), selecting additional dimensions transitions to a parallel coordinate plot (*Right*).

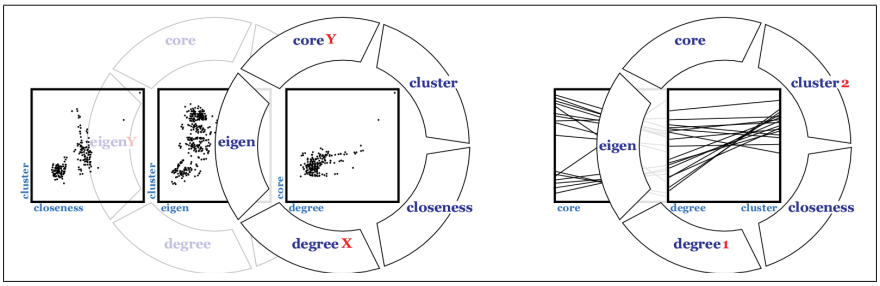


Figure 4.5 Mock-up of a 2nd alternative design: Here, rather than "containing" a visualization, the FlowVizMenu is used to build up a visualization underneath, depositing or editing each piece of the visualization. *Left*: the FlowVizMenu is popped up over each cell of a matrix of scatterplots to choose the desired axes. *Right*: the FlowVizMenu is used to choose the axes within a parallel coordinate plot.

menu and scrub with ease. Such a widget might be combined with functionality in (Büring *et al.*, 2006) for zooming in on the scatterplot.

In addition to the FlowVizMenu, users can also pop up a hotbox (McGuffin and Jurisica, 2009) to manually reposition nodes or manipulate the node-link diagram of the network.

4.6 Attribute-Driven Layout

As already seen, the FlowVizMenu can be used to brush or select nodes that have, for example, both low degree but high betweenness centrality (such nodes could correspond to "bridges" in the network). Such functionality is useful when the node-link diagram is so dense that it is difficult to see individual nodes.

Figure 4.6 Implemented version of the FlowVizMenu for iPhone/iPod touch platform. The dimensions within a single scatterplot can be transitioned with gestures.

This section shows how the FlowVizMenu can also be used to drive an Attribute-Driven Layout (ADL) of the node-link diagram, or of a subset of nodes in the node-link diagram. For this, the user first activates the ADL scatterplot, which is displayed as a square region in the node-link diagram (see left sides of Figures 4.1 and 4.7). Then, when the user pops up the FlowVizMenu and selects two axes, the ADL scatterplot displays red and blue dots showing the corresponding scatterplot positions of selected and unselected nodes, respectively. Any nodes in the node-link diagram that have not been previously locked in place by the user are then animated to lie under their corresponding dots. The user may then select new axes within the FlowVizMenu, causing the positions of the red and blue dots to update immediately, after which the (unlocked) nodes again animate to their new positions under the dots. The use of smoothly animated transitions makes it easier to follow the trajectories of nodes and observe the behavior of groups of nodes during transitions.

As mentioned, with ADL, each unselected node is shown in black, and its corresponding position in the ADL scatterplot is shown with a blue dot. If the node is locked in some position different from the blue dot's position, the blue dot is easy to distinguish against the white background. However, if the node is not locked, then at the end of the animated transition the node moves under the blue dot, which then becomes difficult to see against the black background of the node. This was an intentional design choice, because the blue dot is almost redundant in

that case. The red dots for selected nodes, however, are always visible, enhancing the visibility of selected nodes.

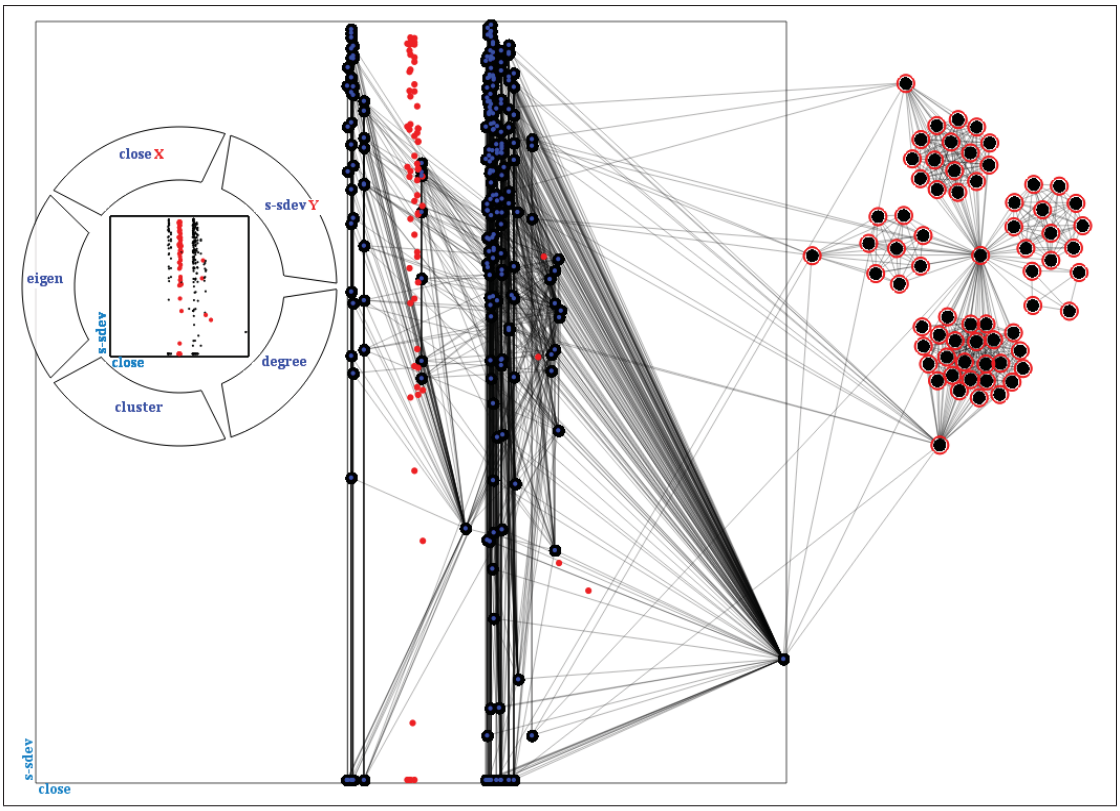


Figure 4.7 A mixture of attribute-driven layout, force-directed layout, and manual layout. At right, four nodes were positioned manually and locked in place. Next, several of their neighbours (also shown at right) were positioned with force-directed layout and then locked in place. The remaining nodes, which were not locked by the user, are positioned according to the ADL scatterplot (left) being manipulated via the FlowVizMenu.

In between invocations of the FlowVizMenu, the user may also manually reposition nodes, lock or unlock certain nodes, and reposition unlocked nodes using force-directed layout. Because ADL only affects unlocked nodes, the user may freely mix manual layout, force-directed layout, and ADL. The result is a hybrid visualization that leverages the fact that scatterplots show two dimensions at once, while also allowing the user to see the edges (drawn as straight line segments) within the graph, and also mix in force-directed or manually repositioned nodes as desired. An alternative approach, where the user has two separate views (one of the node-link

diagram and one scatterplot view), requires more screen space and could require more effort to mentally integrate the two views.

Nevertheless, the user may still sometimes prefer to deactivate the ADL scatterplot, so that they can keep the force-directed node-link diagram separate from, for example, the 2D scatterplot in the FlowVizMenu (which allows for brushing and linking with the node-link diagram). The user is free to choose the approach best suited to their needs.

4.7 Scatterplot matrices (SPLOMs)

The FlowVizMenu only displays a single scatterplot at a time, which can be useful for saving screen space. The user can also transition between different scatterplots within the FlowVizMenu with fast gestures. However, at times the user may want a more global view of the data, and wish to compare several scatterplots simultaneously. For this, we use a scatterplot matrix (SPLOM).

The SPLOM and node-link diagram in our system can be zoomed and panned independently, and the FlowVizMenu's corresponding scatterplot is highlighted in the SPLOM. Brushing and linking can be performed between points in the SPLOM and nodes in the node-link diagram. Furthermore, the user can reduce the SPLOM to a single row of scatterplots, or even a single scatterplot, that can then be zoomed in to a larger size.

Our SPLOM can be ordered in the standard format shown in Figure 4.8 where each row contains only one kind of vertical axis, and each column contains only one kind of horizontal axis. Naturally, this simplifies comparison of scatterplots within the same row or column. Certain systems, such as (Elmqvist *et al.*, 2008), allow the rows and columns of a SPLOM to be reordered, for added flexibility, while still maintaining a single vertical axis within each row and a single horizontal axis within each column. Our system dispenses with this constraint, allowing other arrangements to be investigated. For example, the user may not be interested in comparing scatterplots with a common axis, or in seeing all possible scatterplots. Thus, we developed variants of the standard SPLOM, with different orderings of scatterplots or of their axes, which we now describe.

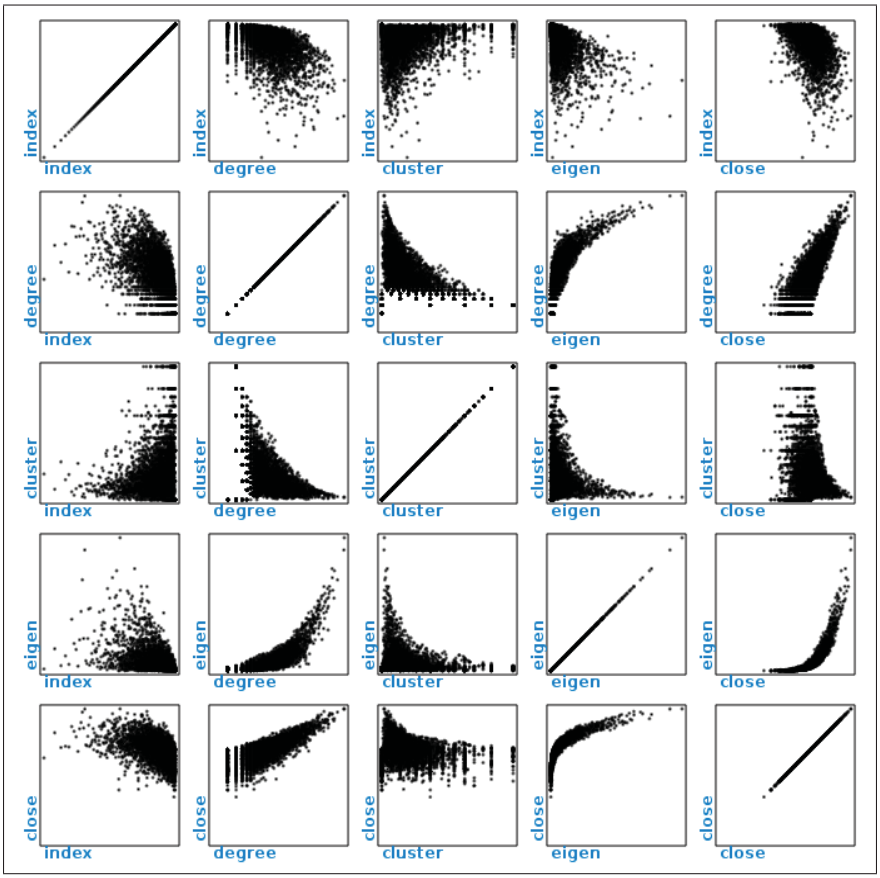


Figure 4.8 A standard SPLOM (scatterplot matrix).

4.7.1 Ranked scatterplot matrix

Our first variant of the SPLOM orders scatterplots according to some ranking or metric of interest. Currently, our implementation of the ranked SPLOM orders scatterplots in descending order of the absolute value of their Pearson correlation coefficient, filling the first row from left-to-right, then the 2nd row, etc. This allows the user to see the scatterplots showing the strongest correlations at the top left of the matrix. It would also be possible to rank scatterplots according to any of the metrics listed in (Wilkinson *et al.*, 2005), or with the "quality-measures" in (Albuquerque *et al.*, 2009), or even allow the user to define their own "ranking" by dragging and dropping scatterplots to insert them at any position in the matrix.

4.7.2 Scatterplot staircase (SPLOS)

This variant, which we call the Scatterplot Staircase or SPLOS (Figures 4.9A and 4.9C), shows only the scatterplots of consecutive pairs of dimensions, arranged in a staircase pattern, such that adjacent scatterplots share an axis along their common edge. To our knowledge, this staircase pattern of scatterplots is novel, but we note that it was inspired by quilts (Watson *et al.*, 2008) (which use a staircase of adjacency matrices) and also by an arrangement of adjacent scatterplots sharing axes on page 135 of (Tufte, 1983). Notice that the scatterplots involved in the SPLOS are located adjacent to the diagonal in the standard SPLOM. If the N dimensions in the data set are $x_1, x_2, \ldots, x_N$, then the $N-1$ scatterplots in the staircase are $x_2 \times x_1, x_2 \times x_3, x_4 \times x_3, x_4 \times x_5, \ldots$ The entire staircase pattern is $\lfloor N/2 \rfloor$ columns wide and $\lceil N/2 \rceil$ rows tall. Although not implemented, it would be possible to rotate the staircase 45 degrees (Figures 4.9D, 4.10C) to take up less screen space. If each axis has length L, we can compare the space efficiency of the SPLOS, rotated SPLOS, and other multidimensional visualization techniques (Figure 4.1), where the Parallel Coordinate Plot is assumed to have spacing between axes that is proportional to L (i.e., the spacing is kL, for some constant k), to prevent the slopes of line segments from becoming too extreme.

From the comparison, we see that the area required by the SPLOS is about 1/4 that required by a standard SPLOM. This means the user can have an overview of data that takes up less screen space, or that can be enlarged to take up the same space as a standard SPLOM but with each scatterplot 4 times larger. The area required by the rotated SPLOS is even smaller at $\frac{3}{2}NL^2$ (i.e., linear in N, and comparable to PCP). Comparing the rotated SPLOS with a single row of scatterplots, the rotated SPLOS requires 50% more area but has an aspect ratio that is 3 times closer to 1 (which may be advantageous in certain contexts) and also has the advantage that each dimension in the SPLOS lies on an axis shared by two scatterplots, easing comparisons. Note that the scatterplots used in the SPLOS are the same as those embedded in PCPs in (Holten and van Wijk, 2010), but with the advantage that pairs of scatterplots with a common axis are adjacent and aligned, again easing comparisons. Finally, given the perceptual

advantages of scatterplots over parallel coordinates (Li *et al.*, 2010), we feel the SPLOS is an interesting compromise between the standard SPLOM and PCPs.

Table 4.1 The area required for different SPLOMs configurations.

	Area Required	Aspect Ratio
Standard SPLOM (lower triangular half only)	$(N-1)L \times (N-1)L = \Theta(N^2L^2)$	1
Single Row of Scatterplots (e.g., Figures 4.10A, 4.10B, 4.11A)	$NL \times L = NL^2$	N
Scatterplot Staircase (SPLOS) (e.g., Figures 4.9A, 4.9C)	$\lfloor N/2 \rfloor L \times \lceil N/2 \rceil L = \Theta(\frac{1}{4}N^2L^2)$	$\Theta(1)$
Scatterplot Staircase (SPLOS) rotated 45° (e.g., Figures 4.9D, 4.10C)	$\frac{1}{\sqrt{2}}NL \times \frac{3}{\sqrt{2}}L = \frac{3}{2}NL^2$	$\frac{1}{3}N$
Parallel Coordinate Plot (PCP) (e.g., Figures 4.10D, 4.11D)	$(N-1)kL \times L = \Theta(kNL^2)$	$\Theta(kN)$

4.7.3 Parallel scatterplot matrix (P-SPLOM)

Our system allows the scatterplots within a SPLOM to be rotated in 3D around either their vertical or horizontal axis. Figure 4.11 illustrates this for a single row of a SPLOM. Figures 4.11A through 4.11D show how a rotation around the vertical axes causes the visualization to transition from a sequence of scatterplots to a PCP. In between these two extremes, the user may rotate around both the vertical and horizontal axes (Figures 4.11E, F) yielding a 3D parallel coordinate plot (3D PCP) similar to those in (Falkman, 2001; Rübel and et al, 2006). Although this is done for a single row of the SPLOM in Figure 4.11, such rotation is also allowed within the full matrix (Figures 4.12, 4.13). Because these visualizations combine SPLOMs, PCPs, and 3D PCPs with seamless transitions, we call this combination based on rotation a Parallel Scatterplot Matrix (P-SPLOM).

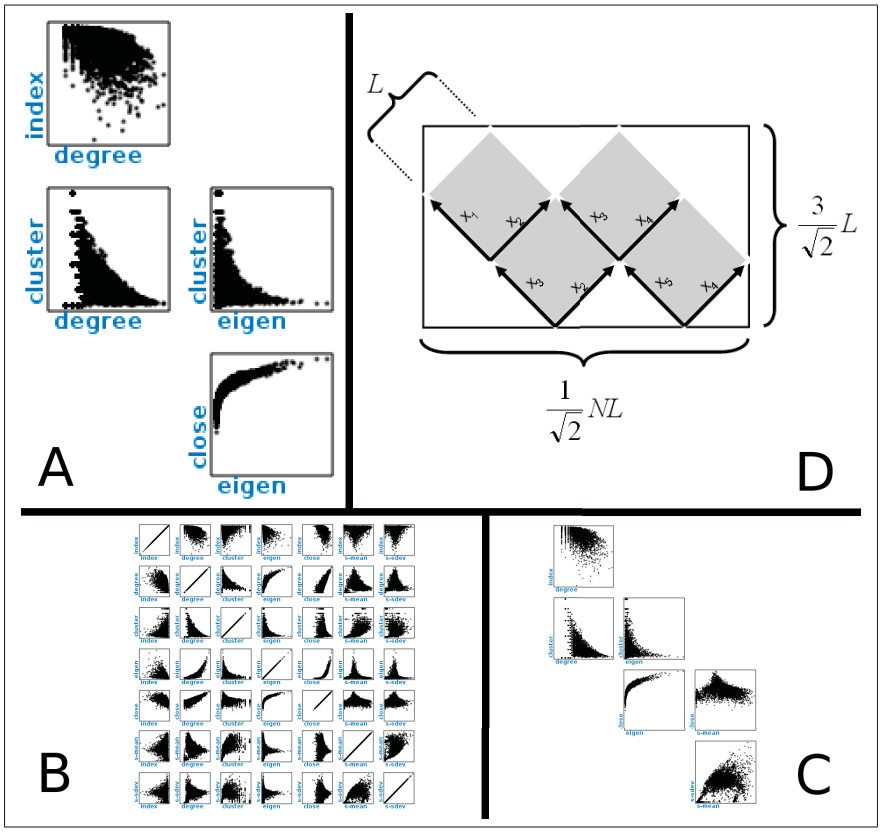


Figure 4.9 *A*: a SPLOS of 4 scatterplots, for the same dataset and same 5 dimensions as in Figure 4.8. *B and C*: a standard SPLOM (*B*) of 7 dimensions, and the corresponding SPLOS (*C*) for the same 7 dimensions but composed of only 6 scatterplots. *D*: If there are N dimensions, and the $N-1$ scatterplots have axes of length L, and the staircase is rotated 45° counterclockwise, the area of the bounding rectangle is linear in N.

Several orderings of the axes are possible within P-SPLOMs. For illustration purposes, consider first the axes within a standard SPLOM (Figure 4.8). If there are five dimensions named A, B, C, D, and E, we could specify the axes within the standard SPLOM with the table 4.2.

If the above ordering of axes is used within a P-SPLOM, the PCPs that result after rotation are not useful, because each row and each column will contain the same PCP axis repeated five times (see Figure 4.13, top row). Thus, our system also allows an ordering of axes that we call *Doubly-Latin*, because it involves two Latin squares (see table 4.3).

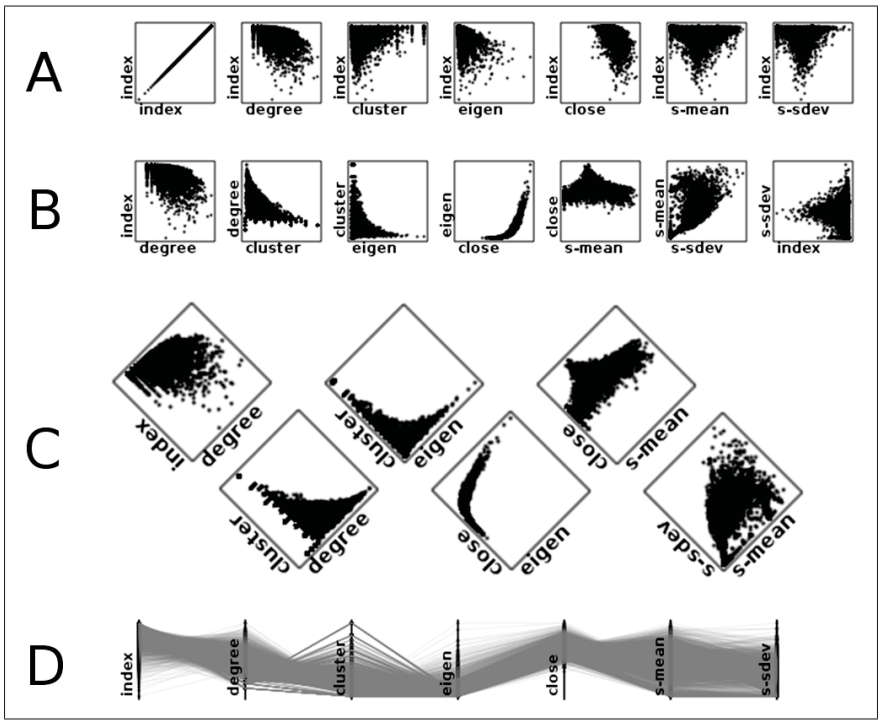


Figure 4.10 Four visualizations of the same 7-dimensional network data. *A*: a single row of scatterplots from a standard SPLOM. Notice that one dimension (`index`) is crossed with all other dimensions. *B*: another row of scatterplots, now where each scatterplot involves 2 consecutive dimensions. *C*: a Scatterplot Staircase (SPLOS) rotated 45 degrees. Each adjacent pair of scatterplots share an axis. *D*: parallel coordinates plot (PCP). The SPLOS is scaled and positioned such that, for each axis shared by two scatterplots in the SPLOS, the midpoint of the axis is aligned with the corresponding axis of the PCP.

Table 4.2 Description of a standard SPLOM.

Standard SPLOM									
horizontal axes					vertical axes				
A	B	C	D	E	A	A	A	A	A
A	B	C	D	E	B	B	B	B	B
A	B	C	D	E	C	C	C	C	C
A	B	C	D	E	D	D	D	D	D
A	B	C	D	E	E	E	E	E	E

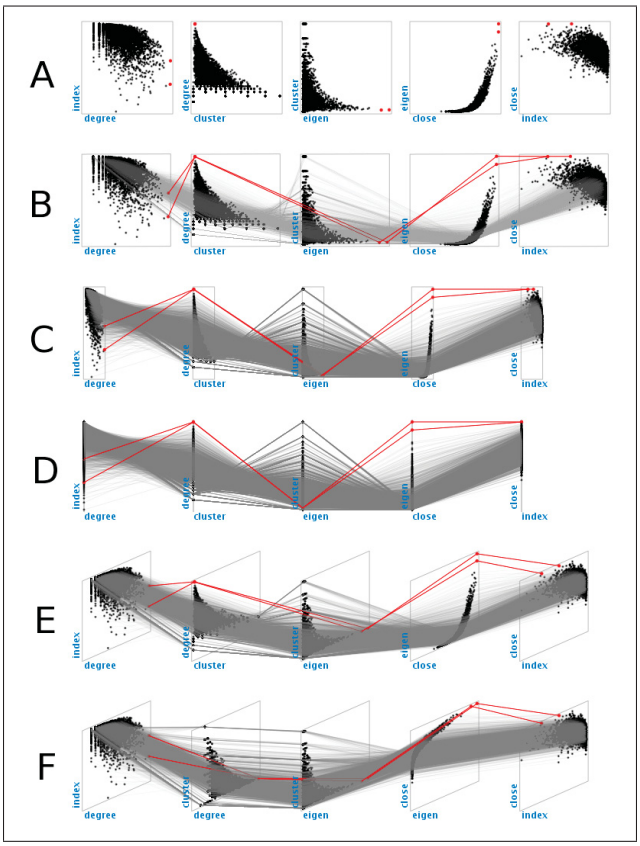


Figure 4.11 A single row of a P-SPLOM. *A*: a single row of scatterplots, with two points selected and highlighted in red. *B*-*C*: as the scatterplots are rotated around their vertical axes, links (in grey) between corresponding points are faded in. *D*: after rotating 90 degrees, the result is a Parallel Coordinate Plot. *E*: during rotation, the user may also rotate around the horizontal axes, resulting in a 3D PCP. *F*: an alternate ordering of axes for 3D PCP, facilitating comparison between consecutive pairs of scatterplots.

Table 4.3 Description of a doubly-latin SPLOM.

Doubly-Latin SPLOM									
horizontal axes					vertical axes				
E	D	C	B	A	A	B	C	D	E
A	E	D	C	B	B	C	D	E	A
B	A	E	D	C	C	D	E	A	B
C	B	A	E	D	D	E	A	B	C
D	C	B	A	E	E	A	B	C	D

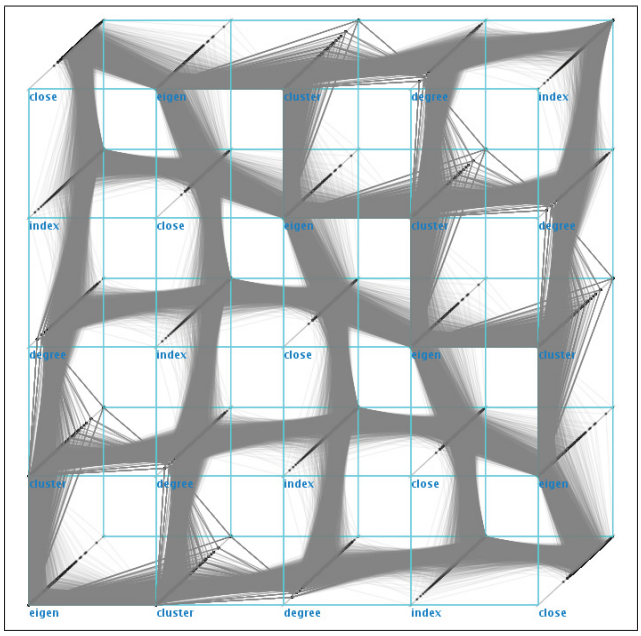


Figure 4.12 Rotation of scatterplots within a P-SPLOM is possible along the horizontal and vertical axes simultaneously. Here, the user has rotated the scatterplots approximately 90 degrees around the horizontal axes, and 45 degrees around the vertical axes. Links between corresponding points are shown in grey. Each row and each column of this matrix could be further rotated into a PCP.

An example of this ordering is in Figure 4.13 (2nd row). The Doubly-Latin ordering is useful for transitioning to PCPs because every row contains each kind of vertical axis once, and every column contains each kind of horizontal axis once. Thus, if the user focuses on any single row or column and rotates toward a PCP, all dimensions will be visible after the rotation within that one row or column. Figure 4.12 shows the full matrix for a Doubly-Latin P-SPLOM during rotation. Each column and each row of this figure is a 3D PCP that would (after complete rotation) contain all axes. As a side note, if the number of dimensions is odd, the Doubly-Latin ordering results in an Euler square (also known as a Graeco-Latin square), which has the property that every possible pair of dimensions occurs once. This is useful *before* rotating toward PCPs, since it guarantees all possible scatterplots will be visible in the full matrix. In other words, the Euler square is a permutation of the scatterplots in the standard SPLOM.

At the same time, the Doubly-Latin ordering has the disadvantage that within any given row or column, there can be redundant scatterplots. For example, within the Doubly-Latin matrix

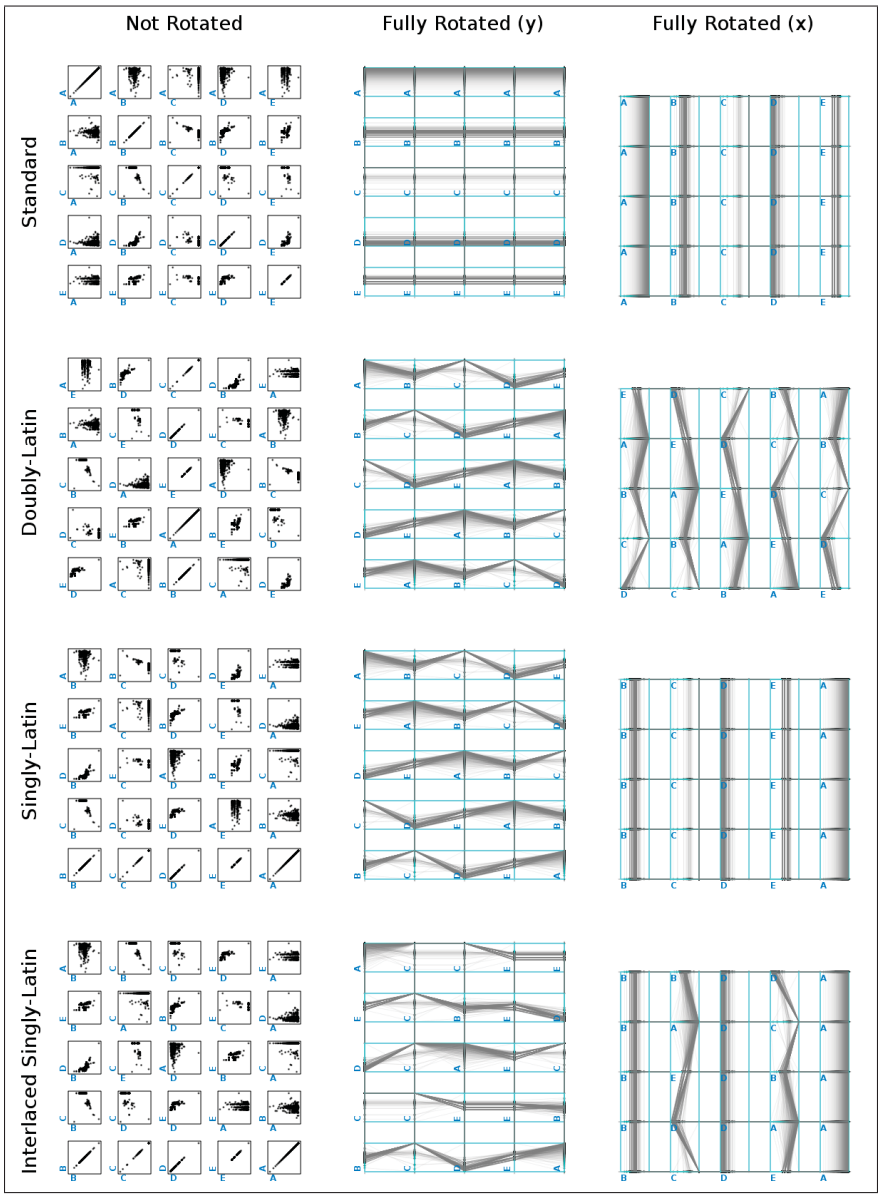


Figure 4.13 Each row shows a different ordering of the dimensions within the P-SPLOM. The left column shows the full scatterplot matrix (without rotation), the middle column shows the PCP resulting after a 90 degree rotation around the *y* axes, and the right column shows the PCP resulting after a 90 degree rotation around the *x* axes.

shown in Figure 4.13, the first row of scatterplots is E×A, D×B, C×C, B×D, A×E; i.e., the last two scatterplots are transpositions of the first two. The same redundancy is seen in the Doubly-Latin scatterplots in Figure 4.14. This redundancy does not matter after a full rotation toward PCPs, but is does make the Doubly-Latin ordering less attractive for use with 3D PCPs of a

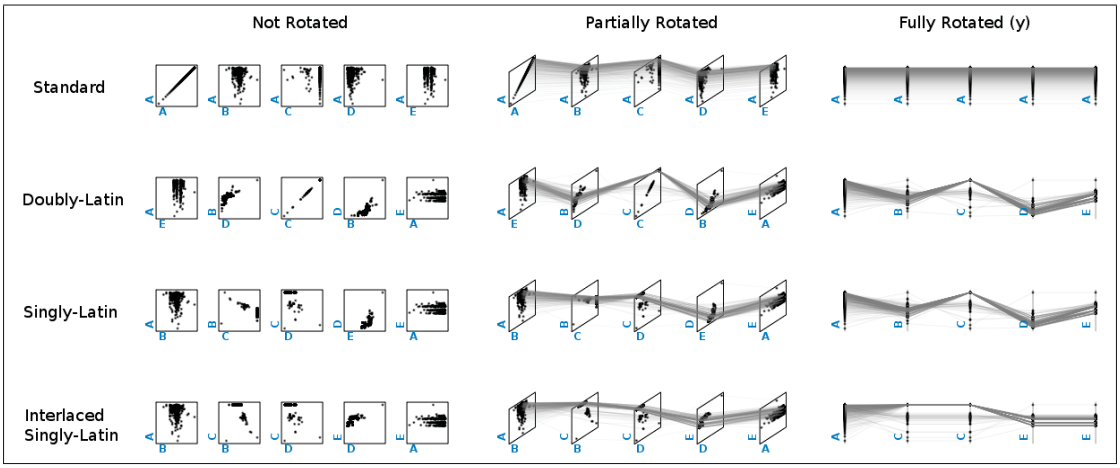


Figure 4.14 Each row shows a different ordering of the dimensions within a single row of scatterplots. The left column shows the scatterplots without rotation, the middle column shows the 3D PCP resulting after a partial rotation, and the right column shows the PCP resulting after a 90 degree rotation around the *y* axes.

single row or column. Thus, our system also allows for what we call a *Singly-Latin* ordering, because it only involves one Latin square:

Table 4.4 Description of a singly-latin SPLOM.

Singly-Latin SPLOM										
horizontal axes					vertical axes					
B	C	D	E	A	A	B	C	D	E
B	C	D	E	A	E	A	B	C	D
B	C	D	E	A	D	E	A	B	C
B	C	D	E	A	C	D	E	A	B
B	C	D	E	A	B	C	D	E	A

The above Singly-Latin ordering has the advantage that within any given row (but not column), we find each vertical axis once (making it good for rotation toward PCP) and each horizontal axis once. Furthermore, within any given row, there are no redundant scatterplots, making the ordering reasonable for 3D PCPs. For example, Figures 4.11A through 4.11E show the first

row of the Singly-Latin P-SPLOM whose ordering is given above. Notice the pattern of pairs of dimensions in the 1st row: $B \times A, C \times B, D \times C, \ldots$ (i.e., the x axis of one is the y axis of the next). The same pattern is seen in Figure 4.10B.

There is, however, still a minor disadvantage of the Singly-Latin ordering for 3D PCPs: within the 1st row, each pair of consecutive scatterplots share a dimension on *different* axes, making comparisons more difficult. For example, in Figure 4.11E, the left-most scatterplot has `degree` on its horizontal axis, and the next scatterplot has `degree` on its vertical axis. Thus, our system also supports an *Interlaced Singly-Latin* ordering, which results from taking the Singly-Latin ordering and swapping the axes of scatterplots in every other column. The first row of an Interlaced Singly-Latin P-SPLOM is shown in Figure 4.11F. Notice now that `degree` is the horizontal axis of the 1st and 2nd scatterplots, and `cluster` is the vertical axis of the 2nd and 3rd scatterplots, etc. Notice also that the scatterplots in Figure 4.11F are the same as in the staircase pattern of Figure 4.9A! This ordering facilitates comparisons across the scatterplots when using the 3D PCP.

The table 4.5 summarizes the tradeoffs between the different orderings for P-SPLOMs.

Table 4.5 Tradeoffs between different configurations for P-SPLOMs

	full scatterplot matrix (not rotated) Fig. 4.13	single row of scatterplots (not rotated) Fig. 4.14	single row 3D PCP (partially rotated) Fig. 4.14	single row PCP (fully rotated) Fig. 4.14
Standard	best	good	good	worst
Doubly-Latin	redundant if N even	redundant	redundant	best
Singly-Latin	good	good	good	best
Interlaced Singly-Latin	good	good	best	redundant

The user may switch between all these orderings at run time, triggering a smoothly animated transition from the old ordering to the new ordering. The user may also switch between viewing the full matrix or focus on a single row, and may also transition between "flat" scatterplots to 3D PCP or PCP through rotation (Figure 4.11). Thus, the user is free to choose the visualization that best suits their needs. Nevertheless, the above table indicates that the Singly-Latin ordering may be the best compromise across all the views of the data, since it is the only one that avoids redundancy in all cases.

4.8 Initial user feedback

We asked five bioinformaticians (four men and one woman, between 24 and 43 years old) to help us evaluate our system's user interface for the exploration of biological networks. All five participants are experienced computer users and work with network data. Each participant was given a ten minute demonstration and explanation of the interface, in part to demonstrate some of the possibilities afforded by metric-based exploration. Next, a set of questions were asked regarding the potential relevance of metric-based graph exploration for bioinformatics.

Participants were then allowed to freely explore the interface while thinking aloud. A second set of open-ended questions was then asked regarding the user interface and the participant's impressions. Participants quickly understood the logic behind each interface element. The effectiveness of the interface for selection was noted in particular ("it's a very quick way to pick out interesting outliers", "it allows you to swiftly do something that might have taken a lot longer just manually picking through the graph.") Smoothly animated transitions were seen as useful ("The transition is helpful, because I think if it was just done without transition, then you would lose sight of [the data]"). One participant stated "It would be cool if there were some sort of protein, say, that had high degree and low betweenness centrality, some unexpected relationship." Upon hearing this, one of us (Jurisica) pointed out that the opposite would be interesting: a node with high centrality and low degree would correspond to a node that, if removed, would disconnect the graph. All participants stated they would use the interface in their daily work if it were available to them.

4.9 Example of use with real-world data

To further evaluate our prototype, we tested its use with a real-world biological network (Figure 4.15). We wanted to visually explore the network and see how we could make use of the advanced features of the interface, going beyond simply selecting clusters or high degree nodes as is possible with status quo software. First, in the P-SPLOM and in the force-directed layout, we could see that the network was composed of a few nodes of high degree surrounded by many nodes of degree one. A preliminary layout was made by combining FlowVizMenu selection with hotbox (McGuffin and Jurisica, 2009) layout commands. The highest-degree nodes were selected using the FlowVizMenu, and then laid out in a circular pattern (and locked in place) using the hotbox. The degree-one neighbors of the highest-degree nodes were then selected and laid out on a secondary circle (also locked in place) surrounding the first. The remaining nodes, in the center of these two circles, were structurally less obvious. To find interesting features in these remaining nodes, we tried to follow the insight that a node with low degree but high centrality could be a good candidate to investigate. The FlowVizMenu was used to select these two dimensions and steer the ADL view. One node stood apart and corresponded to our criteria. A direct selection in the ADL revealed the name of one particular protein, which our bioinformatics collaborators intend to further investigate.

4.10 Conclusions and Future Directions

We have presented novel approaches to multivariate graph visualization that integrate previous techniques, through popup gestural interaction (the FlowVizMenu) and through the use of hybrids (the P-SPLOM which unifies scatterplot matrices, normal 2D parallel coordinates, and 3D parallel coordinates, with seamless transitions between them; and our attribute-driven layout which can be mixed with force-directed layout and manual positioning). We have also presented an investigation of possible orderings of dimensions within P-SPLOMs, and the novel Scatterplot Staircase (SPLOS). Together, we feel these techniques provide a flexible toolbox for dissecting a network, isolating nodes of interest, and manipulating the layout. We also note that the P-SPLOM and SPLOS are general multidimensional visualization techniques that could be applied to non-network data.

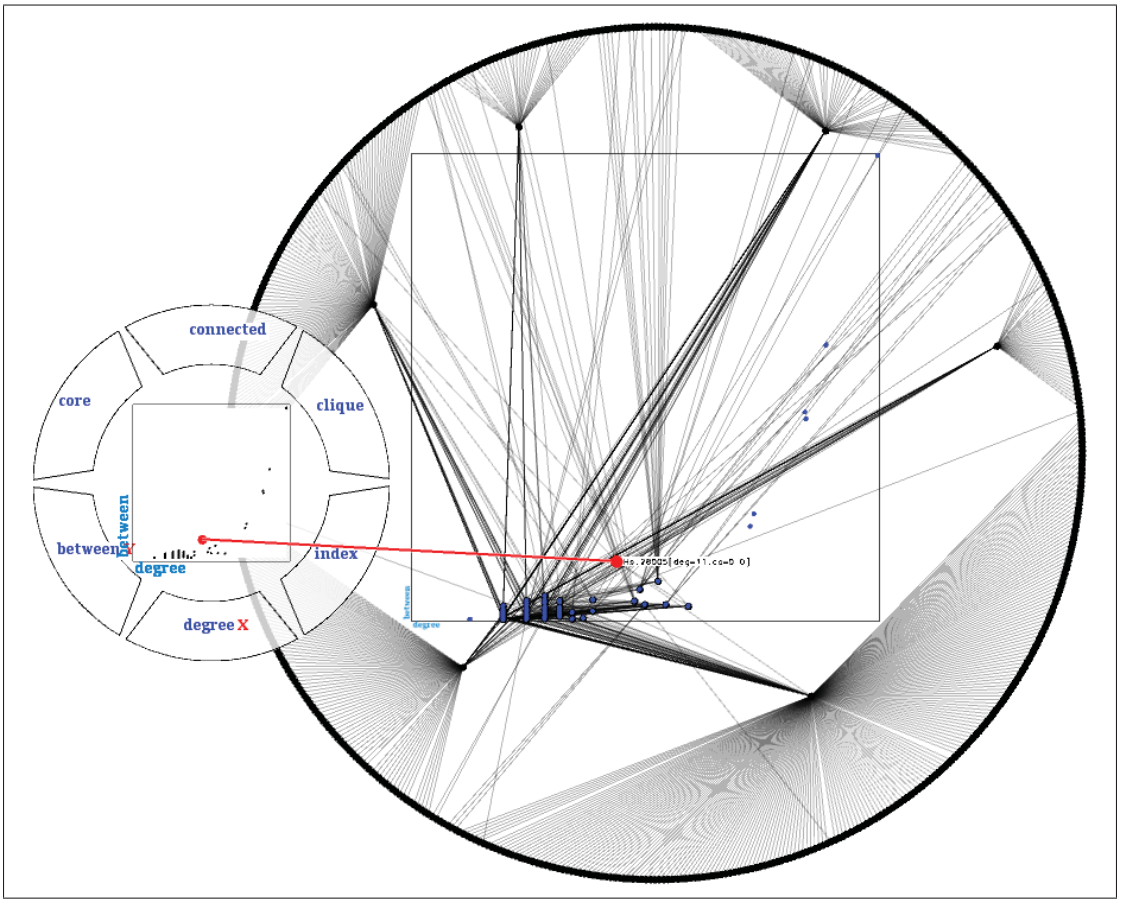


Figure 4.15 Example use of the FlowVizMenu combined with ADL to explore a biological network. Prior to using the ADL shown here, the highest-degree nodes and their low-degree neighbors were manually positioned onto circular layouts. The remaining nodes are laid out with ADL to find a candidate node with high centrality and low degree, since such properties could signify it acts as a bridge, having special biological significance.

Possible future directions include: deploying the techniques as a plugin for biological network software such as NAViGaTOR (Brown *et al.*, 2009) or Cytoscape; applying the FlowVizMenu, P-SPLOM and SPLOS to non-network data; and further adapting the FlowVizMenu or related techniques for use on small screens or mobile platforms.

One question that remains open is what value, if any, there is in displaying a full matrix of parallel coordinates (Figures 4.12 and 4.13). Currently, there is much redundancy in these views due to pairs of axes being repeated. However, a different ordering of dimensions might

eliminate this redundancy. Otherwise, users may be best to reduce the matrix to a single row or column before rotating to PCP.

There are also several avenues open for future evaluation. For example, how does user performance with our FlowVizMenu compare to that with GraphDice's control menu? How accepting are users of SPLOMs that do not use the standard ordering? These can only be answered by future work.

CONCLUSION

Summary and discussion

The last three chapters presented a pipeline for characterizing hybrid visualizations as well as several innovative prototypes. The first article, Characterizing Hybrid Visualizations, describes an extension of a standard visualization pipeline, showing how existing visualizations can be visually described and new ones can be designed using this model. The second article, ConnectedCharts, presents a subset of the possibilities that can be described by the pipeline, focusing on explicit linking between graphics. The final article, FlowVizMenu, presents a more domain-specific application of another subset of possibilities: combining multidimensional visualizations with graphs for biological network exploration.

The three papers explore the design space of hybrid visualizations. Designing hybrids by combining multiple visualization can be a way to use multiple graphical encodings for different parts of the dataset or for different tasks. Two visualizations can be placed side by side in the same functional space. A coordination between them, like selecting from one to highlight corresponding values in the other, can help place them in synergy, and a whole range of assembly patterns can be explored.

Hybrid visualizations

Characterizing hybrid visualizations presents a pipeline describing how to combine visualizations to form new hybrids. Six different kinds of combinations are illustrated by this pipeline: side-by-side assembly, overlay assembly, heterogeneous visualizations, nested visualizations, hybrid layouts, and hybrid glyphs.

The pipeline starts form the dataset that can be split into parts. Each part can use a different encoding at various stages. First, a glyph is generated to represent the data values, then a layout positions these glyphs. More graphical element are added to decorate the result of the previous stage, which is then assembled into the view.

They can be combined in multiple ways. Side-by-side assembly corresponds to small multiples or coordinated views, where many visualizations share the same space, side-by-side or stitched on each other. Some examples are coordinated views (Roberts, 2007), SPLOMs and MatLink (Henry and Fekete, 2007b) Overlaying assembly superimposed multiple visualizations, often using transparency or cutout for both visualizations to be visible. An example is Macroscope (Lieberman, 1997). In heterogeneous combinations of glyphs, two data subsets are rendered using different encoding and then recombined. Elastic hierarchies (Zhao *et al.*, 2005) and NodeTrix (Henry *et al.*, 2007a) both combine two visual representations of a graph dataset. Nesting assembly embed some visualizations into another. For example in TableLens (Rao and Card, 1994), bar charts are nested in a table view. Hybrid layout combines multiple layout into one visualization. For example, GraphDice (Bezerianos *et al.*, 2010b) positions the nodes of a node-link diagram according to the dots of a selected scatterplot. Hybrid glyph generation assembles the different graphical encodings at the glyph generation stage. For example a polar area chart can be combined with a pie chart using varying angles, as in a pie chart, and varying heights, as in polar area chart, for each wedges.

This chapter also demonstrates the usefulness of the pipeline for designing new hybrids. The first example explores the combination of scatterplots. The second one combines scatterplots and PCPs. This conceptual framework helps define what is a hybrid visualization and how to assemble them. Many configurations can be explored using this tool. We presented in the following chapters some new hybrids derived from this conceptual framework.

ConnectedCharts

ConnectedCharts illustrates the assembly by explicit linking between a large number of graphics revealing a system to characterize the data dimensions and variables represented. An interface allows building interactive assemblages inspired by existing or novel hybrid combinations.

ConnectedCharts generalizes some hybrids found in the literature. Flexible Linked Axis (Claessen and van Wijk, 2011) (FLINA) provides tools for the layout of PCP axes in any configurations, allowing arbitrary axis to be converted to a scatterplot or a histogram. Our work can replicate

part of this interface, but with more different types of charts and with a way to automatically infer the connectability of any two charts.

Another model generalized by ConnectedCharts is Dimensional Anchor (Hoffman *et al.*, 1999) (DA) a simple encoding pattern for quantitative data where each value is mapped to a position on a line and is projected from this line. We can easily recognize this pattern in a PCP axis. But the same DA can form the basic structure of a bar chart, a scatterplot and many more multidimensional and multivariate graphics. The author shows many original variants of existing visualizations using DA, like polar PCPs, in his thesis (Hoffman, 1999). DA, as discussed by the author, is a "graphic primitive" to build a lot of different charts. Our framework extends this notion to linking different charts, like if the DA structure were projecting outwards, trying to connect to other related anchors. DA is not a hybrid, it is a graphical primitive to assemble charts. ConnectedCharts use the same kind of primitive extensively to study a complete design space of hybrid visualizations by stitching.

One of the main goals of ConnectedCharts was to explore explicit linking between charts. We see a lot of brushing and linking in the literature. The explicit linking, with arrows and lines added to link related elements is far less common. It is used with graph data structure, where links are part of the dataset. But links between charts are supplementary information not in the dataset. It adds high-level information on the relationship between the different encoding used, revealing their structure and the relationship between these structures. There is a good evaluation of this type of link in a paper from Waldner *et al.* (2010) and another from Steinberger *et al.* (2011) and a good formal description of an explicit linking interface in Collins and Carpendale (2007).

Some applications or improvement of ConnectedCharts were not discussed extensively in the paper. For example, this type of relationship inference could be used to partly automate or to guide the layout of a dashboard. We can imagine having some related charts laid out according to the affinities, to the connectability between charts without displaying explicit links but using the same framework to indentify these relations. For example, bar charts and scatterplots sharing a common axis can be aligned and rescale for easy comparison. Or an interface like

a “Snap-together” visualization (North and Shneiderman, 2000b) could detect the type of the shared axis when two charts are juxtaposed, or snapped, and scale accordingly, or they could even change axis dimension to share with the other. The ability to link any kind of data configuration, copy, subset and aggregation, is one of the main contributions of ConnectedCharts.

One interesting feature of ConnectedCharts is the way it cycles through dimensions when it is cloned. For example, cloning an axis adds a new chart with the next dimension found on the dataset. This could lead to the design of a shortcut to quickly generate an encoding pattern like a PCP from a series of axes and a SPLOM from a grid of scatterplots. A gesture could start a series of cloning following the pattern detected in the first few cloning steps. For example, an axis is cloned, adding another one with the next dimension in the dataset. Then the last one is also cloned. Then, a gesture can tell the interface to generate a series of axes, one for each dimension in the dataset, all linked by polylines to form a PCP of the complete dataset. It is in fact kind of an hybrid generator, but some hybrids are more common and there generation could be further automated.

FlowVizMenu, P-SPLOM and A-D layout

FlowVizMenu and the related hybrid network visualizations explore the combination of multidimensional and graph visualizations. The FlowVizMenu is a radial menu inspired by the FlowMenu (Guimbretière and Winograd, 2000) and the ControlMenu (Pook *et al.*, 2000) manipulating the state of a visualization integrated at its core. This is an example of nesting, to use the same terminology as our pipeline, supplemented by an interaction operating on the menu display. This integration by interaction is common when a visualization is integrated to a control element. A visualization, like a control element, can be used to manipulate another. The scatterplot of FlowVizMenu as well as the P-SPLOM provide a brushing and linking and explicit linking of the nodes of the network corresponding to the points selected in the scatterplot. In addition to this first level of integration by explicit linking, our P-SPLOM consists of SPLOM which can transition to a PCP. This fusion between the points of a PCP and those of a scatterplot is an example of integration at the glyph level, where the elements of a visualization are reassembled, where the structure changes and not only the graphical attributes. The explo-

ration of different configurations for different latin squares SPLOMs shows several *variants* of SPLOMs. A third visualization, the Attribute-Driven Layout (ADL) positions the nodes of the network on each corresponding point of a scatterplot. This is another example of integration by fusing glyphs, where the nodes of a node-link diagram fuse with the points of a scatterplot. The FlowVizMenu is used to select the attributes to be used for the scatterplot of the ADL. All the visualizations are integrated by juxtaposition, brushing and linking, coordinated interaction, explicit linking, nesting and fusion, thus covering a wide range of integration described by the hybrid assembly pipeline of chapter 2.

The FlowVizMenu had some influence on the design of ConnectedCharts. For example, one alternate design shows the FlowVizMenu as a tool to generate some charts, changing chart types with the number of dimensions and laying down a series of chart, almost like in the cloning, changing dimensions and changing chart type process in ConnectCharts. In fact, this whole process could have been done with a FlowVizMenu instead of drag and drop, popup menus and keyboard hotkeys. An assembly similar to one P-SPLOM state can also be built with ConnectedCharts, where each scatterplots of a SPLOM is linked to its neighbor.

In all the hybrid visualizations we proposed, the most tightly integrated one is probably the P-SPLOM. The rotation of each scatterplot of the SPLOM can be seen as a collapsing or a projection of each scatterplot on one of their axes. This kind of projection of every dots on an axis is done the same way as the projection on the axes in ConnectedCharts to define the anchors for linking each axis.

Contributions an evaluation

The main contribution of this thesis is to propose novel hybrid visualizations applied to the exploration of the most common data types in Infovis (i.e., multidimensional multivariate and graph), as well as an original pipeline for the description of hybrid visualizations, to better define, analyze and design this type of visualization so widespread and so little studied.

The type of design and theoretical framework we present in this thesis can't hardly be validated like a paper about a new algorithm or about the application of an existing visualization to a spe-

cific domain. Munzner (2008), in "Process and Pitfalls in Writing Information Visualization Research Papers", categorizes and describe precisely the different types of contributions. For simplicity, we can say that innovations can be in the *design* (of new algorithms, development tools or visualizations), in the *evaluation*, application and comparison of existing visualizations, or in the development of theoretical *models*. Our research touches on two of these three categories. Characterizing Hybrid Visualizations is about the design and the description of visualizations. It was inspired by Chi's and Card's model previously cited. To develop and validate our model, we first based our work on solid ground, extending a standard pipeline metaphor to the design of hybrids and discussing the rationale extensively. Second, we proved the usefulness of this pipeline by creating some new hybrid visualizations. ConnectedCharts and FlowVizMenu are natural extensions of this work and, even if not using the pipeline as a diagram, is using the same workflow for the exploration of the design space of many new ideas.

ConnectedCharts can be seen as an article about a new hybrid visualization exploring why and how to connect multiple charts. But the most important parts of the paper are defining the rules to describe the mapping from data to graphics, inspired and built on previous work, to describe some interesting visual and conceptual relationships between different graphical encodings. We validated this model of relationships between charts and this exploration of the design space of explicit linking by building a prototype and by describing how the model generalizes previous work like FLINA (Claessen and van Wijk, 2011), DA (Hoffman *et al.*, 1999) and Product plots (Wickham and Hofmann, 2011). This work also validates the hybrid conceptual framework by precisely describing a constrained design space where whole charts are linked on the view levels by adding lines to emphasize the relationships between the data elements mapped on their axes.

FlowVizMenu is less of an application study than the description of a design space. We started with a general problem in bioinformatics: exploring complex data with a combination of multidimensional and graph visualizations. Starting from this challenge, we explored the possibility of graphically and functionally combine traditional multidimensional and graph visualizations, like SPLOM, PCP and node-link diagram. We validated this design by qualitative assessments,

but the most important part of this work, the exploration of the design space of hybrid visualizations, was validated mostly by analytical justification of design choices and by discussions of alternatives.

Future work

The study that we reserve for future work would make a more comprehensive overview of hybrid visualizations and evaluate the effectiveness of hybrids for a variety of tasks in a variety of areas. The most important comparison to do would be between a set of integrated visualizations and the same set without integration. We hope our characterization work can allow further studies of whether the integration of different visualizations is an effective strategy.

Exploring the design space of hybrid visualizations is also a subject worth investigating. One way of exploring these possibilities is by dissecting visualizations in smaller components and finding a way to reassemble them in a new visualization. For example, a bar charts is made of rectangles, axis, and textual marks. For each rectangle, the height is encoding quantitative data and the relative position is possibly encoding an ordering. One way to integrate it with a PCP, for example, is to align a horizontal bar chart on each axis, as in FLINA and Guided Analysis of Hurricane Trends (Steed *et al.*, 2009). In both cases, the bar chart is a histogram, the height of each bar encoding the frequency of each range. But other configuration are possibles. A bar chart could redundantly encode the values also encoded by the dot of each axis. In this configuration, the bar chart help spread the marks, that can be superimposed and occluded on the axis, and allows them to be sorted according to an arbitrary criteria. Each PCP axis can have one bar representing, for example, the mean value, only forming a bar charts when looking at all the bars on all axes. The spacing between axes can vary in width according to the values encoded by the bar chart. There is a lot of possibilities to explore, possibly one for each way of combining two elements of the visualizations to integrate.

A tool we could call a design space matrix, as illustrated in fig 4.16, can present a matrix of possible combinations. Both charts to assemble can be dissected in arbitrarily small components, like dots, axes, lines, points. The components of one chart can be listed on the rows and

the other on the columns. Each cell is the intersection of a component of the first chart and of the second. For each possibility, the designer could try to imagine what would be a good way to integrate those components. This design space matrix can be a tool to explore less obvious possibilities before relying on well known patterns.

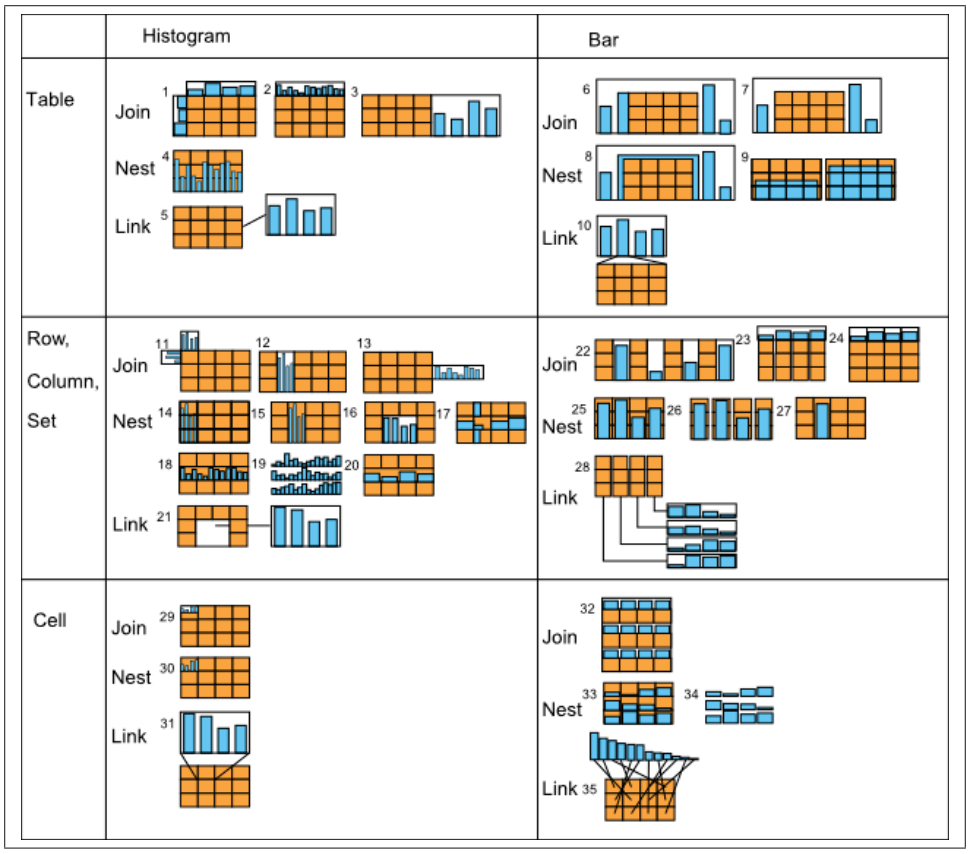


Figure 4.16 Multiple combinations can be displayed on a matrix with the graphical elements listed on the rows and on the columns. Each cell represents some possible combinations of a row element and a column element.

A design space largely unexplored is the transitioning between visualizations. Instead of finding ways to integrate charts spatially, it can be done temporally, a chart transforming into another. One excellent evaluation of animated transitions is the one by Heer and Robertson (2007).

Each integration pattern could be studied more deeply. ConnectedCharts was all about explicit linking. But the same exercise could be done with nesting, juxtaposing, synchronized interac-

tion or certain types of fusion of elements. For example, the systematic embedding of charts in a table, like the bar charts in TableLens (Rao and Card, 1994), could lead to novel hybrid visualizations transitioning from a table to a matrix of bars, a scatterplot, a stacked bar charts and even a node-link diagram.

An example of this idea of using transition between multiple charts nested in a table is shown in figure 4.17. A standard table shows some numbers on a grid. Then, each cell can be divided vertically proportionally to the value of the data. We can call the result a "matrix of bars", like the one in ConnectedCharts 3.6. The cells can also be divided vertically, and an animated transition helps to keep the mental map between the different configurations. In the next configuration, the bar on each cell can leave their cell and stack at the bottom of the table. The visualization is thus decoupled from the table, but the transition from matrix of bars to stacked bar chart makes it easy to relate the chart to the table. More complicated configurations can be done. For example, the figure 4.17 shows the transition to a scatterplot and to a node-link diagram. This whole concept is the result of a free design exploration, without starting from the needs or the tasks and without evaluation, but is useful to visualize and to challenge part of the design space of hybrid visualizations by starting with arbitrary design constraints, like using nesting on a grid and transitions between configurations. Future work could start from a real-world problem and explore a constrained design space instead of starting with a preconception of how the visualization task should be solved.

Our work extends part of the standard visualization pipeline going from the data to the view. But a large part of this model has been put aside, for example interaction. Chi's operator's interaction framework (Chi and Riedl, 1998) describes how a user can interact on every stage of the visualization pipeline. Extending this interaction framework to hybrid visualizations or to multimodal interaction could be an interesting challenge. More specifically, using visualizations as control elements, like we do with a menu or a form element, a study closer to HCI, is a promising avenue. Our FlowVizMenu integrates a visualization in a radial menu. But the visualization itself could act as a menu. The brushing and explicit linking between the FlowVizMenu and the network acts more like a selection menu. An example of this exploration

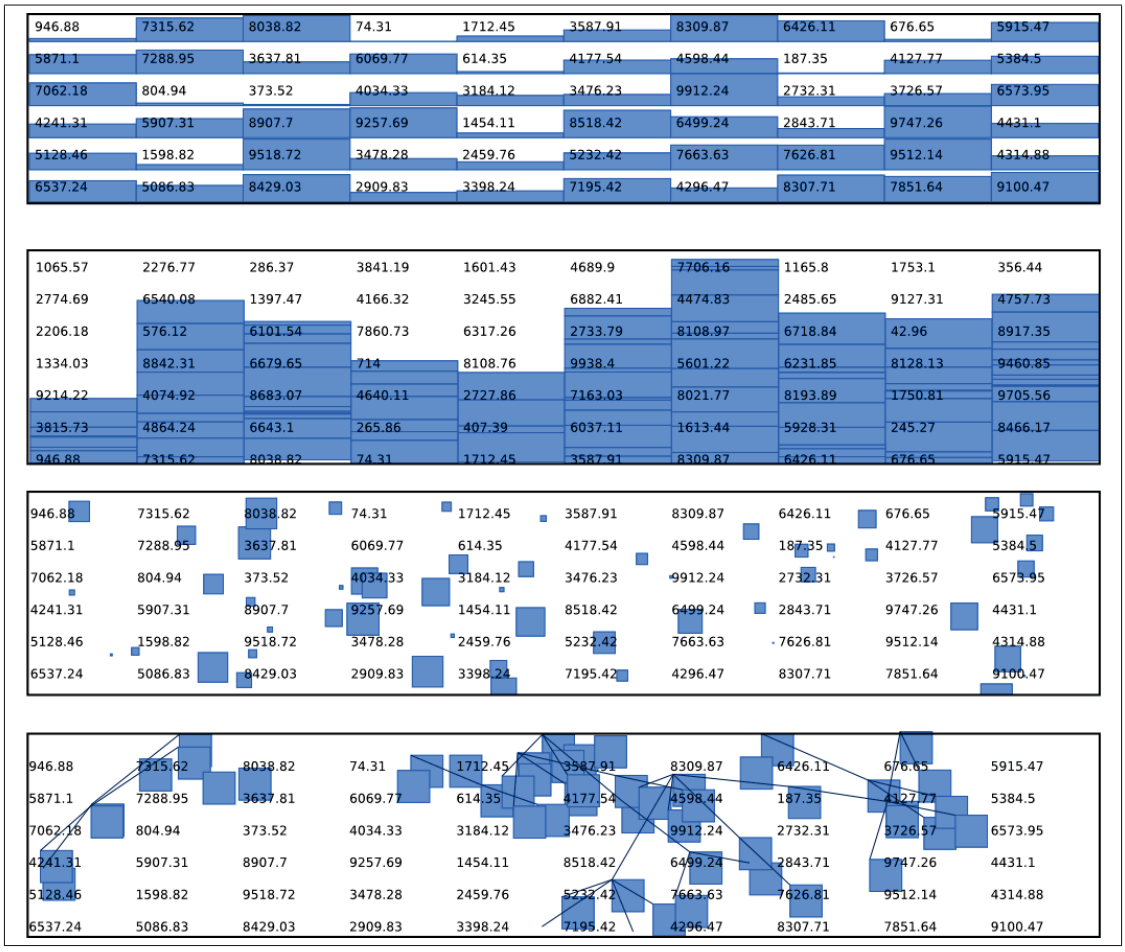


Figure 4.17 Prototype of a hybrid nesting visulizations in a table with animated transitions between them.

path is shown in figure 4.18. A visual menu is made of a percentage column chart. A click on a slice of the column opens another column, transitioning into a kind of icicle tree acting as a visual hierarchical menu.

Infovis is at the intersection of multiple field of research, like perception psychology, HCI, graphic design, computer engineering. Strong models are borrowed from semiology as well as from statistics and other apparently disjoint domain. The design of visualizations, more specifically, is a mostly a mix between information design and data science. But these different speciality collaborate in Infovis aiming for a specific goal: understanding and creating new tools to help users extract useful information and get insights from their data. Hybrid visualization is a design space mainly unexplored, but the many new hybrids and design studies we see

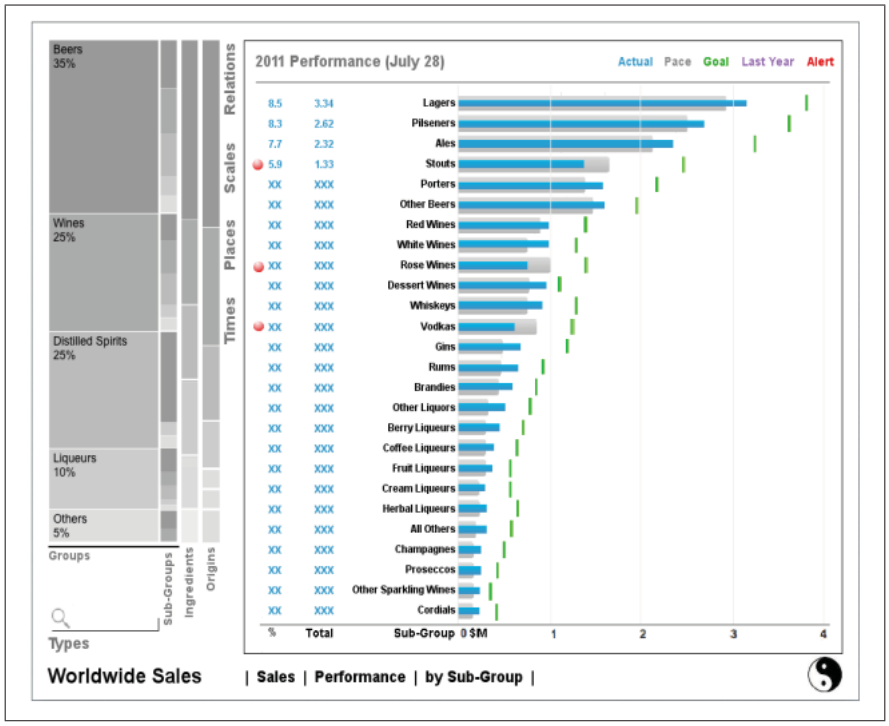


Figure 4.18 A visualization can be used as a control element. Here a 100% column chart acts as a menu. Each menu elements have a height representing some values associated with the element listed. A click on a menu element opens a submenu in a way that looks like an icicle tree. (© 2011 SAP)

in the recent literature show that the subject matters. Understanding how visualizations can be combined can be a challenging design task. But it is also a way to study how different mental models, different goals, different tasks, can collaborate in an exploration process to “amplify cognition” and to give a meaningful shape to data.

BIBLIOGRAPHY

J. Abello and F. van Ham. 2004. "Matrix zoom: A visual interface to semi-external graphs". In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*. p. 183–190. IEEE.

G. Albuquerque, M. Eisemann, D. J. Lehmann, H. Theisel, and M. Magnor. 2009. "Quality-Based Visualization Matrices". In *Proceedings of Vision, Modeling, and Visualization (VMV)*.

D. Archambault, T. Munzner, and D. Auber. 2007. "TopoLayout: Multi-Level graph layout by topological features". *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 13, n. 2, p. 305–317.

A. Aris and B. Shneiderman. 2007. "Designing semantic substrates for visual network exploration". *Information Visualization*, vol. 6, n. 4, p. 281–300.

D. Auber, D. Archambault, R. Bourqui, A. Lambert, M. Mathiaut, P. Mary, M. Delest, J. Dubois, and G. Melançon. 2012. *The Tulip 3 framework: A scalable software library for information visualization applications based on relational data*. Technical Report RR-7860. INRIA Bordeaux Sud-Ouest.

D. Auber, Y. Chiricota, F. Jourdan, and G. Melançon. 2003. "Multiscale Visualization of Small World Networks". In *Proceedings of IEEE Symposium on Information Visualization (InfoVis)*.

R. A. Becker and W. S. Cleveland. 1987. "Brushing Scatterplots". *Technometrics*, vol. 29, n. 2, p. 127–142.

B. B. Bederson and J. D. Hollan. 1994. "Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics". In *Proceedings of ACM Symposium on User Interface Software and Technology (UIST)*. p. 17–26.

J. Bertin, 1967. *Sémiologie graphique: Les diagrammes, Les réseaux, Les cartes*. (2nd edition 1973, English translation 1983), Paris : Éditions Gauthier-Villars.

J. Bertin, 1998. *Sémiologie Graphique : Les Diagrammes, Les Réseaux, Les Cartes*. Les ré-impressions. Ecole des Hautes Etudes en Sciences Sociales.

A. Bezerianos, F. Chevalier, P. Dragicevic, N. Elmqvist, and J.-D. Fekete. 2010a. "GraphDice: A System for Exploring Multivariate Social Networks". In *Proceedings of Eurographics/IEEE-VGTC Symposium on Visualization (EuroVis)*.

A. Bezerianos, F. Chevalier, P. Dragicevic, N. Elmqvist, and J.-D. Fekete. 2010b. "GraphDice: A System for Exploring Multivariate Social Networks". In *Proceedings of Eurographics/IEEE-VGTC Symposium on Visualization (EuroVis)*.

M. Bostock and J. Heer. 2009. "Protovis: A graphical toolkit for visualization". *Visualization and Computer Graphics, IEEE Transactions on*, vol. 15, n. 6, p. 1121–1128.

M. Bostock, V. Ogievetsky, and J. Heer. 2011. “D3: Data-Driven Documents”. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 17, n. 12, p. 2301–2309.

K. R. Brown, D. Otasek, M. Ali, M. J. McGuffin, W. Xie, B. Devani, I. L. van Toch, and I. Jurisica. 2009. “NAViGaTOR: Network Analysis, Visualization and Graphing Toronto”. *Bioinformatics*, vol. 25, n. 24, p. 3327–3329.

A. Buja, J. A. McDonald, J. Michalak, and W. Stuetzle. 1991. “Interactive Data Visualization using Focusing and Linking”. In *Proceedings of IEEE Visualization (VIS)*. p. 156–163, 419.

T. Büring, J. Gerken, and H. Reiterer. 2006. “User Interaction with Scatterplots on Small Screens - A Comparative Evaluation of Geometric-Semantic Zoom and Fisheye Distortion”. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 12, n. 5, p. 829–836.

W. Buxton. 1986. “Chunking and Phrasing and the Design of Human-Computer Dialogues”. In *Proc. IFIP World Computer Congress*. p. 475–480.

J. Callahan, D. Hopkins, M. Weiser, and B. Shneiderman. 1988. “An empirical comparison of pie vs. linear menus”. In *Proc. ACM Conference on Human Factors in Computing Systems (CHI)*. p. 95–100.

S. Card, J. Mackinlay, and B. Shneiderman, 1999. *Readings in information visualization: using vision to think*. Morgan Kaufmann.

S. K. Card and J. D. Mackinlay. 1997. “The Structure of the Information Visualization Design Space”. In *Proceedings of IEEE Symposium on Information Visualization (InfoVis)*. p. 92–99.

M. S. T. Carpendale. March 1999. “A Framework for Elastic Presentation Space”. PhD thesis, School of Computing Science, Simon Fraser University, Burnaby, Canada.

M. S. T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia. 1997. “Extending Distortion Viewing from 2D to 3D”. *IEEE Computer Graphics and Applications (CG&A)*, vol. 17, n. 4, p. 42–51.

S. Casner. 1989. *Cognitive efficiency considerations for good graphic design*. Technical report. DTIC Document.

H. Chernoff. 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.

E. Chi and J. Riedl. 1998. “An operator interaction framework for visualization systems”. In *Information Visualization, 1998. IEEE Symposium on*. p. 63–70. IEEE.

Y. Chiricota, F. Jourdan, and G. Melançon. 2004. "Metric-Based Network Exploration and Multiscale Scatterplot". In *Proceedings of IEEE Symposium on Information Visualization (InfoVis)*. p. 135–142.

K. L. Chung and W. Zhuo. 2008. "Graph-Based Visual Analytic Tools for Parallel Coordinates". In *ISVC08 International Symposium on Visual Computing*. p. (poster).

J. H. T. Claessen and J. 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.

C. Collins, F. Viegas, and M. Wattenberg. 2009. "Parallel tag clouds to explore and analyze faceted text corpora". In *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on*. p. 91–98. IEEE.

C. Collins and S. Carpendale. 2007. "VisLink: Revealing relationships amongst visualizations". *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 13, n. 6, p. 1192–1199.

M. Dastani. 2002. "The role of visual perception in data visualization". *Journal of Visual Languages & Computing*, vol. 13, n. 6, p. 601–622.

G. di Battista, P. Eades, R. Tamassia, and I. G. Tollis, 1998. *Graph Drawing: Algorithms for the Visualisation of Graphs*. Prentice Hall.

P. Diaconis and J. H. Friedman, 1980. *M and N plots*. Department of Statistics, Stanford Univ.

M. d'Ocagne, 1885. *Coordonnées parallèles et axiales: méthode de transformation géométrique et procédé nouveau de calcul graphique déduits de la considération des coordonnées parallèles*. Gauthier-Villars.

N. Elmqvist, P. Dragicevic, and J.-D. 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.

Y. Engelhardt. 2002. "The language of graphics: A framework for the analysis of syntax and meaning in maps, charts and diagrams". *Amsterdam: ILLC-Publications*.

G. Falkman. 2001. "Information visualisation in clinical odontology: multidimensional analysis and interactive data exploration". *Artificial Intelligence in Medicine*, vol. 22, p. 133–158.

E. Fanea, S. Carpendale, and T. Isenberg. 2005. "An Interactive 3D Integration of Parallel Coordinates and Star Glyphs". In *Proceedings of IEEE Symposium on Information Visualization (InfoVis)*. p. 149–156.

S. Few, 2004. *Show Me the Numbers: Designing Tables and Graphs to Enlighten*. Analytics Press.

A. Gee, M. Yu, and G. Grinstein. 2005. *Dynamic and interactive dimensional anchors for spring-based visualizations*. Technical report. Technical report, computer science, University of Massachussetts Lowell.

J. Goldstein, S. Roth, J. Kolojejchick, and J. Mattis. 1994. " A framework for knowledge-based interactive data exploration ". *Journal of visual languages and computing*, vol. 5, n. 4, p. 339–363.

F. Guimbretière. January 2003. *Measuring flowmenu performance*. Technical Report CS-TR-4408, UMIACS-TR-2002-88, HCIL-TR-2002-17. University of Maryland.

F. Guimbretière and T. Winograd. 2000. " FlowMenu: Combining Command, Text, and Data Entry ". In *Proc. ACM UIST*. p. 213–216.

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

J. Heer and G. Robertson. 2007. " Animated transitions in statistical data graphics ". *Visualization and Computer Graphics, IEEE Transactions on*, vol. 13, n. 6, p. 1240–1247.

J. Heer, J. D. Mackinlay, C. Stolte, and M. 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.

J. Heinrich, J. Stasko, and D. Weiskopf. 2012. " The Parallel Coordinates Matrix ". In *EuroVisShort2012*. p. 37-41.

N. Henry and J. Fekete. 2006. " Matrixexplorer: a dual-representation system to explore social networks ". *Visualization and Computer Graphics, IEEE Transactions on*, vol. 12, n. 5, p. 677–684.

N. Henry, J. Fekete, and M. McGuffin. 2007a. " NodeTrix: a hybrid visualization of social networks ". *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, p. 1302–1309.

N. Henry and J.-D. Fekete. 2007a. " MatLink: Enhanced Matrix Visualization for Analyzing Social Networks ". In *Proceedings of IFIP TC13 International Conference on Human-Computer Interaction (INTERACT)*. p. 288–302.

N. Henry and J.-D. Fekete. 2007b. " MatLink: Enhanced Matrix Visualization for Analyzing Social Networks ". In *Proceedings of IFIP TC13 International Conference on Human-Computer Interaction (INTERACT)*. p. 288–302.

N. Henry, J.-D. Fekete, and M. J. McGuffin. 2007b. " NodeTrix: A Hybrid Visualization of Social Networks ". *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 13, n. 6, p. 1302–1309.

P. Hoffman, G. Grinstein, and D. Pinkney. 1999. "Dimensional Anchors: A Graphic Primitive for Multidimensional Multivariate Information Visualizations". In *Proceedings of workshop on New Paradigms in Information Visualization and Manipulation (NPIVM)*. p. 9–16. ACM.

P. E. Hoffman. 1999. "Table Visualizations: A Formal Model and Its Applications". Phd, University of Massachussetts.

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

A. Inselberg. 1985. "The Plane with Parallel Coordinates". *The Visual Computer*, vol. 1, p. 69-91.

W. Javed and N. Elmqvist. 2012. "Exploring the design space of composite visualization". In *Pacific Visualization Symposium (PacificVis), 2012 IEEE*. p. 1–8. IEEE.

J. Johansson, P. Ljung, M. Jern, and M. Cooper. 2006. "Revealing structure in visualizations of dense 2D and 3D parallel coordinates". *Information Visualization*, vol. 5, p. 125–136.

D. Keim. 2002. "Information visualization and visual data mining". *Visualization and Computer Graphics, IEEE Transactions on*, vol. 8, n. 1, p. 1–8.

Y. Koren and D. Harel. 2005. "One-dimensional layout optimization, with applications to graph drawing by axis separation". *Computational Geometry*, vol. 32, n. 2, p. 115–138.

G. Kurtenbach, G. Fitzmaurice, R. Owen, and T. Baudel. 1999. "The Hotbox: Efficient Access to a Large Number of Menu-items". In *Proc. ACM CHI*. p. 231–237.

G. Kurtenbach and W. Buxton. 1993. "The Limits of Expert Performance Using Hierarchic Marking Menus". In *Proc. ACM CHI*. p. 482–487.

J. LeBlanc, M. O. Ward, and N. Wittels. 1990. "Exploring N-Dimensional Databases". In *Proceedings of IEEE Visualization (VIS)*. p. 230–237.

J. Li, J.-B. Martens, and J. J. van Wijk. 2010. "Judging correlation from scatterplots and parallel coordinate plots". *Information Visualization*, vol. 9, p. 13–30.

H. Lieberman. 1997. "A multi-scale, multi-layer, translucent virtual space". In *Proceedings of IEEE Conference on Information Visualization (IV)*. p. 124–131.

J. Mackinlay. 1986. "Automating the design of graphical presentations of relational information". *ACM Transactions on Graphics (TOG)*, vol. 5, n. 2, p. 110–141.

J. D. Mackinlay, 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.

H. L. Maria Cristina Ferreira de Oliveira. July-September 2003. "From Visual Data Exploration to Visual Data Mining: A Survey". *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, n. 3, p. 378-394.

M. J. McGuffin and I. Jurisica. 2009. "Interaction Techniques for Selecting and Manipulating Subgraphs in Network Visualizations". *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 15, n. 6, p. 937–944.

T. Mihalisin, 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.

T. Munzner. 2008. "Process and pitfalls in writing information visualization research papers". *Information visualization*, p. 134–153.

T. Munzner. 2009. "A Nested Process Model for Visualization Design and Validation". *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 15, p. 921-928.

T. Nelson. 1999. "Xanalogical Structure, Needed Now More than Ever: Parallel Documents, Deep Links to Content, Deep Versioning, and Deep Re-Use". *ACM Computing Surveys*, vol. 31, n. 4es.

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

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

C. L. North. 2000. "A User Interface for Coordinating Visualizations based on Relational Schemata: Snap-Together Visualization". PhD thesis, Department of Computer Science, Department, University of Maryland, College Park, Maryland.

S. Pook, E. Lecolinet, G. Vaysseix, and E. Barillot. 2000. "Control Menus: Excecution and Control in a Single Interactor". In *CHI'00 extended abstracts on Human factors in computing systems*. p. 263–264. ACM.

H. Purchase, N. Andrienko, T. Jankun-Kelly, M. Ward, A. Kerren, J. Stasko, J.-D. Fekete, and C. North. 2008. Theoretical foundations of information visualization information visualization. *Information Visualization*, volume 4950 of *Lecture Notes in Computer Science*, p. 46-64. Springer Berlin / Heidelberg.

H. Qu, W.-Y. Chan, A. Xu, K.-L. Chung, K.-H. Lau, and P. 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. Rao and S. 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*. p. 318–322. ACM.

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

S. Roth and J. Mattis. 1990. "Data characterization for intelligent graphics presentation". In *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people*. p. 193–200. ACM.

O. Rübel and et al. 2006. "PointCloudXplore: Visual Analysis of 3D Gene Expression Data Using Physical Views and Parallel Coordinates". In *Proceedings of Eurographics/IEEE-VGTC Symposium on Visualization (EuroVis)*.

A. J. Sellen, G. P. Kurtenbach, and W. A. S. Buxton. 1992. "The prevention of mode errors through sensory feedback". *Human Computer Interaction*, vol. 7, n. 2.

B. Shneiderman. 1996. "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualization". In *IEEE Conference on Visual Languages (VL'96)*. (Boulder, CO 1996), p. 336-343. IEEE CS Press.

Y. B. Shrinivasan and J. 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.

C. A. Steed, J. E. Swan II, T. J. Jankun-Kelly, and P. 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.

M. Steinberger, M. Waldner, M. Streit, A. Lex, and D. Schmalstieg. 2011. "Context-Preserving Visual Links". *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 17, n. 12, p. 2249–2258.

S. S. Stevens. 1946. "On the Theory of Scales of Measurement". *Science*, vol. 103, n. 2684, p. 677-680.

C. Stolte, D. Tang, and P. Hanrahan. 2002. "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.

S. T. Teoh, K.-L. Ma, S. F. Wu, and T. J. Jankun-Kelly. 2004. "Detecting Flaws and Intruders with Visual Data Analysis". *IEEE Computer Graphics and Applications (CG&A)*, vol. 24, n. 5, p. 27–35.

J. J. Thomas and K. A. Cook, August 2005. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*.

M. Tobiasz, P. Isenberg, and S. Carpendale. 2009. "Lark: Coordinating Co-located Collaboration with Information Visualization". *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 15, n. 6, p. 1065–1072.

M. Tory and T. Moller. January/February 2004. "Human Factors in Visualization Research". *IEEE Transactions on Visualization and Computer Graphics*, vol. 10(1), p. 72-84.

E. R. Tufte, 1983. *The Visual Display of Quantitative Information*. Graphics Press.

C. Viau and M. J. McGuffin. 2012. "ConnectedCharts: Explicit Visualization of Relationships between Data Graphics". *Computer Graphics Forum*, vol. 31, n. 3, p. 1285-1294.

C. Viau, M. J. McGuffin, Y. Chiricota, and I. 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.

I. Viola, A. Kanitsar, and M. E. Gröller. 2004. "Importance-Driven Volume Rendering". In *Proceedings of IEEE Visualization (VIS)*. p. 139–145.

R. Voigt. October 2002. "An Extended Scatterplot Matrix and Case Studies in Information Visualization. Published as Diplomarbeit". Master's thesis.

M. Waldner, W. Puff, A. Lex, M. Streit, and D. Schmalstieg. 2010. "Visual links across applications". In *Proceedings of Graphics Interface 2010*. p. 129–136. Canadian Information Processing Society.

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

C. Ware, 2004. *Information visualization: perception for design*, volume 22. Morgan Kaufmann.

B. A. Watson, D. Brink, M. Stallmann, R. Devajaran, M. Rakow, T.-M. Rhyne, and H. Patel. 2008. *Matrix depictions for large layered graphs*. Technical Report TR-2008-17. Dept. Computer Science, North Carolina State University.

M. Wattenberg. 2002. "Arc Diagrams: Visualizing Structure in Strings". In *Proceedings of IEEE Symposium on Information Visualization (InfoVis)*. p. 110–116.

M. Wattenberg. 2006. "Visual Exploration of Multivariate Graphs". In *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI)*. p. 811–819.

C. Weaver. 2005. "Visualizing Coordination In Situ". In *Proceedings of IEEE Symposium on Information Visualization (InfoVis)*. p. 165–172.

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

H. Wickham, 2009. *ggplot2: Elegant Graphics for Data Analysis (Use R!)*. Springer.

H. Wickham and H. Hofmann. 2011. "Product Plots". *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 17, n. 12, p. 2223–2230.

L. Wilkinson. 2010. "The grammar of graphics". *Wiley Interdisciplinary Reviews: Computational Statistics*.

L. Wilkinson, A. Anand, and R. Grossman. 2005. "Graph-Theoretic Scagnostics". In *Proc. IEEE Symposium on Information Visualization (InfoVis)*.

P. C. Wong and R. D. 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.

S. Wuchty and E. Almaas. 2005. "Peeling the yeast protein network". *Proteomics*, vol. 5, p. 444–449.

Y. Xu, W. Hong, X. Li, and J. Song. 2007. "Parallel Dual Visualization of Multidimensional Multivariate Data". In *Proceedings of IEEE International Conference on Integration Technology*. p. 263–268.

X. Yuan, P. Guo, H. Xiao, H. Zhou, and H. Qu. 2009. "Scattering Points in Parallel Coordinates". *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 15, n. 6, p. 1001–1008.

J. Zhang. 1996. "A representational analysis of relational information displays". *International journal of human computer studies*, vol. 45, n. 1, p. 59–74.

S. Zhao, M. J. McGuffin, and M. H. Chignell. 2005. "Elastic Hierarchies: Combining Treemaps and Node-Link Diagrams". In *Proceedings of IEEE Symposium on Information Visualization (InfoVis)*. p. 57–64.

C. Ziemkiewicz and R. Kosara. 2009. "Embedding information visualization within visual representation". *Advances in Information and Intelligent Systems*, p. 307–326.