This is a pre-print of:

Dwyer, Tim. "Network visualization as a higher-order visual analysis tool."* IEEE computer graphics and applications *36.6 (2016): 78-85.

Column Series: Visualization Viewpoints
Editor: Theresa-Marie Rhyne

Formatting differs from published article.

Network Visualization as a Higher-Order Visual Analysis Tool

Tim Dwyer
Monash University

Networks are a way to model relationships between people, places, things, ideas, commodities, or really any type of entity you can think of. We can call the relationships "links," and the things they are linking "nodes". The nodes and links may be visualized diagrammatically or using a tabular or adjacency matrix. In the information visualization literature, networks are typically discussed as a first-order data type where the relations are clearly defined by the application. For example, in a rigorous text on visualization analysis and design, Tamara Munzner introduced the "network data type" as arising in applications such as social networks, computer networks, or gene interaction networks.[1] Generally, Munzner positions data visualization as a tool for exploring large quantities of data using fairly straightforward mappings of the data to "marks" and "channels" (such as spatial position and color) and then harnessing the power of human visual perception to identify structure in the data. Such observations allow the viewer to internally build a mental model of the important aspects of the data.

Munzner defines network visualization as a specialization of general data visualization, but one applied specifically to network-structured data. This article examines two aspects of network visualization that blur this distinction and are commonly underappreciated by the information visualization community. Taken together, these two aspects suggest that—in addition to the straightforward visualization of raw network data—there is another role for network views in the visual-analytics workflow.

First, networks can be used to model structural properties in all kinds of non-network data, including quantitative and tabular data. For example, neuroscientists compute functional brain networks from correlated activity detected in different regions of the brain.[2] Work has also begun to explore frameworks for extracting networks from schemas of tabular data.[3,4] However, such operations are by no means standard in general-purpose visual analytics toolkits. In this article, I discuss a use case where complex movements in quantitative data in the finance domain are modeled with small-scale network visualization.

Thus, networks can be derived from other types of data through automatic and semiautomatic processes, and they can be viewed in combination with other data. In either case, the network offers a higher-level view of the data than a visualization of the raw data itself. But we can go further. The second aspect of networks that is underappreciated by the visualization community is that, in addition to modeling data, they can model knowledge.

Munzner and others argue that the "mental model" is the end goal of data visualization. But if we can express this mental model as a network, then we can visualize it as a meta-map of the analyst's exploration. Such a map could link to the various low-level views explored by the analyst as well as their hypotheses and insights. Research in cognitive science has shown that such external representations of knowledge through different types of diagrams fundamentally augment human problem-solving abilities, beyond mere "inputs and stimuli to the internal mind" or "memory aids". [5]

Higher-Order Network Visualization

Expressing complex thought processes as node-link diagrams is not a new idea. People frequently explain or explore their complicated ideas visually,[6] and software exists to assist with this mind-mapping procedure.[7] The opportunity I explore here is the systematic use of network visualization as a kind of externalization of the analyst's thought processes throughout the visual-analytics process. In the following sections, I review several exemplar systems that are distinguished from the textbook network data visualization scenario in the following ways:

- bottom-up rather than top-down data exploration,
- showing the network as a small subset or abstraction of a much larger network data source,
- considering rich multivariate and/or heterogeneous data, and
- showing the network as a conceptual model of the dependencies or relationships between other kinds of data items.

A common thread between these distinct examples is that they all offer views that are closer to highly curated externalizations of the analysts' insights than to views of the raw data. They are higher-order visualizations in the sense that the network visualized is composed from a number of smaller analysis steps, from relationships derived or inferred from raw data, or both.

In a fit of recursion, Figure 1 employs a higher-order network diagram to identify three possible paths for the creation of higher-order network visualizations in the data-analysis process:

1. *Through interactive exploration using standard visualization techniques*. Each node in the higher-order network view corresponds to a distinct insight and may correspond to a whole region of interest in the underlying data. In that case, each node corresponds to a distinct query on the underlying data.
2. *Through more domain specific and possibly nonvisual interactive techniques*. One of the example systems I describe here offers a bottom-up exploration of network data through queries on tabular data. A second example system allows a developer to create a highly curated view of a particular aspect of software while the code remains the primary focus.
3. *By direct translation from the data to a higher-order network view through an automated transformation*. The transformation could be a relatively simple extrapolation of correlated movement in quantitative data, or it could be more sophisticated machine-learning techniques.

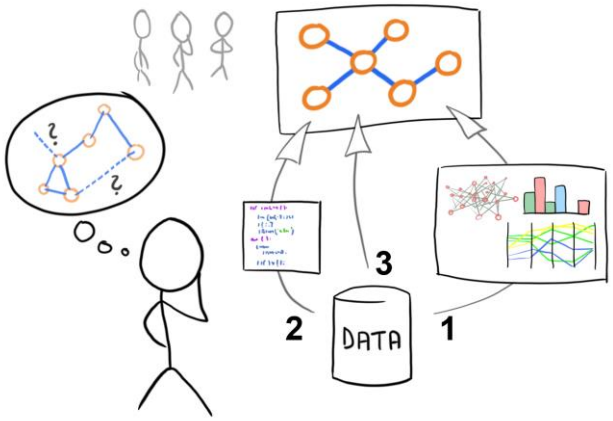

Figure 1. Higher-order network visualizations are externalizations of the analyst's mental model of insights about complex data. This figure identifies three paths to the creation of such a visualization of knowledge: (a) via interactive exploration of the data using standard visualisation techniques, (b) through more domain-specific interactive techniques, and (c) by direct translation from the data to a higher-order network view.

In each case, higher-order network visualizations offer a kind of meta-analysis, helping users reason about what they do and do not know about the data. This aspect may be particularly useful in collaborative analysis scenarios where the diagrams form an externalization of multiple analysts' shared understanding and can facilitate discussion. Similarly, they are useful for communicating complex findings to a wider audience. For example, the people standing in the background in Figure 1 may be a passive audience or may be actively contributing to the higher-order view through their own explorations.

Collectively, these types of higher-order network visualization tools represent an interesting trend that is at odds with the traditional infovis mantra: "Overview-first, zoom-and-filter, then details on demand."[6] Where previously there was pressure for network visualizations (and their supporting technologies such as layout algorithms) to scale to large representations of every node and link, these types of small, focused diagrams have a different set of requirements.

Consider the layout problem alone as an example: layout algorithms in these scenarios need to be able to provide a high-quality layout that can produce diagrams that are as clear as possible; work incrementally to support interaction, user manipulation, and bottom-up construction; and be compact to maximize the area available for nodes to show additional detail. Many aspects of interaction will also differ for such higher-order network visualizations. For example, in the code visualization scenario I describe later on, the graph visualization is no longer the primary artifact; it is an adjunct to the analyst's main focus, which in this case is a code editor.

Example Systems

Figure 2 shows a typical view of an anti-money-laundering analysis scenario using the Influent system.[8] Influent is the latest in a larger family of software products from Uncharted Software, and it builds on ideas established in their more freeform nSpace Sandbox software,[9] which they call a "thinking environment."

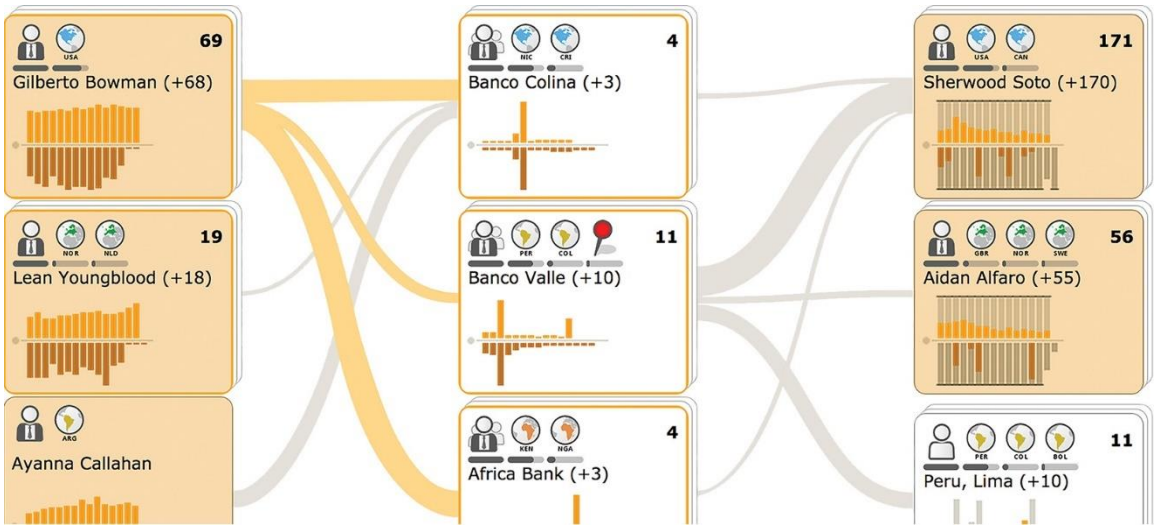

Figure 2. The Influent system supports a bottom-up exploration of the movement of funds between accounts. New account nodes are added only as the result of a direct user query, and the interface makes it easy to collapse links that turn out to be uninteresting to the investigation. (Copyright 2016 Uncharted Software. Used with permission.)

The visualization is built up through an outward (or bottom-up) exploration beginning with accounts belonging to the individual who is the focus of an investigation. The starting point of the exploration involves a domain-specific search with intermediate tabular and geographic representation, making this an example of path 2. Each node consists of a micro chart view of funds coming into and leaving an account. The macro level allows the analyst to follow these transfers into other accounts to look for suspicious activity. The bottom-up construction of these network views means that the visible network is closely aligned to the

analyst's growing understanding of the critical transfers. Each new account node is added only as the result of a direct user query. The interface makes it easy to collapse links that turn out to be uninteresting to the investigation and hence keep the diagram small and focused.

Figure 3, an example of path 1 to higher-order network creation, is dual-view visualization of multivariate network data developed by Stef van den Elzen and Jarke van Wijk.[3] Figure 3a shows the initial view of all data elements (in this case US counties) arranged according to their data attributes (here latitude and longitude). The user can select subsets of these nodes using lasso operations in this attribute plot. Multiple selections may be created. When such a selection is made, a meta-node appears in the diagrammatic view (see Figure 3b). If data elements within a pair of selections are linked in the underlying data, then an arrow also appears in the diagram, showing the cardinality of such links. As the figure shows, various visual summaries of the data attributes within each selection may be displayed within the meta-nodes in the diagram. As the analysis progresses, the diagram gradually grows into a rich infographic representation of the analyst's insights from exploring the data. Also, because each meta node corresponds to a distinct range query (selection) over data attributes, the provenance of the final diagram is easily verified through an interactive interrogation of this mapping to the original data.

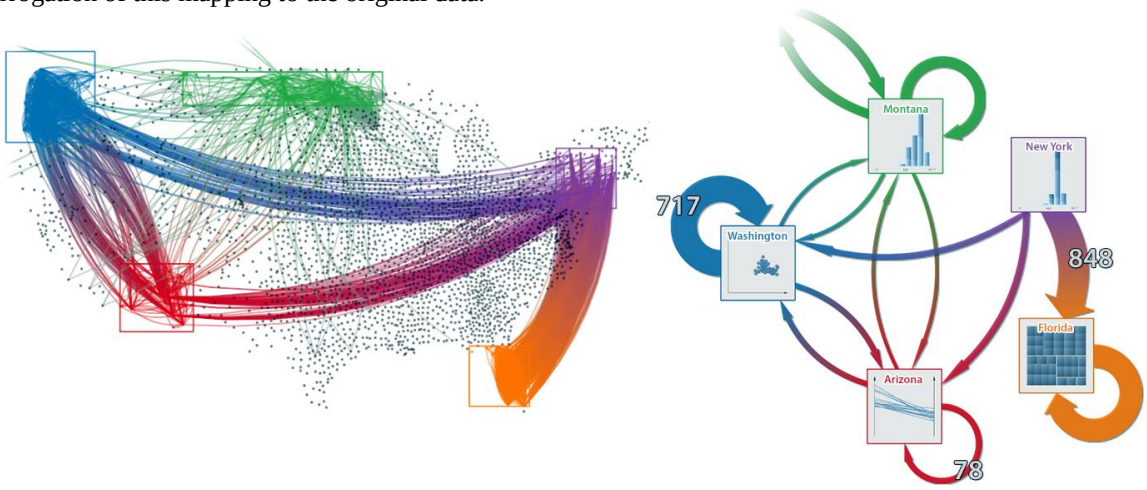

Figure 3. Detail to overview via selections and aggregation. (a) The initial view of all data elements (in this case US counties) are arranged according to their data attributes (here latitude and longitude). (b) Using lasso operations, the user can select subsets of these nodes. If data elements within a pair of selections are linked in the underlying data, then an arrow also appears in the diagram, showing the cardinality of such links. (Courtesy of van den Elzen.[3])

The two systems in Figures 2 and 3 are representative of what can be seen as a growing trend in network visualizations to move beyond simple representations of raw data. In the following sections, I present two additional examples from my personal experience and discuss them in greater detail**.** The first differs slightly from the first two examples in that it is a design for a network visualization that aims to summarize quantitative movement in financial data. This application domain was chosen for this article specifically because practitioners in such fields commonly use charts and other direct mappings of their data to visuals. Indeed, they are typically strong customers of visualization software. Still, financial analysts are typically less accustomed to a network view of their data.

The final example describes a commercial product where network visualization was initially adopted according to fairly standard information visualization principles, as a general way to explore large networks of software dependencies. Over the course of an extended user-centered design process, the tool evolved into something quite different and arguably closer to a tool that allows an analyst to externalize a complex mental model.

From Quantitative Movement to Flow

Fund managers invest in stocks across a variety of industry sectors, adjusting their holdings over time to try

to achieve consistent overall growth in the fund's value. The raw data describing a single fund's composition is a set of time-series data, a sequence of values by stock or industry sector over time. The visualization task to provide a summary view of the significant movements within the fund over time.[10] There are several standard alternatives for visualizing such data. A small-multiples display could show a subset of the most volatile market sectors, or the time series could be stacked into a single chart.

However, our analysts were not just interested in the increases or decreases in value or quantities of stock within an individual market sector. Rather, they wanted to be able to quickly and succinctly know, when money is shifted out of an underperforming industry sector, which other sectors within the fund were the most significant beneficiaries of the manager's reweighting. To better show this higher-level information, I designed the network view in Figure 4. It is an example of higher-order network creation through automatic transformation of underlying quantitative data (hence, path 3).

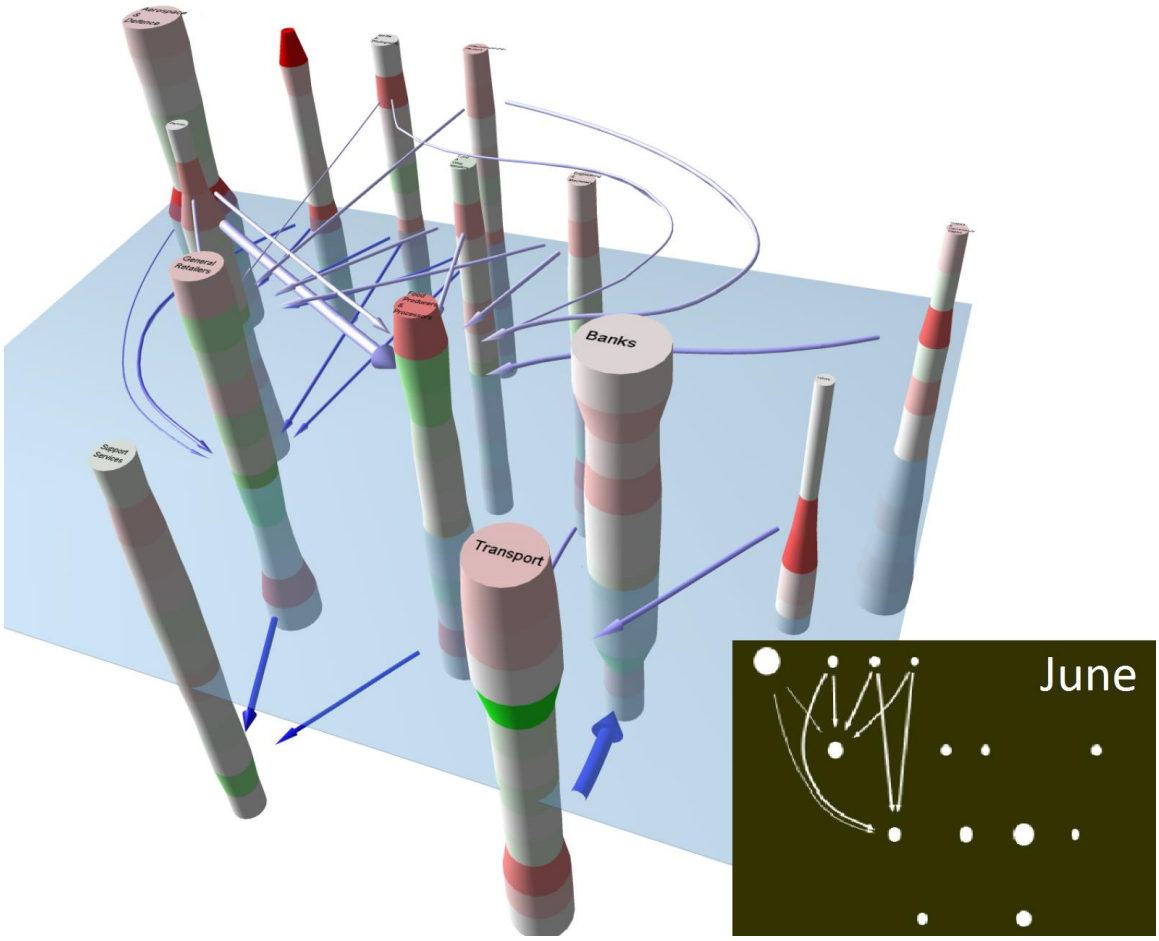

Figure 4. 2.5D network view of a stock fund showing movement over time. (a) Each column represents a market sector, and the diameter of a column changes with the amount invested by the fund in the corresponding market sector. The arrows between the columns indicate the most significant reweightings at a particular point in time. (b) The cross sectional view corresponds to the time period indicated by the level on the columns intersected by the blue plane, or "water level."

The view uses a space-time cube concept such that time increases from bottom to top in the orientation shown (see Figure 4a). Each column represents a market sector. The diameter of a column changes with the amount invested by the fund in the corresponding market sector. Viewed from the side, the columns are like time series charts. The arrows between the columns indicate the most significant reweightings at a particular point in time—in effect, showing a flow of money. Determining which "significant" movements need to be shown with arrows can be quite sophisticated. Thus, this node-link diagram tells a rich story about the movements in the portfolio that an analyst can use to summarize key events over the history of the fund.

Figure 4b shows a cross sectional view for the time period corresponding to the level on the columns intersected by the blue plane, or "water level." This smaller view distinctly highlights one of these key events, movements out of the four most troubled sectors into two safer sectors in a period of particular market volatility.

This network view then neatly summarizes the movements in a way that is intuitive to the analyst in exploration but can also function as a tool that supports presentation of fund performance to investors or regulatory bodies. The transformation of data into visuals that are as effective and engaging as possible for compelling presentation is an emerging topic in information visualization research.[11] Higher-order network flow diagrams can play an important role in data storytelling as a way of summarizing key movements in otherwise complex data.

Code Dependencies from the Bottom Up

Over the decades information visualization researchers and practitioners have developed powerful tools for visualizing large and complex network data in a single view. Fast and clever algorithms have been developed to untangle networks with many thousands of nodes in seconds,[12] and modern graphics hardware can easily render millions of lines or boxes. When people who work with complex systems or data that has an inherent network structure discover these facilities, they become excited about the possibility of seeing their network in its entirety for the first time.

Software engineers are a prime example of practitioners who commonly work with complex systems that are, in essence, huge graphs. For example, software dependency graphs can easily encompass hundreds of thousands or millions of dependency relationships between a variety of software elements. Software practitioners usually only view of their software is the code, so it is exciting for them to obtain a god-like overview of the structure of their entire software system for the first time. Many visualization tools for software engineers have followed this top-down workflow,[13] allowing software engineers to drill in from an overview of the highest-level components (such as separate libraries) to the finest details (such as individual functions or variables).

In such a top-down exploration, the hope is that dependencies that violate best software engineering practice will become apparent. For example, well-designed components should have a clear separation of concerns, reflected by weak coupling in terms of the visible links between them. Again, this is precisely the kind of analysis task for raw network-structured data that is core business for the information visualization workflow.[14]

In 2010 Microsoft shipped an Architecture Explorer feature in the Visual Studio IDE (integrated development environment) that followed precisely such design precepts. While potentially of use to software architects in refactoring projects, in retrospect it was felt that such top-down exploration had little day-to-day application for most users. For the next version of the software, the development team (of which I was a member) decided to add support for a bottom-up exploration of code. The new feature, called Code Map, shipped with Visual Studio 2012. The Code Map feature can be invoked from many different places in the IDE to map out a small neighborhood of dependencies around any element of code. Hence, it is an example of path 2, or the domain-specific bottom-up creation of the higher-order network visualization.

To demonstrate the Code Map implementation, I detail a small scenario: a developer debugging a crash in a Tetris game. The crash occurs when the program attempts to invoke a method on an object reference (called Figure) that is null. To fix the bug, the developer needs to trace through the code and find the reason for Figure being null at that time.

The usual approach to diagnosing such a problem is to repeatedly rerun the program, working backward from the crash site with breakpoints and inspecting state until something suspicious is noticed. However, when a crash is infrequent and inconsistent, repeatedly reproducing it can be time consuming. Therefore, a systematic exploration through the code is preferable, but tracing the various paths through the code can be complex and mentally taxing.

While developing Code Map, we interviewed developers about how they kept track of this exploration and found that none of the methods used were much more sophisticated than note-taking with a pen and paper. Thus, the Code Map tool seeks to automate this visual note-taking as much as possible.

To begin the debug task for the Tetris game, Code Map shows the call stack as a simple chain, side by side with the code editor. These initial call-stack nodes are red in Figure 5. The current stack frame is marked

with a yellow arrow. The null-reference error occurs in the DrawPreview method. The call stack shows that DrawPreview is reached from preview_Paint. Double-clicking on the preview_Paint node in the diagram navigates the code editor to that method, where the developer learns that the null parameter is passed into DrawPreview as the result of the method GetNextFigure. From a context menu or key press, a method for GetNextFigure is added to the diagram.

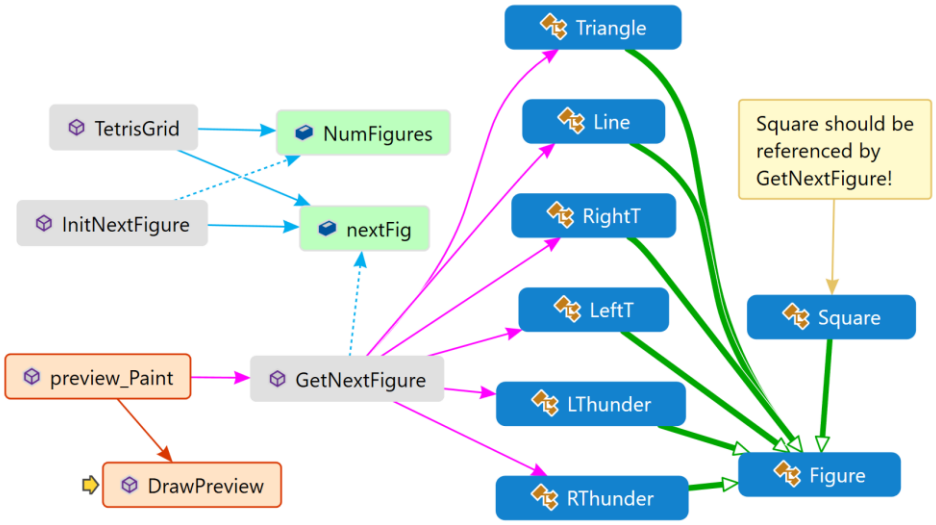

Figure 5. Code Map for the Tetris bug scenario. The initial call-stack nodes are in red, the current stack frame is marked with a yellow arrow, and nodes representing types are in blue. Double-clicking on a node in the diagram navigates the code editor to that method. The lack of any arrow to Square indicates that it is the missing case.

Navigating to GetNextFigure (again by double-clicking on the node in the diagram), the developer finds that GetNextFigure has a switch statement, using the value of an integer field called nextFig to determine a subclass of Figure to instantiate and return. Clicking on the nextFig node in the diagram, she can run a query to show all the places where the field is accessed.

It turns out that nextFig is only modified in two places: once from the class constructor and again from a method call InitNextFigure. Here the developer learns that nextFig is set to a random value in the range of zero to NumFigures. Navigating to the definition of NumFigures in the diagram (again with a double-click), she finds it is set to seven. Is this the correct number of subtypes of Figure? Adding the Figure type to the diagram and running a query of all of its descendant types, she finds that there are seven (nodes representing types are blue in the Code Map). Only six of these are referenced from GetNextFigure, and she sees from the lack of any arrow to Square that it is the missing case. Adding this case resolves the bug. The developer can document this with a comment node.

In addition to adding nodes and links as the result of queries on the underlying dependency graph, users are free to edit the graph to better reflect or document their mental models of the key system function. For example, they can remove nodes associated with code elements when they are realized to be irrelevant to the scope of the search, or they can add arbitrary nodes containing notes, as we did in this example. Thus, the diagram can be submitted along with code changes to assist with review or be kept as documentation.

Discussion

I have explored several network visualization use cases that I consider a higher-order form of visual analytics than raw data visualization. In each case, the networks are relatively small and highly abstracted from the data, either manually by the analyst or by an automatic transform, to represent insights rather than raw data. In Figure 1, I identified three distinct paths for producing these types of higher-order network diagrams: from lower-level data visuals, from domain-specific views, and through automatic transformation from the data. I chose specific examples that illustrated each of these paths. However, the three paths need not be mutually

exclusive and, ideally, would be seamlessly combined into an exploratory system.

For example, higher-order network visualizations can perform several functions, providing (again, not mutually exclusively) the following:

- A visual summary can tie together key data analysis insights and support the analyst's own reasoning process or communication with other collaborators or a more passive audience (data storytelling).
- A navigational aid can provide a history of past queries or landmarks in exploration. This lets an analyst recall or audit the provenance of important findings.
- A minimal subset or abstraction of data may be too granular to clearly see the most significant movements or relationships at full detail.

Additional Context

There is strong precedent for modeling knowledge (or insights, in data analytics terms) as networks in Semantic Web research. There are significant efforts to automatically compile huge networks of facts from automatic reading of the Web (such as the Never-Ending Language Learning Project[15]) or manual annotation of documents.[16] Visualization of the Semantic Web, or knowledge visualization, is also an active area of development—for example, Alberto Cañas and his colleagues developed a formalism for this called concept maps.[17]

Others have spoken in general terms about the need for visualization of various aspects of the visual analytics process itself. For example, Min Chen and his colleagues proposed a high-level architecture for visual analytics systems that feature a knowledge-supporting infrastructure with its own visual pipeline, parallel to the direct data visualization.[18] The kind of higher-order network visualizations I have described slot quite neatly into this framework.

Part of our discussion in this article is about the transformation of different types of data to a network model. To further place this discussion within a broader context, David Kasik and his colleagues discussed the space and import of various data transformations in the visual analytics process and warn that "there is no single transformation or representation method to uniformly address all data issues."[19] There are certainly visual forms other than node-link diagrams that could be used to convey high-level or abstract information about the data or the analyst's insights. Indeed, three of the four systems highlighted in our discussion here feature different types of quantitative data summaries within their nodes. Thus, the diagrammatic node-link form is an abstract and powerful starting point that can model many different semantics as well as being easily augmented with these types of annotations. Still, a more complete discussion of higher-order visual analytics would also need to consider alternatives such as nested, tiled, or tabular views. Alternatively, we could broaden the definition of network diagrams to be any spatial arrangement that implies relation.

This article has argued for the potential of network diagrams to fulfill a higher-order role of visualizing insights about the data rather than a direct representation of the data itself. I have described different paths from the data to such a diagram, both supervised and unsupervised. A logical middle-ground that has not been discussed is semisupervised higher-order network visualization construction, where the construction of the higher-order view is a collaboration between human and automated techniques.

In the future, a formal exploration of the design and application spaces for higher-order network and other types of rich diagrammatic visualization would be as a fruitful direction for our research community. Doing so could lead to tools that support every aspect of the visual-analytics workflow, supporting not only the search for insights in raw data, but also helping the analyst to build logical connections between those insights. In other words, the externalization of the analysts' thought process could support more systematic initial exploration, but also communication of results and ongoing collaborative data analytics.

Acknowledgments

I thank the reviewers of this article for their insightful feedback and suggestions, Richard Brath and Uncharted Software for permission to use the Influent screenshot in Figure 2, Steph van den Elzen giving us permission to reproduce Figure 3, my colleagues on the original Code Map Team at Microsoft, and Tom Bochynek for his assistance with Figure 1.

References

1. T. Munzner, *Visualization Analysis and Design*, CRC Press, 2014.
2. B. Bach et al., "Small Multipiles: Piling Time to Explore Temporal Patterns in Dynamic Networks," *Computer Graphics Forum*, vol. 34, no. 5, 2015, pp. 31–40.
3. S. van den Elzen and J. van Wijk, "Multivariate Network Exploration and Presentation: From Detail to Overview via Selections and Aggregations," *IEEE Trans. Visualization and Computer Graphics*, vol. 20, no. 12, 2014, pp. 2310–2319.
4. Z. Liu, S.B. Navathe, and J.T. Stasko, "Ploceus: Modeling, Visualizing, and Analyzing Tabular Data as Networks," *Information Visualization*, vol. 13, no. 1, 2014, pp. 59–89.
5. J. Zhang, "The Nature of External Representations in Problem Solving," *Cognitive Science*, vol. 21, no. 2, 1997, pp. 179–217.
6. D. Shneiderman, "A Grander Goal: A Thousand-fold Increase in Human Capabilities," *Educom Rev*., vol. 32, no 6, 1997, pp. 4-10.
7. T. Buzan, *Mind Mapping*, Pearson Education, 2006.
8. D. Jonker et al., "Influent: Scalable Transaction Flow Analysis with Entity-Relationship Graphs," poster, IEEE EuroVis 2014.
9. J. Walny et al., "Visual Thinking in Action: Visualizations as Used on Whiteboards," *IEEE Trans. Visualization and Computer Graphics*, vol. 17, no. 12, 2011, pp. 2508–2517.
10. T. Dwyer, "Two-and-a-Half Dimensional Visualization of Relational Networks," PhD dissertation, Univ. of Sydney, 2005.
11. B. Lee et al., "More Than Telling a Story: Transforming Data into Visually Shared Stories," *IEEE Computer Graphics and Applications*, vol. 35, no. 5, 2015, pp. 84–90.
12. Y. Hu, "Efficient, High-Quality Force-Directed Graph Drawing," *Mathematica J*., vol. 10, no. 1, 2005, pp. 37–71.
13. F. van Ham, "Using Multilevel Call Matrices in Large Software Projects," *Proc. IEEE Symp. Information Visualization* (*INFOVIS*), 2003, pp. 227–232.
14. B. Lee et al., "Task Taxonomy for Graph Visualization," *Proc. 2006 AVI Workshop on Beyond Time and Errors: Novel Evaluation Methods for Information Visualization* (*BELIV*), 2006.
15. A. Carlson et al., "Toward an Architecture for Never-Ending Language Learning," Proc. 24th Conf. Artificial Intelligence (AAAI), 2010, pp. 1306–1313.
16. P. Ciccarese, M. Ocana, and T. Clark, "DOMEO: A Web-Based Tool for Semantic Annotation of Online Documents," *Bio-Ontologies*, 2012; http://bio-ontologies.knowledgeblog.org/297.
17. A.J. Cañas et al., "Concept Maps: Integrating Knowledge and Information Visualization," *Knowledge and Information Visualization*, LNCS 3426, Springer, 2005, pp. 205–219.
18. M. Chen et al., "Data, Information, and Knowledge in Visualization," *IEEE Computer Graphics and Applications*, vol. 29, no. 1, 2009, pp, 12–19.
19. D.J. Kasik et al., "Data Transformations and Representations for Computation and Visualization," *Information Visualization*, vol. 8, no. 4, 2009, pp. 275–285.

Tim Dwyer *codirects the Immersive Analytics Initiative at Monash University. Contact him at tim.dwyer@monash.edu.*

Contact department editor Theresa-Marie Rhyne at theresamarierhyne@gmail.com.

//digital library abstract & keywords//

The transformation of data into visuals that are as effective and engaging as possible for compelling presentation is an emerging topic in information visualization research. With this goal in mind, several data-analysis system approaches offer a higher-order network visualization that map insights rather than raw data. Knowledge visualization or externalization in the form of small, focused diagrams can communicate insights as well as help analysts structure their reasoning process.

computer graphics, visualization, visual analytics, network visualization, information visualization

orig-research