2 Literature Review

In this chapter, we first discuss the core work in individual fields related to our research problem including sensemaking, visualization and analytic provenance. Then, we focus on the intersection of these three fields: visualization of provenance data for supporting sensemaking. At the end, we present visualization techniques of general time-oriented and network data because these two data types possess similar characteristics as provenance data.

2.1 Sensemaking

Sensemaking reflects how we make sense of the world so that we can act in it [160]. Sensemaking has been studied in different contexts such as information science [41], human-computer interaction [150], organizational studies [184], and intelligence analysis [95, 135]. In this section, we review the sensemaking concepts discussed in these contexts, with an emphasis on the last two sensemaking models that have been highly applied in the visualization community. A recent and comprehensive review of sensemaking can be found in the article by Maitlis and Christianson [108].

2.1.1 Gap-Bridging Metaphor

Dervin develops a sensemaking theory focusing on information seeking and use behaviors [41]. It underlies the cognitive gap that individuals experience when attempting to make sense of observed data. Figure 2.1 summarizes this *gap-bridging* metaphor. The theory assumes that people moves through time-space in some particular context and situation. Sensemaking starts when they encounter a gap that needs to overcome such as something is unclear or confused. To bridge the gap, they

may seek and use information from a variety of sources such as documents, media and other people. These sources are evaluated based on relevant attributes to assess their usefulness.

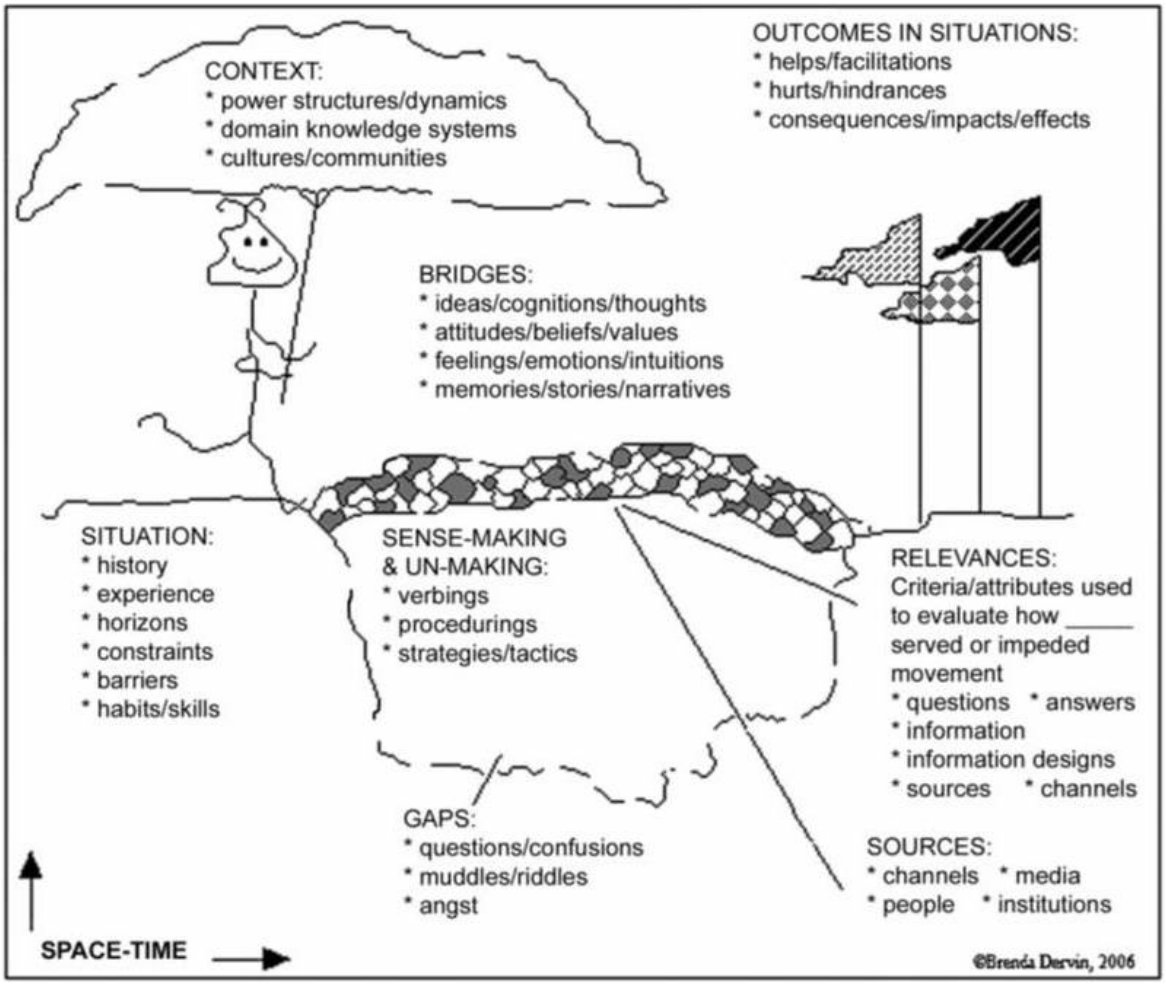


Figure 2.1: The gap-bridging metaphor of sensemaking. People encounter gaps when moving through time-space, then seek for information, evaluate and use it to bridge the gaps. *Image source: [42].*

Dervin also implements the theory into a set of questions that can be used in interview to understand sensemaking within a context [41]. The questions elaborate all parts of the model, aiming to establish an understanding on the situation (*What happened?*), the gap (*What did you struggle with?*), the bridge (*What idea did you come to?*) and the outcome (*How did that help?*).

2.1.2 Learning Loop Complex

In the context of human-computer interaction, Russell et al. [150] defines sensemaking as the process of searching for a representation and encoding data in that representation to answer task-specific questions. That cyclic process is called the *learning loop complex* as illustrated in Figure 2.2. First, the sensemaker searches for a representation to capture salient features of the data (*Generation Loop*). During sensemaking, new information is sought and encoded into this representation (*Data Coverage Loop*). The data unfit to the representation (*residue*) requires the sensemaker to adjusts and produces a more suitable one. This entire learning loop complex is guided by the task with an aim to reduce its cost.

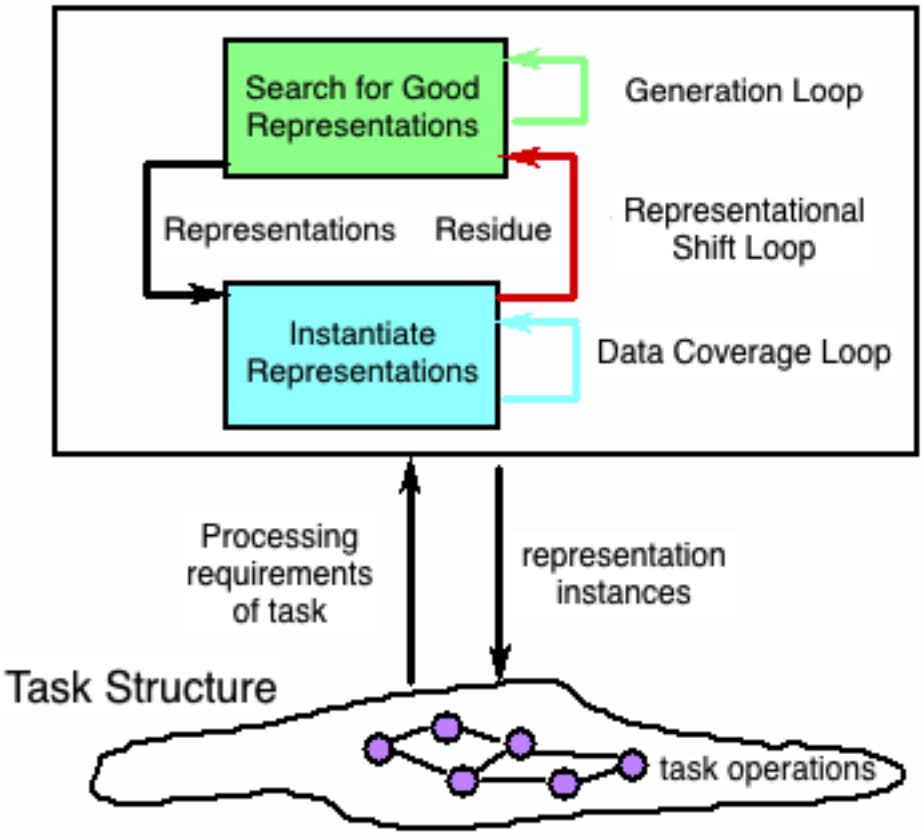


Figure 2.2: The learning loop complex theory of sensemaking. It consists of three iterative loops: searching for a good representation, encoding data to the representation, and adjusting the representation for a better data coverage. *Image source: [150].*

2.1.3 Sensemaking in Organizations

Different from Dervin and Russell who study sensemaking for individuals, Weick focuses on sensemaking at an organization level [184]. He proposes that sensemaking consists of these seven following properties.

1. *Grounded in identity construction*. Who people think they are, both individually and collectively, affect what they interpret and act.
2. *Retrospective*. People look back and make sense from what they have said and what they have done before.
3. *Enactive of sensible environments*. People make sense and contribute to the environments during their sensemaking processes.
4. *Social*. This is an inherent property of sensemaking in organization where people interact and socialize with others, and also are influenced by others.
5. *Ongoing*. Sensemaking is a continuous flow because the world and our understanding about the word are constantly changing.
6. *Focused on and by extracted cues*. Cues are things from the context that people have attention to and may use them to guide further exploration and assessment of the sensemaking problem.
7. *Driven by plausibility rather than accuracy*. Sensemaking is about plausibility and sufficiency rather than accuracy and completeness. People tend to stop searching when they find an acceptable solution.

2.1.4 A Process Model

Pirolli and Card [135] describe sensemaking as an iterative process that gradually transforms raw data into rational knowledge. The process includes two sets of activities: one that cycles around finding relevant information, and another that cycles around making sense of that information, with plenty of interaction between them. They map to the *foraging loop* and the *sensemaking loop* respectively, as shown in Figure 2.3. The sensemaking process can progress upward (from data to knowledge) or downward (from knowledge to data). The steps in the *bottom-up* process are summarized as follows.

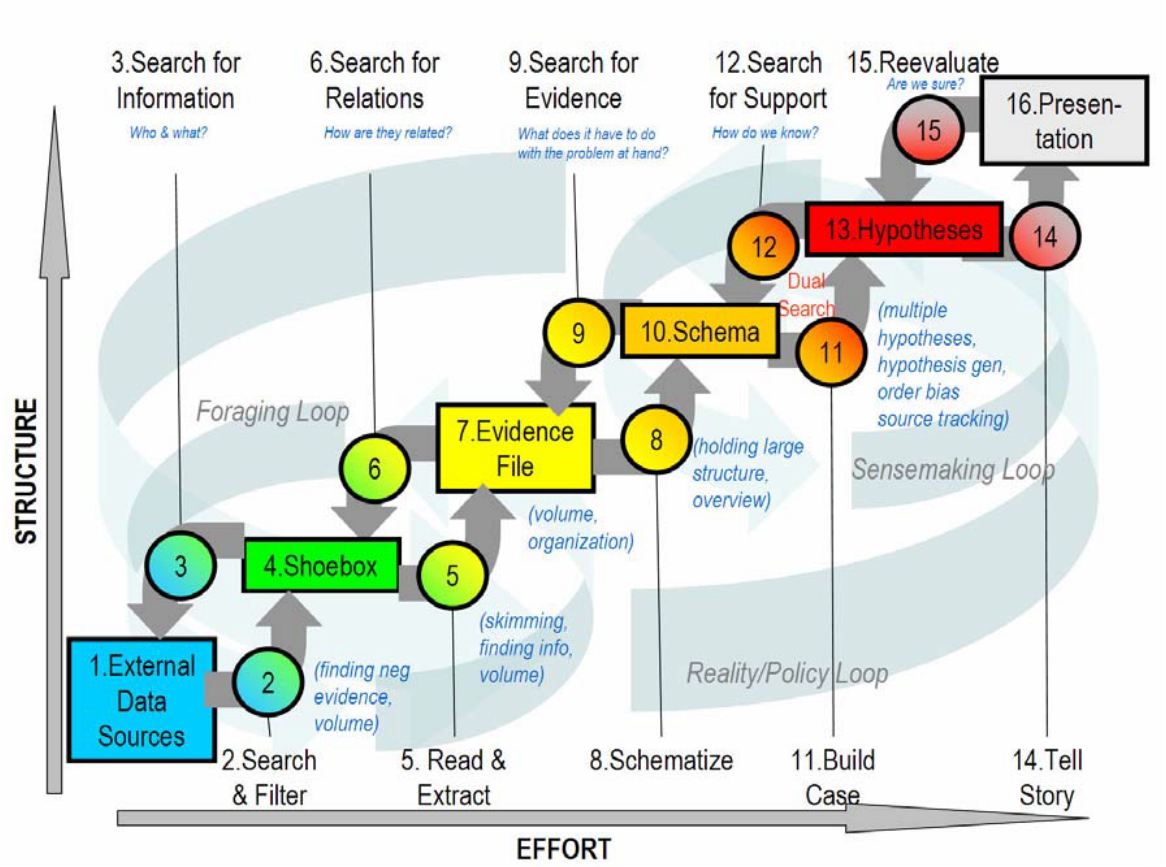


Figure 2.3: A notional model of sensemaking. The sub-processes (numbered circles) and their data input/output (numbered rectangles) are arranged in a two-dimensional space, in which the horizontal axis represents the degree of effort from users, and the vertical axis represents the degree of structure in information representation. *Image source: [135].*

- *Search and filter*. External data sources, such as classified databases or the web, are searched and filtered to retrieve relevant documents to the task.
- *Read and extract*. These documents are examined to extract pieces of information that may be used as evidence later.
- *Schematize*. The collected information is organized in a way that aids the analysis. This may be executed in user mind, using paper and pen, or with a complex computer-based system.
- *Build case*. Multiple hypotheses are generated; evidence are marshaled to support or disconfirm them.
- *Tell story*. Discovered cases are presented to some audience.

In this model, *schematization* plays an important role in converting raw evidence to rational explanations, bridging the foraging and sensemaking loops. A study by Kang, Görg and John Stasko [89] agrees with this observation. In their study, all the participants who performed the sensemaking task well spent considerable time and effort in organizing their collected information. Their organizational schemes were flexible: a *timeline* of related events, a *map* connecting locations that a person has been to, and a *diagram* showing relationships among suspicious targets.

2.1.5 Data–Frame Model

Klein et al. [95] propose a sensemaking model that centers around *data* and *frame*. Data is the information that a person receives or searches for, and frame is the mental structure that organizes and explains the relationship of such data. For instance, a frame can be a *story*, explaining the chronology of events and the causal relationships between them; or a *map*, showing where the events take place and the routes between them. Sensemaking is considered as a deliberate effort to understand an event, starting when a person realizes a gap of their current understanding of that event. Klein and his associates describe seven activities involved in sensemaking and are summarized in Figure 2.4.

- *Connect data and a frame*. A person recognizes relevant pieces of data and constructs an initial frame to explain them. The frame then helps the person to filter and search for new data.
- *Elaborate the frame*. As more is learned about the situation, the frame becomes more elaborate with new data and new relationships.
- *Question the frame*. It happens when a person encounters data that is inconsistent with the existing frame. At this point, the person may be unsure that the frame is incorrect, or the inconsistent data is inaccurate.
- *Preserve the frame*. A person may consider the severity of the inconsistent data, justify why it mismatches the frame, and ignore it.
- *Compare multiple frames*. Depending on experience, a person may think of alternative frames explaining the same set of data. These frames need to be compared to select the most likely one.

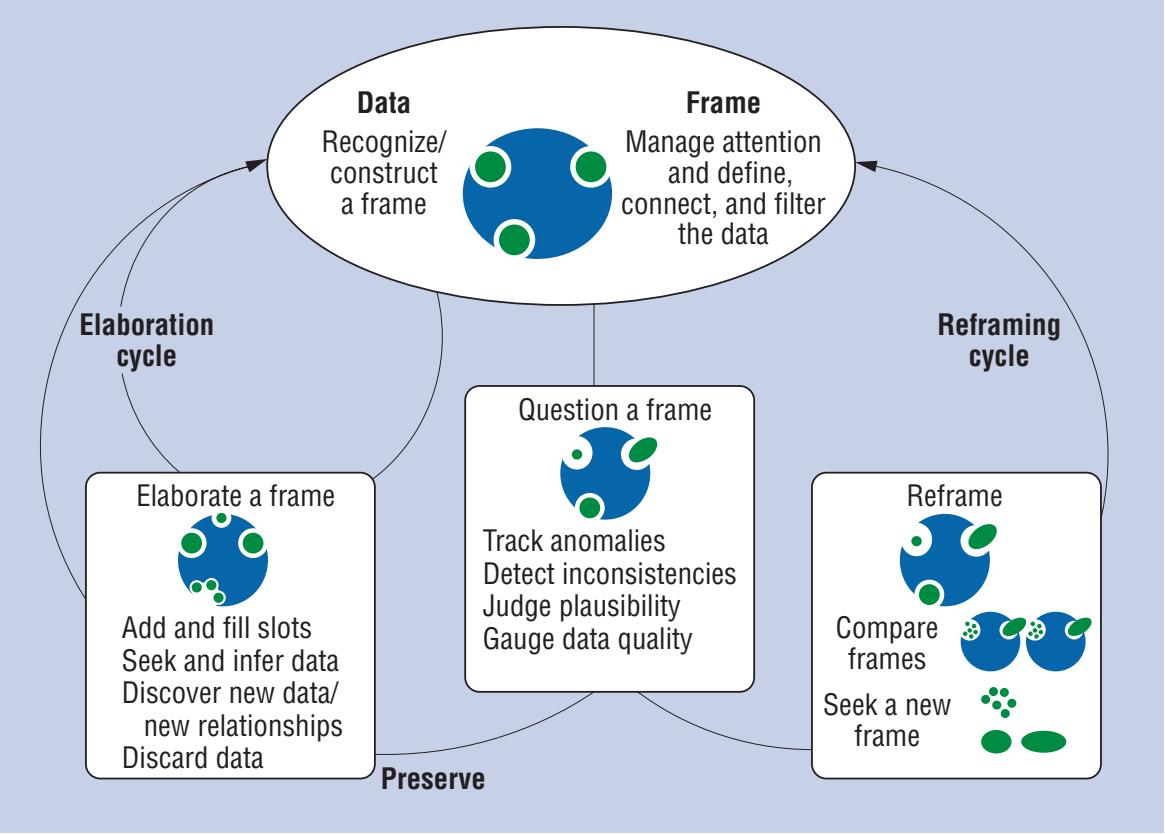


Figure 2.4: The data–frame model of sensemaking. It describes a set of interconnected sensemaking activities centering around data and frame – the explanatory structure of data. *Image source: [95].*

- *Reframe*. When encountering inconsistent and contrary data, the person may need to find a replacement that can explain all data. Considering discarded data and/or reinterpreting data could facilitate this activity.
- *Seek a new frame*. A person may deliberately search for a new frame when encountering plenty of conflicted data. One or two key data elements may serve as *anchors* to help the person to elicit another frame.

The Pirolli and Card's model describes a step-by-step process of sensemaking, in which the analyst collects relevant data and eventually transposes it into rational answers. However, the various sensemaking activities in the Data–Frame model may explain the strategies used by the analyst more comprehensively.

2.2 Provenance

In the Oxford dictionary, *provenance* is defined as "the place of origin or earliest known history of something". Provenance plays an important role in many aspects of our daily lives. For example, in *food* industry, before purchasing a bottle of fruit juice, it is useful for the customers to know about its origin, ingredients, methods of collecting, storing and processing fruits, etc. In the context of *art*, the provenance information of a painting such as authorship, material, painting time and the story behind the painting greatly decides its value. In computer systems, the provenance of a piece of data is defined as "the process that led to that piece of data" [116]. We broadly categorize provenance into two groups. The first group, *data provenance*, uses the computer systems definition of provenance including the source information of data and the process that produced it. The second group, *analytic provenance*, focuses on visualization and analysis including the higher level reasoning involved in the analysis process and the interactive data exploration driven by sensemaking.

2.2.1 Data Provenance

Data provenance research has been taken in different fields, notably scientific workflows and databases. Scientific experiments may consist of thousands of steps, with each step involving distributed data sources and computational data models [58]. Workflows have been used to facilitate the assembly, automation and management of such experiments. Notable scientific workflow systems with provenance enabled include Tarvena [196], Kepler [16] and VisTrails [13]. Provenance plays an important role in scientific workflows, aiming to support data interpretation, reproduction of experiment results, troubleshooting and optimization [114]. The provenance of long and complex workflows is huge, thus pose challenges in storing, querying, and making sense of such data [39].

Curated databases are populated and updated with a great deal of human effort, typically published on the web [19]. A well-known example is Wikipedia – a free Internet encyclopedia that allows its users to edit almost any article accessible. Each record in these databases, such as a Wikipedia article, may be edited by many users and referred to other internal and external sources. This produces problems in attribution and provenance: who edited what at when. Research in database provenance can be characterized into a why-where-how framework [27]. *Why*-provenance focuses on the lineage of the output: for each tuple t in the output, the lineage of t is a set of tuples in the input data that helps produce t [38]. *How*-

provenance concerns how the output tuple t is derived from the query [64]. Finally, *where*-provenance describes specific locations, or cells in relational databases, of the input data that contribute to the query output [20]. To compute these types of provenance, two general approaches have been introduced [19]. An *eager* approach adjusts the query to pass the extra provenance information to the output. Whereas, a *lazy* approach computes provenance on demand.

Data provenance research in scientific workflows and databases has mainly focused on closed systems, which have full access to the data and its provenance. Modern applications with service-oriented [129] and cloud-computing [23] architectures bring challenges in tracking and exchanging provenance information across systems. The *Open Provenance Model* is designed to address these challenges [116]. It also supports a digital representation of provenance for any objects, whether produced by computer systems or not. Three types of objects are defined in the model for building this representation. An *artifact* is a state that can be a digital or physical object. A *process* is a series of actions performed on or caused by artifacts, and resulting in new artifacts. An *agent* acts as a catalyst of a process, managing its execution. Different types of causal relationships can be added between these nodes, forming a *provenance graph* as shown in Figure 2.5.

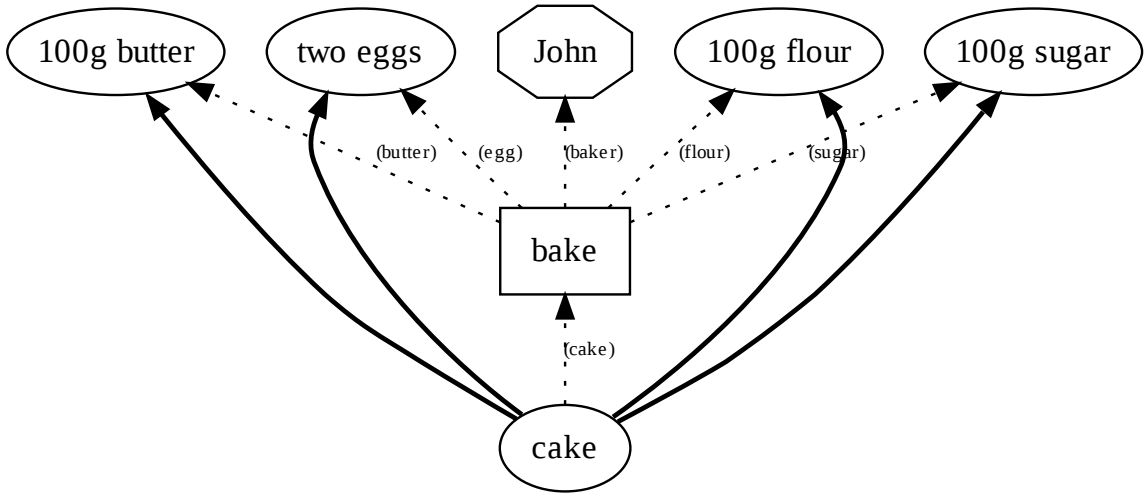


Figure 2.5: A provenance graph for "cake baking" using the Open Provenance Model. The cake (artifact) was baked (the process) by John (the agent) using ingredients including butter, eggs, flour and sugar (artifacts). *Image source: [116].*

The Open Provenance Model has been implemented in many systems, including notable scientific workflows such as Tarvena [196], Kepler [16] and VisTrails [13]. The model is general and can be extended in both the structure and vocabulary

to represent domain-specific problems. *D-profile* [65] describes an extension of the model for representing provenance in distributed systems. ProveML [178] is another extension for recording the provenance of data, analytical process and interpretations in human terrain visual analytics.

2.2.2 Analytic Provenance

Analytic provenance focuses on understanding a user's reasoning process through the study of their interactions with a visualization [126]. It captures both the interactive data exploration process and the accompanied human reasoning process during sensemaking [191]. In this section, we discuss different models for representing analytic provenance data and methods to capture the data.

2.2.2.1 Model

Analytic provenance information can be categorized using a four-layer hierarchical model based on its semantic richness, proposed by Gotz and Zhou [62]. Figure 2.6 illustrates this model with the level of semantics increasing from bottom to top. The bottom-level *events* consists of low-level user interactions such as mouse clicks and keystrokes, which have little semantic meaning. The next level up includes *actions*, which are analytic steps such as querying the database or changing the zooming level of data visualization. The parameters such as data description and visualization settings are also part of the provenance. Further up are the *sub-tasks*, which are the analyses required to achieve the sensemaking goal. Considering stock market analysis as the top-level *task*, examples of sub-tasks could be identifying top performing companies and determining long term trends.

Analytic provenance is closely linked both within and across layers. Within a layer, analytic provenance is linked temporally (i.e., one event happens after another) and logically (e.g., one action depends on the two previous actions). A sequence of activities in each layer is performed to serve for a single activity in the next highest layer. For example, a database query action consists of several mouse click and key stroke events, and it is part of a higher level sub-task such as "comparing stock performance".

This four-layer model is general, allowing the designers to determine the specific activities they want to capture within each layer for their systems. For example, many existing taxonomies of visualization interaction and tasks can be used for the *action* and *sub-task* layers. Low-level analytic activities such as "retrieve value",

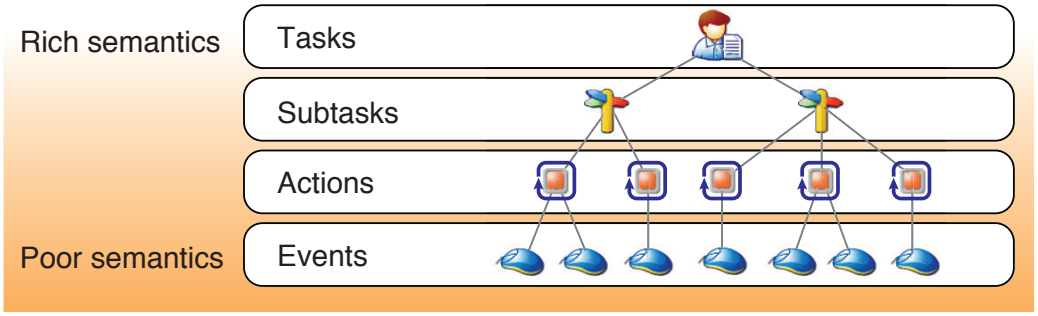


Figure 2.6: The hierarchical analytic provenance model with semantic richness increasing from bottom to top. The bottom layer includes *events* such as key presses and mouse clicks, which have little semantics. The next level up contains *actions* such as the database query and visualization zooming. Further up level consists of *sub-tasks*, which usually are the analyses performed during the sensemaking. The top level *tasks* are the overall sensemaking undertaking. *Image source: [62].*

"sort" and "filter" [9, 62] can represent the *action* layer. A taxonomy of interaction techniques based on their intent such as "show me more related items" may stay in-between the *action* layer and *sub-task* layer. A multi-level typology of abstract visualization tasks by Brehmer and Munzner [17] distinguishes why and how a task is performed, as well as what the task inputs and outputs are. The *how* part of this typology is a good candidate for representing the *action* layer, and the *why* part may be suitable for the *sub-task* layer.

A recent taxonomy by Ragan et al. [141] characterizes provenance in visualization and analysis systems based on the type of provenance and the purpose of collecting it. Five "flat" provenance types include *data*, *visualization*, *interaction*, *insight* and *rationale*. The type of data provenance is similar to what we describe previously in Section 2.2.1. Visualization provenance concerns with the history of graphical views and visualization states, and interaction provenance focus on the history of user actions within a system. The combination of these two types of provenance is equivalent to the *action* layer. Insight provenance describes the findings and rationale provenance explains the reasoning process that led to these findings. These two types of provenance are good candidates for the *sub-task* layer.

2.2.2.2 Capture

Analytic provenance capture provides the data for its visualization. We describe capturing methods for each of the four layers in the Gotz and Zhou's model [62].

Events There is limited literature on capturing events because it is relatively easy and provides little semantics alone. Among these, Glass Box [37] can record a wide range of low-level events such as mouse clicks, key strokes and window events. Its objective is to capture and archive intelligence analysis activities so they can be retrieved later. Simply capturing these events alone is insufficient to understand their purpose and rationale. For example, we know that a *mouse click* is captured; however, what the purpose of that click was (e.g., to sort the data), and why the user performed that click (e.g., to find an interesting pattern from the data) are unknown. Commonly, when analyzing data with a visualization, an analyst needs to perform many operations with trials and errors to find the answer to the problem. In that case, a series of poor-semantic events makes it more difficult for the analyst to recall what has been done. Therefore, more meaningful activities also need to be captured.

Actions During the course of analysis with a visualization, all user interactions can be systematically recorded. The visual exploration process can be modeled using a *graph* metaphor. Nodes in the graph represent *states* of the visualization and edges represent *actions* that transform one state into another state. A state includes all the information required to reconstruct the captured visualization. For example, a state of a *scatter plot* may include the dataset, attributes mapped to spatial positions, color and size. An action while interacting with a scatter plot could be changing attribute mapping of size. Commonly, *undo/redo* features are provided to allow revisiting to previous states. If a new action is performed from a past state, a new branch will be created to store that new line of operations.

Two common strategies can be applied to automatically capture the exploration process. The first strategy is to capture the initial state and all the actions so that they can be executed to reproduce any states [88]. This allows reapplying the analysis process with a different dataset, but could be time-inefficient if the number of actions is high. The second strategy is simply to capture all visualization states after each action [13]. This is easier to implement, but may be memory-expensive if a state contain too much information.

In the context of everyday, online sensemaking, users interact with a standard web browser instead of a visualization system. Visited web pages are automatically captured including visit time, web titles, URLs and favorite icons. Linking relationship between pages such as opening from a web link and using the browser's back button can also be captured [12, 81, 115]. This results in a hierarchical history rather than a linear list of visited web pages in the standard history feature. Manual

capture using bookmarks is also a standard feature of web browsers. It allows users to save web pages for revisitation purpose. Besides bookmarking a whole web page through its URL, a page element such as *table* and *form* HTML tags [84], and a specific fragment of text [44] can also be bookmarked. These finer-grained captures allow users to record what they want with higher accuracy.

Sub-Tasks and Tasks Tasks and sub-tasks provide important clues to the purpose and rationale underlying the sensemaking. They describe an abstract summary of the task-solving process. However, they are largely part of users' thinking, which a visual analytics system does not have direct access to. Also, the time window to capture such information is very limited. Even the users themselves may forget what they were doing after a while, making it difficult to recover the analytic provenance information. Therefore, capturing high-level tasks and sub-tasks is one of the biggest challenges in analytic provenance capture.

Existing approaches to capture high-level analytic provenance can be broadly categorized into *manual* and *automatic* methods. The manual methods largely rely on users recording their analysis processes and sensemaking tasks, whereas the automatic methods try to infer the higher level tasks and sub-tasks from lower level events and actions. Even though the manual approaches are usually more accurate, they can distract users from the actual analysis tasks, which may discourage users from recording analytic provenance. Alternatively, the automatic approaches do not introduce interruption to the sensemaking process, but their capability of inferring semantic-rich analytic provenance information is limited [62]. Individual differences also introduce additional challenges to designing a robust algorithm for the inferring process. Users' knowledge and experience have a considerable impact on the way they conduct analyses. As a result, the sensemaking process (i.e. the analytic provenance) can vary significantly from users to users, even with the same dataset and analysis task.

Manual User annotation is one of the most common forms in manual capture. Users create *notes* or *annotations* that are associated with certain data, analysis result, and/or visualization [76, 178]. The content of a "note" is not limited to findings or discoveries; it can also include the thinking that leads to a finding or the relationships between findings. *Data-aware annotation* [71] links a finding and the associated visualization to the underlying data used to produce it, which makes it possible to

apply new analysis and visual mapping at a later stage if further investigation is needed.

Even though an individual note only represents a fraction of the analytic provenance, it is possible to provide a reasonably good overview of the sensemaking process if a number of notes and the connections between them are captured. For example, the *Scalable Reasoning System* [133] allows users to record their discoveries of interesting patterns about the data, then specify their causal relationships, and generate a hypothesis based on these artifacts. However, this is only useful when users are willing to take notes, which can be perceived as distractions sometimes. Two common strategies to alleviate this include minimizing interruption/cognitive effort [84] and providing immediate benefits to the sensemaking task such as planing exploratory analysis for complex tasks [106]. However, currently there is a lack of general design guidelines for how to achieve them, and there are few user studies evaluating how effective they are, in terms of both the benefits they bring and the potential cognitive cost they can introduce.

Automatic One of the main disadvantages of manual capture is the requirement of direct input from users. Automatic approaches try to address this by inferring higher level analytic provenance from what can be automatically captured. As discussed earlier, it is easier to capture analytic provenance at the event and action level. Therefore, most automatic approaches try to infer sub-task and task-level information from event and action provenance.

This turns out to be a difficult challenge. An experiment studied how much of a user's reasoning process can be recovered from user action information [45]. A domain-specific sensemaking task was used and experts were recruited to analyze the user action log. Higher-level analytic provenance manually inferred from the interaction logs were compared with the ground truth obtained through interview. The results showed that 79 percent of the findings, 60 percent of the methods, and 60 percent of the strategies were correctly recovered. The accuracy is not high even in such a constrained setting with domain experts doing the inference. Given the diversity of data and analysis involved in the sensemaking and the difficult of replicating expert knowledge/thinking in a computer system, the chance of having a generic technique that can accurately infer semantic-rich analytic provenance information for a variety of analysis tasks is not high.

Instead, existing methods either constrain the problem/analysis domain or aim for less semantically rich analytic provenance. By limiting the choice of data and

analysis/visualization, an inference algorithm has better chance to make the right guess. However, even within a specific domain (such as finance), the types of data and analyses involved are still of very large amount. Also, being limiting on the data and analysis can constrain the system capability, having a negative impact on the sensemaking task.

Given the difficulty of inferring task/sub-task information, a few methods target less semantic-rich provenance. One such example is *action chunking*, i.e., identify a group of actions that are likely to part of the same sub-task, without knowing what the sub-task is. Such approaches apply heuristics to infer patterns from action logs based on repeated occurrence and proximity in data/visualization space or analysis time [62]. Such chunking information can be useful in several ways. For example, the system can prompt user to take a note if such an action usually occurs within a specific sequence. Also, the grouping information can be used for aggregation when large amount of provenance information is to be visualized. This method is later extended to monitor user behavior for implicit signals of user intent and uses the information to suggests alternative visualization [62]. It is an open research problem to explore similar analytic provenance that can be effectively inferred and provides semantic information that can be used for supporting sensemaking.

2.3 Visualization

Computer-based visualization systems provide visual representations of datasets designed to help people carry out tasks more effectively [118]. Because the design space of possible visual "idioms" is huge, it is challenging to create effective visualizations. Understanding well-established information design principles and interaction techniques could guide designers toward the right direction. Also, every visualization needs to be evaluated to check whether it meets its design purposes and how it helps or hinder users.

2.3.1 Information Design Principles

2.3.1.1 Marks and Channels

Marks are basic geometric elements that depict items or links, and channels control their appearance [118]. Item marks can be zero-dimensional as a *point*, one-dimensional as a *line*, two-dimensional as an *area*, and three-dimensional as a *volume*, but rarely used. Link marks include *connection* showing a pairwise relationship

between two items using a line and *containment* showing hierarchical relationships using areas. Figure 2.7 illustrates these marks.

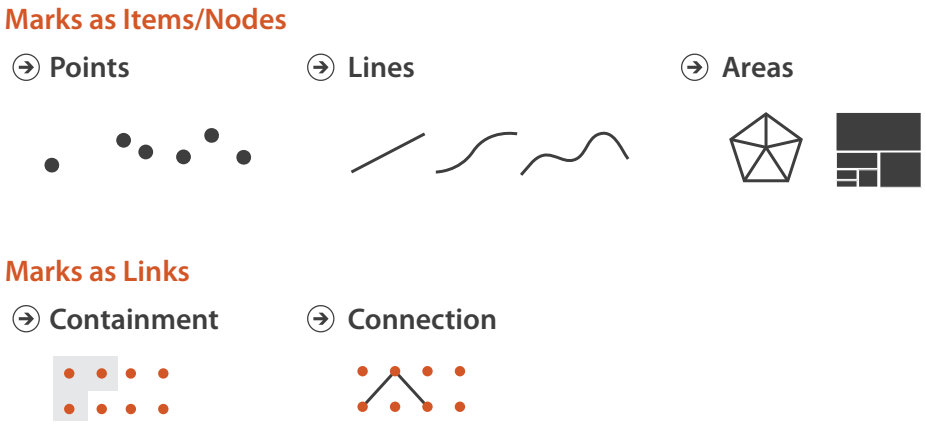
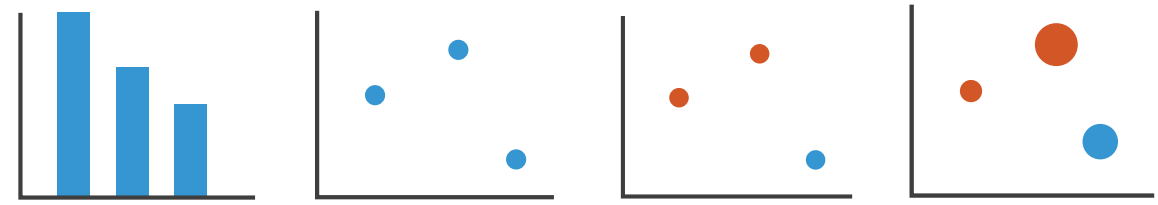


Figure 2.7: Item and link marks as geometric primitives. *Image source: [118].*

A visual channel control the appearance of marks, independent of their dimensionality. Examples include position, color, shape, angle and size. However, not all channels can be applied to all marks. For instance, an area mark is used in a geographic map to denote a region. It is already associated with a shape and size, thus cannot be size coded to represent another quantitative attribute. some visual channels. some cannot encode size, example. Figure 2.8 shows a progression of chart types, with each showing one more data attribute using one more visual channel. Figure 2.8a shows a bar chart representing a single quantitative attribute using the *vertical position* channel. Figure 2.8b shows a scatter plot encoding the second quantitative attribute using the *horizontal position* channel. Figure 2.8c adds the *color* channel to represent a categorical attribute, and Figure 2.8d adds the *size* channel to represent another quantitative attribute. In these examples, each attribute is encoded with a single channel. However, multiple channels can be combined to redundantly encode the same attribute, helping perceive it more easily.

All channels are not equal; they are processed and perceived differently by our human visual systems. Also, not all channels are appropriate for encoding both ordered and categorical attributes. Ordered attributes should be shown using magnitude channels, with *aligned spatial position* as the most effective channel and *3D volume* as the least effective one. Categorical attributes should be shown using identity channels, with *spatial region* as the most effective channel and *shape* as the least effective one. Figure 2.9 shows the detailed ranking of effectiveness of many visual channels, separated by the type of attribute. This ranking is documented by



(a) Line marks with horizontal position for a categorical attribute and vertical position for a quantitative attribute.

(b) Point marks with both horizontal and vertical position channels for quantitative attributes.

(c) A categorical attribute is added using the color channel.

(d) Another quantitative attribute is added using the size channel.

Figure 2.8: Using marks and channels.

Munzner [118], based on many empirical studies such as the work by Cleveland and McGill [30], and by Heer and Bostock [72].

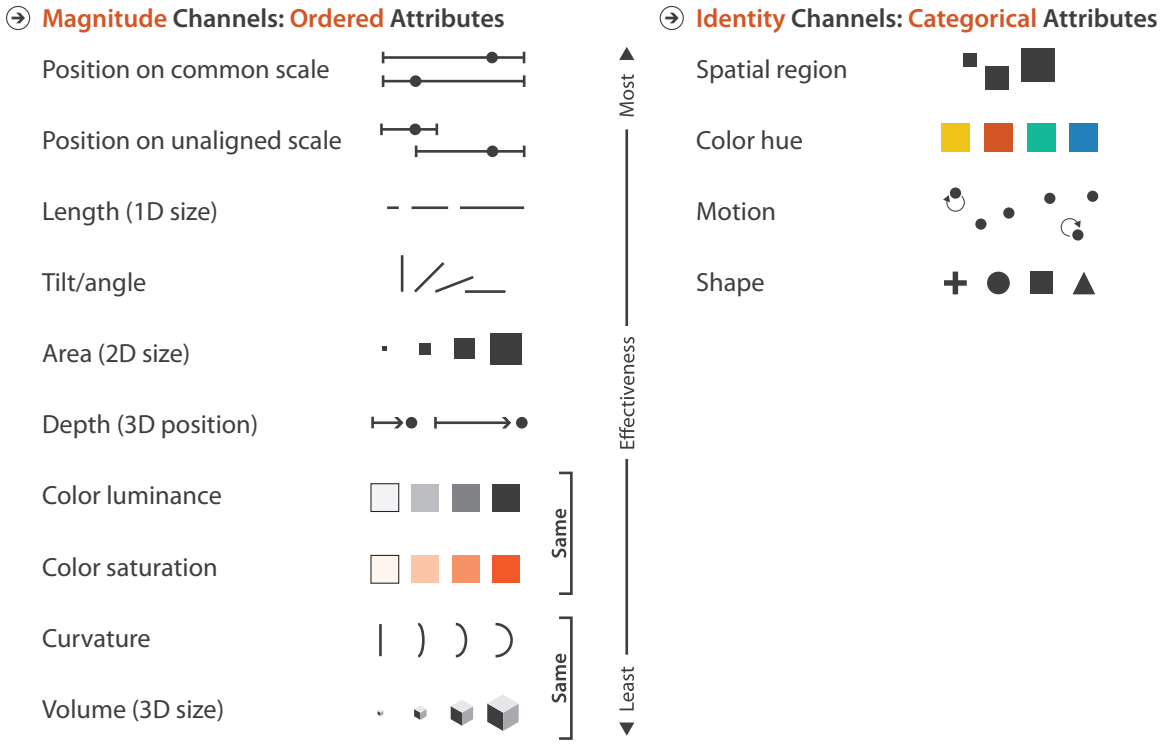


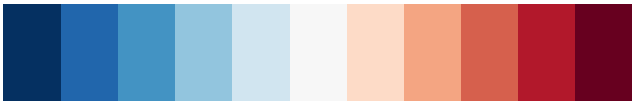
Figure 2.9: Channels ranked by effectiveness according to data and channel type. *Image source: [118].*

Color is a special channel that can be used for both data attributes. As shown in Figure 2.9, color luminance and saturation are used in magnitude channels, and color

hue is used in identity channels. A colormap specifies a mapping between colors and data values, and designing an effective colormap is challenging. ColorBrewer [68] is an excellent source for colormap reference, providing color schemes for both categorical and ordered attributes. Human can only distinguish around 12 colors simultaneously [118]. Figure 2.10a shows such a categorical colormap with 12 distinguished color hues. Ordered colormaps can be either sequential (Figure 2.10b) or diverging (Figure 2.10c). Diverging colormaps use two different color hues to emphasize values below and above the middle point.

(a) Categorical colormap with distinguishable color hues.

(b) Sequential colormap: a single color hue with different saturation level.



(c) Diverging colormap: two color hues emphasizing positive and negative values.

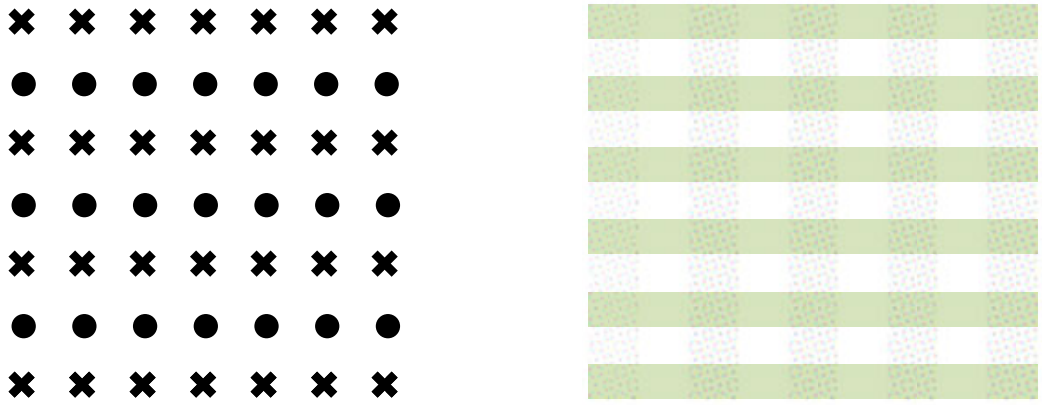
Figure 2.10: Colormaps from ColorBrewer. *Image source: [68].*

2.3.1.2 Gestalt Principles

Gestalt principles describe how we see patterns in visual displays [86]. This section reviews three commonly used principles in representing groups of items.

Similarity Similar elements tend to be grouped together. Figure 2.11a shows a matrix of point marks with uniform spacing, but using two different shapes: dot and cross. The similarity of shapes helps us see the rows more clearly than the columns. Two separable channels can be applied together to reveal patterns by either rows or columns. In Figure 2.11b, green is used to depict rows, and texture is used to depict columns.

Proximity Elements that are close together are perceptually grouped together. Figure 2.12a clearly shows two groups of dots. Figure 2.12b shows rows of dots. However, with a small change of spacing, these dots are perceived as columns

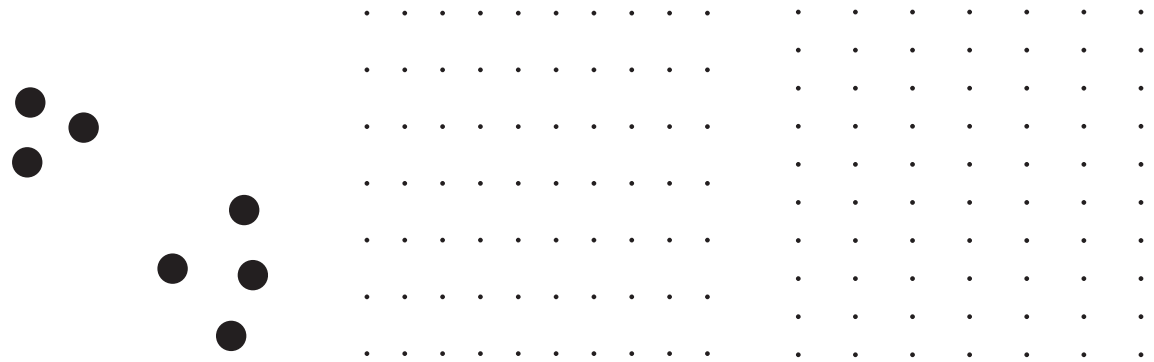


(a) Similarity of shapes distinguishes rows.

(b) Color and texture delineate rows and columns, respectively.

Figure 2.11: Similarity principle: similar elements are perceived as a group. *Image source: [181].*

in Figure 2.12c. The application of this principle is straightforward: organizing related information close together. It helps separate groups of unrelated objects and facilitates searching for information.



(a) Two groups of dots.

(b) Rows of dots.

(c) Columns of dots.

Figure 2.12: Spatial proximity principle: spatially close elements are perceived as a group. *Image source: [181].*

Connectedness Elements that are connected by visual properties are perceived as being more related than elements that are not connected. This principle can be achieved simply by drawing a border around a group of elements as in Figure 2.13a.

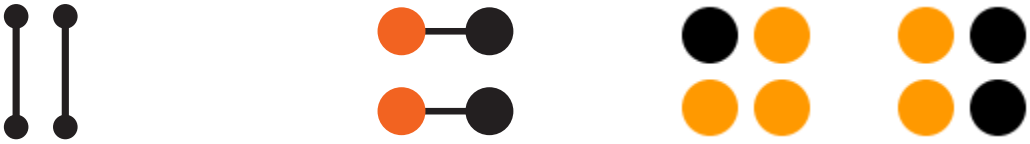
This is extensively applied in designing complex graphical user interface: groups of related features are separated by borders. Another approach to implement connectedness is by drawing lines between related elements as in Figure 2.13b. This is the basics of *node-link diagrams* – one of the most common methods of representing relationships between elements.

(a) Using border to denote a group.

(b) Using lines to denote a group.

Figure 2.13: Connectedness principle: visually connected elements are perceived as a group. *Image source: [181].*

Among these three Gestalt principles of representing groups of elements, connectedness has the strongest effect, followed by proximity and then similarity. Figure 2.14 illustrates this comparison. In Figure 2.14a, even though spacing between dots in rows is shorter than spacing between dots in columns, the lines make the vertical links clearer than rows. In Figure 2.14b, the lines also make the horizontal links more notable than groups of colored circles. In Figure 2.14c, two spatial groups are more clearly perceived than colored groups.



(a) Links are more clearly perceived than spatial groups.

(b) Links are more clearly perceived than colored groups.

(c) Spatial groups are more clearly perceived than colored groups.

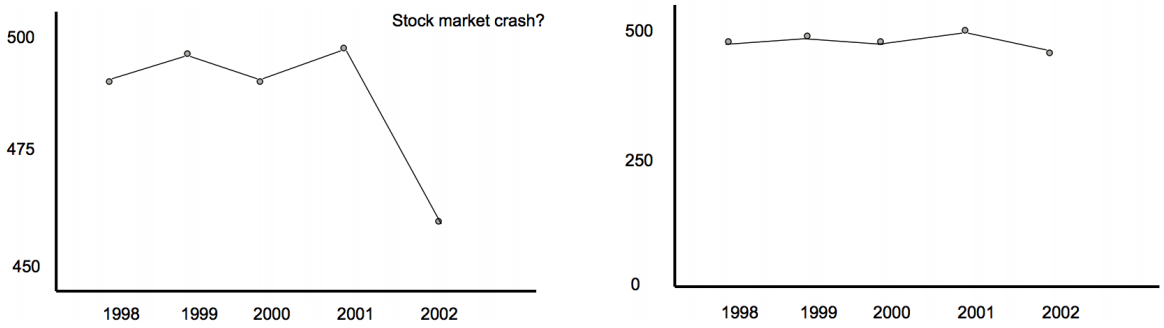
Figure 2.14: Comparison of Gestalt principles. Connectedness is stronger than proximity, and proximity is stronger than similarity. *Image source: [181].*

2.3.1.3 Tufte's Principles

Tufte proposes a number of principles for a well-designed graphic, documented in his series of books, most notably including *The Visual Display of Quantitative Infor-*

mation [172] and *Envisioning Information* [173]. This section reviews a few principles that have been commonly applied in graphic design and visualization.

Graphical Integrity This principle emphasizes that the graphical representation should tell the truth about the data. Representation of numbers, as physically measured on the surface of the graphic itself, must be directly proportional to the numerical quantities represented [172]. Figure 2.15a shows a falsely big drop in stock market value between 2001 and 2002. It because the chart uses a relative scale with the value range from 450 to 500, causing its height disproportional to the market value. Figure 2.15b corrects this error by using an absolute scale with the value range starting from 0.



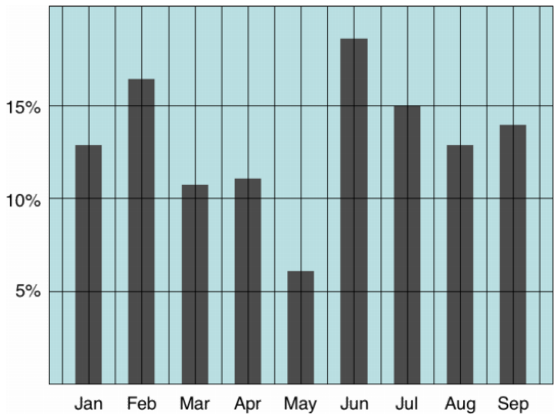
(a) Using a relative value range causes a falsely big drop of stock market value between 2001 and 2002.

(b) Using an absolute value range to depict the data accurately.

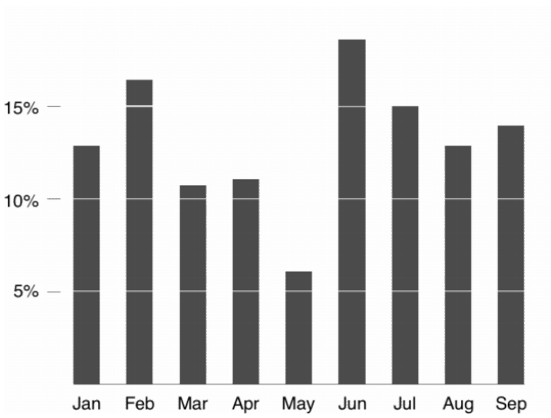
Figure 2.15: Graphical integrity principle. The chart should tell the truth about the data.

Data-Ink Ratio Maximization Data-ink includes the pixels in the graphic that are used for representing the data. Data-ink ratio is defined as the ratio between the data-ink and the total non-background pixels used in the graphic. This principle aims to maximize this ratio by erasing non-data-ink and erasing redundant data-ink.

Micro/Macro Readings This principle suggests that a graphic can contain both enormous details and an overall pattern. This allows the viewer to glance from a distance to observe the big picture, and later drill-down closely to examine its individual pieces. Classic stem-and-leaf plot is a great example to illustrate this principle (Figure 2.17). The plot shows all individual data items at meaningful level of detail, and provides an understanding of the data distribution. The micro/macro principle is extensively applied in interactive visualization, where zooming are



(a) A bar chart with poor data-ink ratio.



(b) A bar chart with high data-ink ratio by removing background, border and grid lines.

Figure 2.16: Data-ink ratio maximization principle: removing the graphic that does not contribute to the understanding of the data.

panning are made possible, such as Google Maps. Data items at different scales can be represented with different levels of detail to provide appropriate information based on the allowed display area.

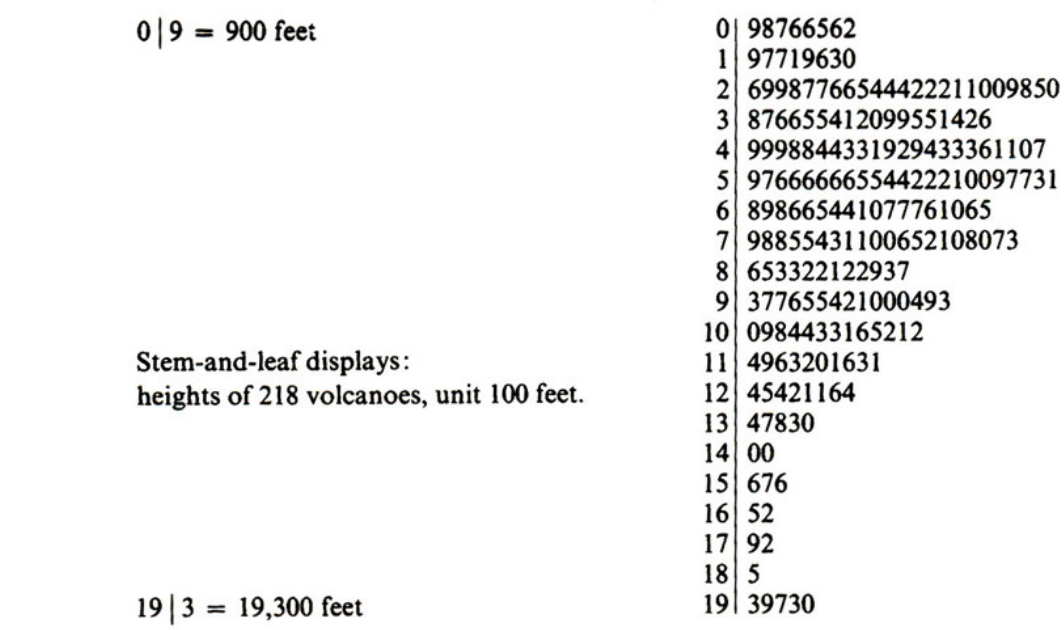
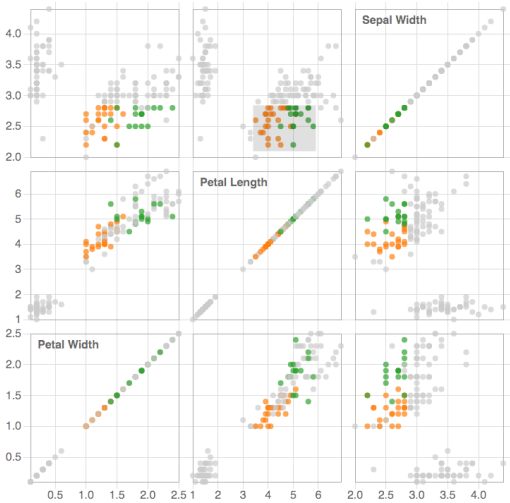


Figure 2.17: Micro/macro principle. A stem-and-leaf plot shows both the data distribution and individual items. *Image source: [172].*

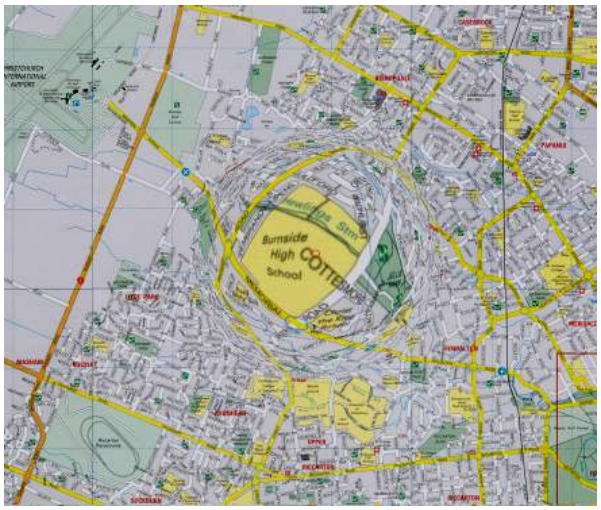
2.3.2 Interaction Techniques

Interaction typically refers to the set of controls provided to the user to manipulate an interface [133]. A static visualization may only show one aspect of a dataset. When the dataset is large enough, showing all the data at once may also make the visualization become cluttered. Interaction plays an important role in solving these problems. It can help explore large datasets at multiple levels of detail, identify patterns through examination of different visual representations and understand the connections between them.

Examples of interaction include standard techniques used in graphical user interface such as mouse clicking and scrolling, and more visualization specific techniques such as *linking and brushing* [97], and *focus+context* [32]. Multiple views in a visualization are often linked together to exploit their strengths. The user can select points of interest using the brushing technique, typically done directly on the visual data representation such as dragging a rectangular area. The points are brushed in one view will be highlighted in other views, allowing the user to explore them with different perspectives and representations (Figure 2.18a). Focus+Context is a technique that brings both the overview (context) and the detailed information (focus) together in one view. A fisheye view [53, 54] is one example of this technique: the focal region is magnified and displayed within its surrounding context (Figure 2.18b).



(a) Linking and brushing. Data points brushed in one view are linked and highlighted in other views.



(b) Fisheye view for focus+context. Both the overview and the detailed information are displayed in one view.

Figure 2.18: Examples of interaction techniques.

Many other interaction techniques can be found in the taxonomies by Dix and Ellis [43], by Keim [92], and by Wilkinson [186]. Very often, a user performs an interaction to achieve some goal, thus interaction techniques can also be classified based on their intents. Actually, different interaction techniques in different visualizations may serve the same purpose. For example, both drilling-down a treemap [156] and semantic zooming [131] aim to get more details. Taxonomies of high-level interaction can be found in the work by Yi et al. [194], by Heer and Shneiderman [75], and by Brehmer and Munzner [17]. These classifications could help visualization designers select suitable interaction techniques to serve for the capabilities they want to offer to the users.

Traditional graphical user interface widgets are often used to control different settings of a visualization, such as buttons and sliders. The disadvantage is that visual feedback does not appear where the interaction happens, but in some parts of the visualization. It also takes time for the user to search for the appropriate setting controllers. Direct manipulation [155] is an approach to address these problems. It enables the user to directly interact with the visual representation and receive immediate feedback. One example is using mouse scrolling to adjust zoom level while exploring a map instead of clicking on a button in the toolbar. Another example is parallel coordinates plot with axes can be reordered by direct dragging and values can be filtered by direct brushing on the axes (Figure 2.19). Surrogate objects can be used when the data objects are small or distant [29]. An example is the use of interactive legends as filtering means [79]

Another concept that can be applied to improve existing interaction techniques is *fluid interaction* [49]. Besides using direct manipulation as discussed previously, the interaction should produce smooth animated transitions between the state before and the state after an interaction, helping users maintain their mental maps; and provide immediate visual feedback, allowing users to know what is happening and/or what will happen next.

Typically, interaction techniques are combined to explore the data or present a known story. A classic visual information-seeking mantra by Shneiderman [157] summarizes many design guidelines and interaction techniques for designing information visualizations: *Overview first, zoom and filter, then details-on-demand*. With large datasets, it it challenging to create an overview and provides cues for further exploration. A more suitable approach in this case is *Search, show context, expand on demand* [176].

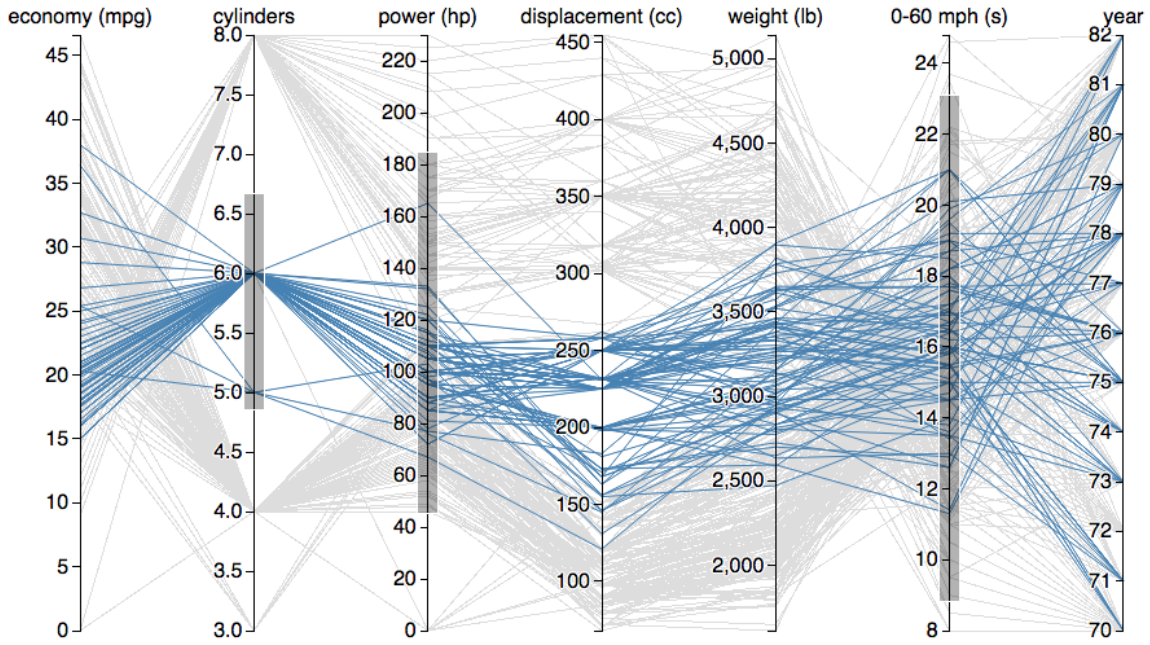


Figure 2.19: Direct manipulation in a parallel coordinates plot with reorder-able and brush-able axes.

2.3.3 Evaluation Methods

A visualization, no matter how novel and interesting it is, needs to be evaluated to check whether it meets the design goals and supports the target users to complete the intended tasks. Evaluation has been a research topic in visualization when the field becomes more matured [136]. Excellent reviews of visualization evaluation with different perspectives include evaluation techniques [26], scenarios [101] and design process [117].

In this section, we review the evaluation techniques based on the visualization design model by Munzner [117], helping address different concerns separately. The four levels include: explain the tasks and available data in the vocabulary of the problem domain, abstract them into domain-independent operations and data types, design visual encoding and interaction techniques to solve the abstract tasks, and develop algorithms to execute these techniques efficiently. Each level has its own *threats* to validity and methods to address them. Two types of methods are distinguished: *immediate* approaches can be done before inner levels are implemented, whereas *downstream* approaches requires all inner levels are completed. The threats and evaluation methods are summarized in Figure 2.20.

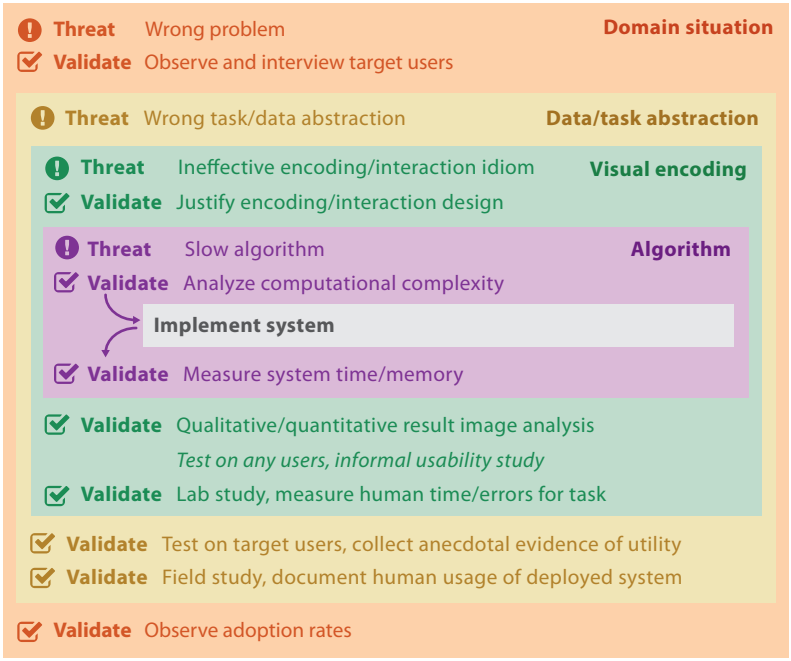


Figure 2.20: Threats and validation at each of the four nested levels of visualization design. *Image source: [118].*

2.3.3.1 Domain Problem and Data Characterization

The domain problem of target users is investigated to see if visualization is a potential solution. The primary threat is that the problem is mischaracterized: the users do not really suffer from the identified problem. An immediate form of validation is *field study* [26], where the investigator observes how target users act in real-world settings in order to learn and verify the characterization. Another technique is *contextual inquiry* [83], which allows the investigator to occasionally interview while the user is engaged in the process. One example is the study by Sedlmair et al. [154] on current working behavior and environments of analysis and diagnosis experts in the automotive industry.

One downstream form of validation is to report the rate the adoption rate of the tool by the target users. High effort is required to make the visualization solution reliable and deployable in the real-world environment. Examples include a field study of Google's Notebook product [149] and 6-week field trial of SparTag.us – a tagging system for foraging web content [84].

2.3.3.2 Operation and Data Type Abstraction

The threat at this level is the identified data and task abstraction do not solve the characterized problem. Only downstream approaches can be used to validate the abstraction. The deployed system needs to be used by target users completing their routine tasks in real-world environment. The goal of this evaluation is to collect anecdotal evidence that the solution is in fact useful. The observation and interview need to focus on understanding how the tool is used, and how it helps or hinders the users in performing their tasks. An example is a longitudinal field study of LiveRAC system that supports analysis of system management time-series data [111].

Evaluating the visualizations for supporting sensemaking can be done at this level, as the *evaluating visual data analysis and reasoning* scenario in the taxonomy by Lam et al. [101]. Due to the nature of sensemaking, evaluation is often carried out as case studies [89] with observation and interview, and followed by qualitative data analysis [102]. Attempts also have been made to quantify the insight or knowledge gained during sensemaking [187].

2.3.3.3 Visual Encoding and Interaction Design

At the design level, the threat is the chosen design is ineffective at communicating the desired abstraction to the user. One immediate form of validation is to justify every design decision based on known design principles such as the ones discussed in Section 2.3.1, or more comprehensive predefined guidelines as in heuristic evaluation [197]. Asking experts to review the design prototype also provides valuable feedback [170].

A common downstream approach is to conduct a controlled experiment comparing the design with other state-of-the-art alternatives [193]. A number of participants, depending on the expected size of the experiment, carry out a number of tasks representing real-world cases. Typically, task completion time and accuracy are measured and analyzed using hypothesis testing methods [50]. Post-task interviews are often combined to establish deeper understanding about how the visualization is used. If the experiment can be completed online, crowd-sourcing approach using Amazon's Mechanical Turk service can help largely increase the size of participants [72]. Another downstream approach is the measurement of common aesthetic metrics such as the number of edge crossings and edge bends that have been used in graph visualization [164].

2.3.3.4 Algorithm Design

The primary threat at this level is the algorithm is suboptimal in terms of time or memory performance. In interactive visualization, it is essential to ensure the interaction responsive in real-time. Analyzing the complexity of the algorithm using the standard approaches from the computer science literature [36] is an intermediate form of validation. The complexity can be computed based on the size of dataset or the display screen. Downstream approaches include measuring running time and memory usage for benchmark datasets.

2.4 Visualization of Provenance Data

We characterize the visualization of provenance data based on the level of semantics involved in the collected data [62], including event, action, sub-task and task. Because low-level events contain little semantics, they are often visualized and analyzed after users finished their tasks in order to gain deeper understanding about their processes rather than to provide near real-time sensemaking support to the users during their tasks. For example, visualization of users' mouse clicks can reveal patterns of application usage [110] and highlight some important usability issues, such as pages where users spent a lot of time and pages where they got lost during the task [182]. User interactions with visual analytics systems can be visually examined to recover their reasoning processes employed in their analysis tasks such as specific findings they found and strategies they used [45, 67]. Interaction logs have also been used to predict user performance of basic visualization tasks like visual search [18].

Next, we focus on the visualization of levels with richer semantics. Actions and states (the visualization results of the actions) are commonly used to show the analysis process. Sub-tasks and tasks are often documented by users in a graphical reasoning process.

2.4.1 Visualizing the Analysis Process

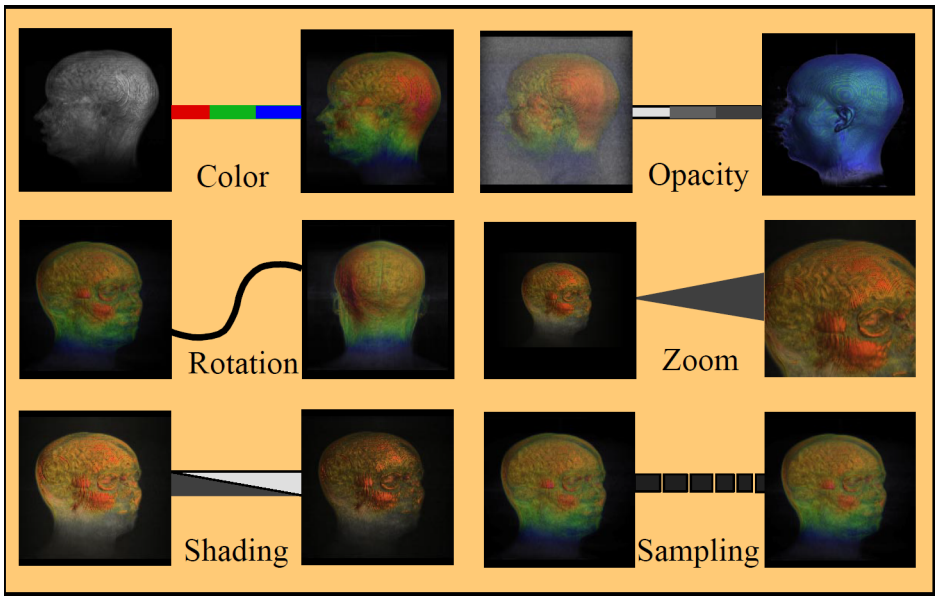
2.4.1.1 Visual Representation

Text Text is commonly used to describe actions or states, such as names of actions, titles of visited web pages and content of user notes. Text can provide accurate information, but long text makes it difficult for users to understand and recognize.

A graphical browser history by Ayers and Stasko [12] shortens web page titles to accommodate more pages in the view. Within a given width, its algorithm preserves complete words at both ends of the title and trims characters in the middle if necessary. Kaasten, Greenberg and Edwards [87] compare the recognizability of titles between various string sizes for all three truncation methods (text is truncated at the beginning, middle or end of the title). The results show that for medium (60%) recognition, we need 15–20 letters (depending on the truncation method) for web sites, and 28–39 letters for exact pages. For longer text such as user notes, machine learning techniques for text summarization could be beneficial [120].

Icon Sensemaking actions can be represented by graphical symbols, allowing users to easily distinguish them. They can be used alone to represent the analysis process when the visual result of each action is out of interest (Figure 2.21a). Alternatively, these icons can be used together with interface states, connecting the one before and the one after an action (Figure 2.21b).

(a) A sequence of icons displaying the analysis process. *Image source: [62].*



(b) Iconic actions connecting two visualization states. *Image source: [107].*

Figure 2.21: Using icons for representing actions.

Thumbnail Thumbnails are commonly used to represent visualization states, aiding users' recognition of previous ones. One study suggests that a thumbnail size of 96 pixels square could provide 60% accurate recognition of a visited web site [87]. For the same accuracy but in recognizing an exact web page, the thumbnail size rises to 144 pixels square. Additional information can be added to a web thumbnail to improve its recognition such as how often the represented page is revisited and whether that page is bookmarked or not [31]. For visualization thumbnails, visual encodings and parameters that were used to produce the visualization can also be embedded (Figure 2.22), providing more provenance information to the users.

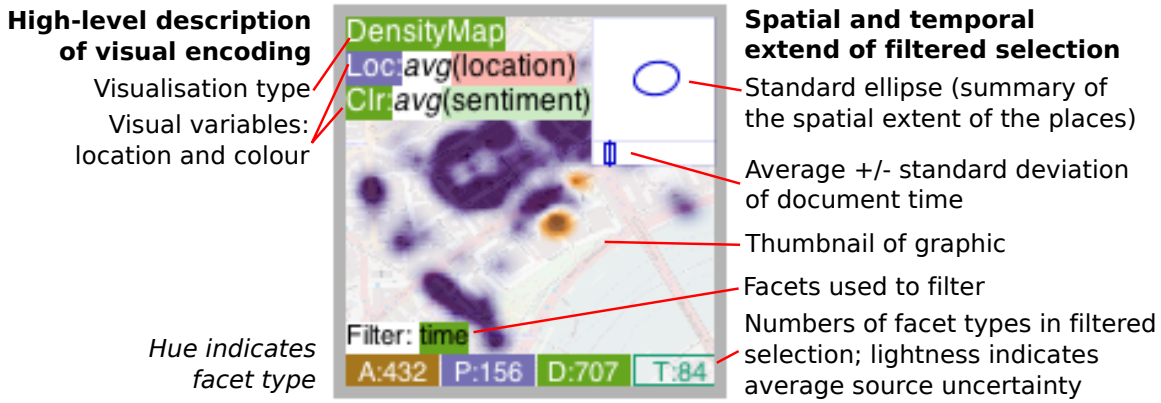


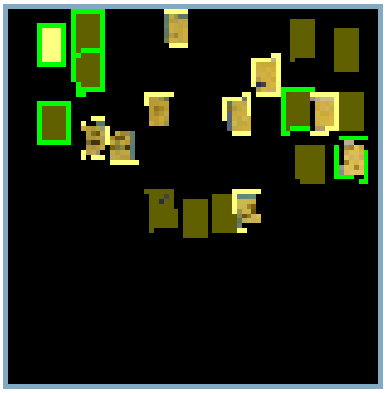
Figure 2.22: Visualization thumbnails with additional information about any filtering used, the characteristics of the filtered subset of data and the visual encoding. *Image source: [178].*

It may be necessary to apply pre- and post-processing adjustments to make the visualization thumbnails more recognizable [74]. For example, high-frequency visual elements that are not helpful in a small size such as gridlines and element borders can be removed to prevent their domination in the resulting thumbnail. As a result of down-sampling techniques, colors of data items may be different from them in the original visualization. Therefore, readjusting the color to match its intended value could help users to recognize the visualizations they analyzed in the past.

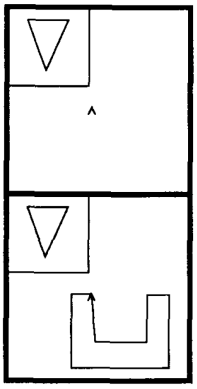
The default snapshot may also be an imperfect representation of a web page, especially if it contains a lot of text. Teevan et al. [167] propose an automatic method to produce a thumbnail improving its recognition. It consists of three components: some salient text at the top-left hand corner, a salient image below the text, and a watermarked logo superimposed at the bottom left hand corner of the image. The salient text contains about 20 first characters of the web page title. The salient image and the branding logo are chosen from the page using machine learning. Figure 2.23 shows such a thumbnail.

Figure 2.23: An enhanced web thumbnail (right) as a composite of some salient text, a salient image and a branding logo. *Image source: [167].*

Besides recognizing previous states, seeing the difference between a state and the one before it is also important in understanding the analysis process. One approach is to highlight the difference between two consecutive states (Figure 2.24a). The changes may only happen at one small portion of the entire interface. Therefore, showing only that area in both states could help users quickly identify the difference (Figure 2.24b).



(a) Modified items are highlighted with green borders. *Image source: [96].*



(b) Changed area is cropped and shown in both states. *Image source: [100].*

Figure 2.24: Techniques improving recognition of state changes.

2.4.1.2 Layout

Linear Layout Provenance data usually contains an inherent *time* attribute, recording when an action happened. An approach that emphasizes on the order of

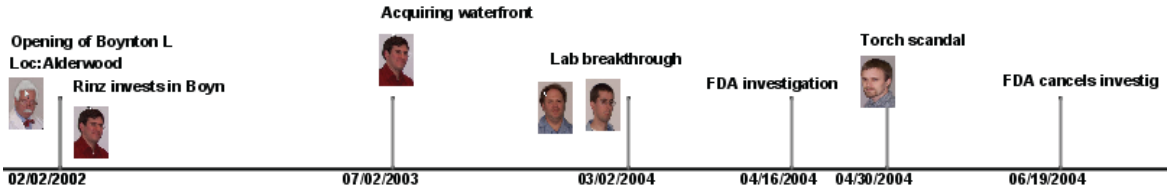
completed actions is to show them as a linear sequence of items like a comic strip [100, 112]. This layout facilitates visual scanning of past actions, allowing users to quickly understand the analysis process. Figure 2.25 shows such a layout.

Figure 2.25: Comic strip layout. A sequence of past actions in chronological order. *Image source: [74].*

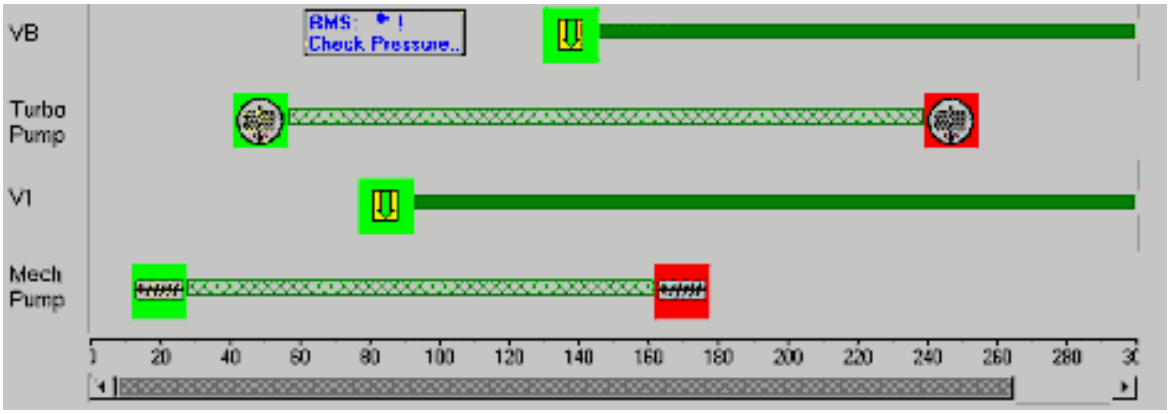
If the absolute timestamps of actions are of interest, a continuous timeline [40] is a more suitable layout. Actions are positioned along the time axis at when they happen (Figure 2.26a) or during which they last (Figure 2.26b). The time axis can be either horizontal or vertical [161]. The layout algorithms found in these provenance timelines are quite naive. POLESTAR [134] and the timeline from Jigsaw [105] require manual rearrangement of the events from users to solve the overlapping problem. The timeline from nSpace2 Sandbox [161] shows events at the exact time when they happen without considering possible intersection between of them.

Another approach is to use both the horizontal and vertical axes to represent time. BrowseLine [82] uses the vertical axis for *macro-time* and the horizontal axis for *micro-time*, similar to stem-and-leaf plots. More specifically, a two-dimensional timeline (Figure 2.27) is divided into rows, each for a big time-slot, such as one hour. In each row, events happening within that hour are positioned along the horizontal axis without considering their absolute timestamps. This design assumes that users may only remember roughly when events happen, thus absolute positioning in the vertical axis facilitate them in recognizing past events. Moreover, relative positioning in the horizontal axis could help pack more events and prevent overlapping.

Branching Layout Many sensemaking systems allow users to revisit their past states such as through undo/redo features or backward/forward in web browsers. From a past state, if the user performs a new action, it should be recorded in a new branch forking from that state. This branching history is typically visualized with a tree layout to represent the logic of the analysis process effectively. In such a tree, nodes represent a summary of system states, and edges represent actions that transition the system from one state to another. Examples can be found in provenance-enabled systems in different fields including scientific visu-



(a) Time-point provenance data. User annotations are positioned along a time axis at when their associated events happen. *Image source: [61].*



(b) Interval provenance data. User actions are shown as horizontal bars a long a time axis covering their durations. *Image source: [139].*

Figure 2.26: Timeline layout for visualizing the analysis process.

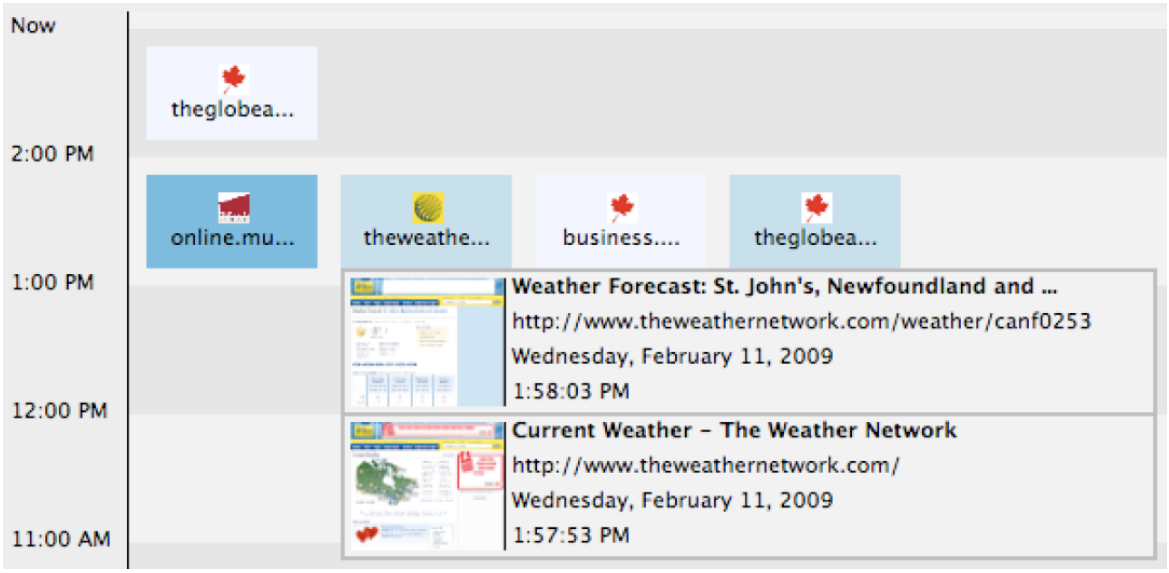


Figure 2.27: 2D timeline. The vertical axis represents macro-time, whereas the horizontal axis represents micro-time. Events within a macro time-slot are positioned in chronological order as a comic strip. *Image source: [82].*

alization [107], information visualization [46], visual analytics [88] and browser history [12]. Figure 2.28 shows such a provenance tree.

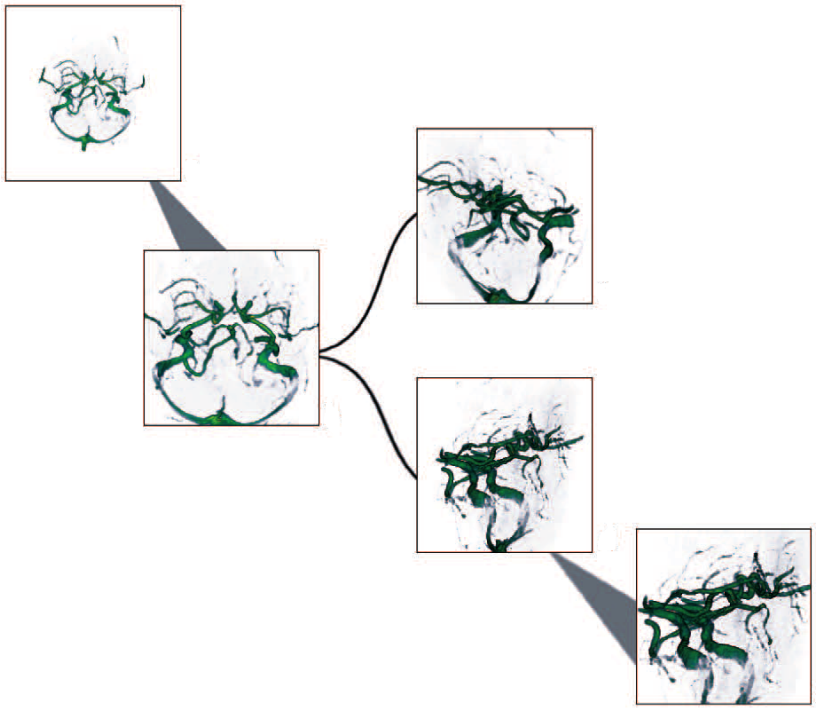
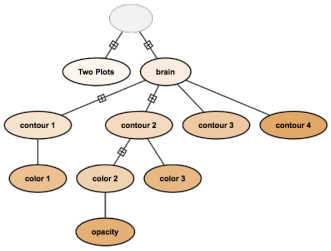
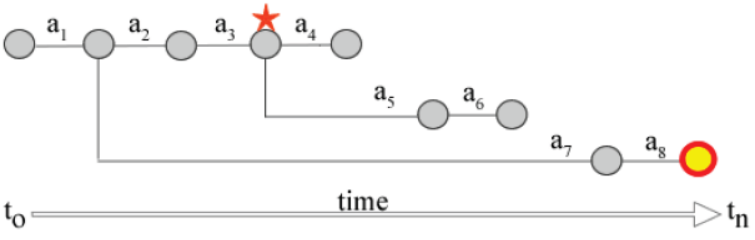


Figure 2.28: Tree visualization for branching analysis process. Nodes are thumbnails of past visualization states and links are transforming actions. *Image source: [85].*

In a tree layout, the order of actions can be inferred through the direction of edges. Moreover, exact time gap between actions can also be visually encoded into the visualization. VisTrails [13] color-codes the background of nodes according to their creation time (Figure 2.29a). Aruvi [158] uses the length of edges to represent the relative time gap between two states (Figure 2.29b).



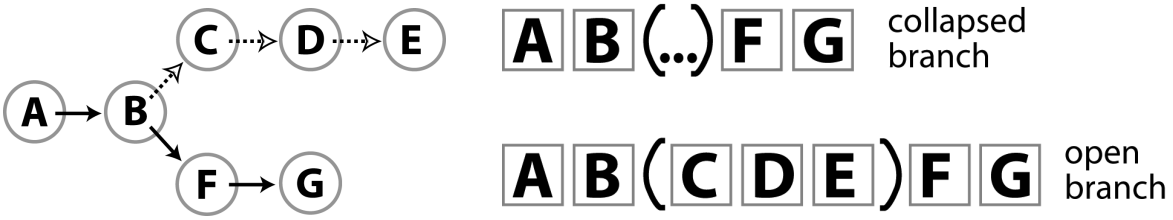
(a) Node backgrounds are color coded based on time. *Image source: [174].*



(b) The edge length between two nodes represents the interval between them. *Image source: [158].*

Figure 2.29: Encoding temporal information into provenance trees.

The diagonal arrangement of tree as in Figure 2.52 may consume a lot of space. One approach is to use only horizontal and vertical edges as in Figure 2.29b. Another approach is to display only the path that led to the currently active visualization [96]. Other paths can be expanded on demand. Figure 2.30b shows examples of representations of the active path and the full history.



(a) Branched history. The user first performed actions A, B, C, D and E, then undone to D, C and B, then performed new actions F and G.

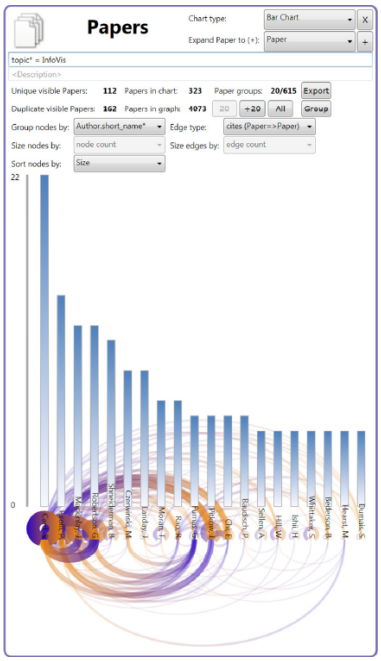
(b) Inline branching history representation. Top: displaying the current path. Bottom: displaying the full history; actions not part of the current path are placed between brackets.

Figure 2.30: Branching layout for tree visualization of the analysis process. *Image source: [96].*

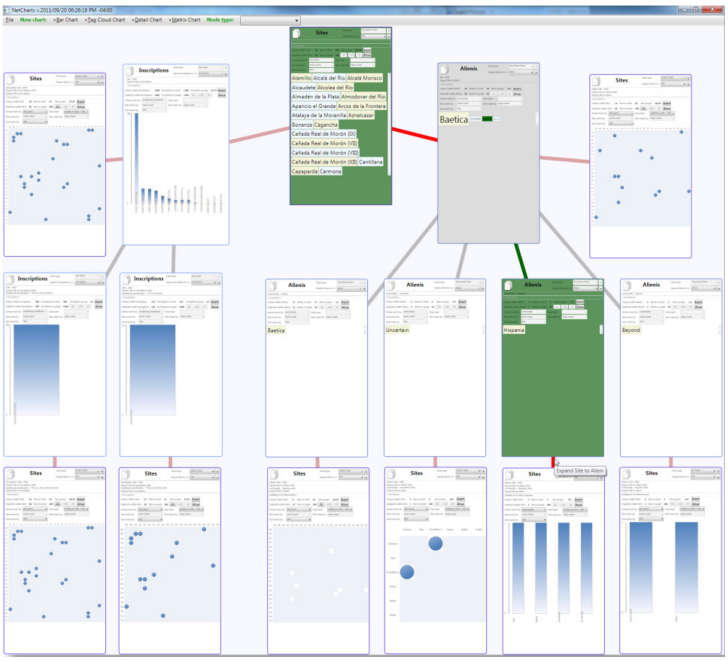
Interaction is also commonly used to address the scalability issue. Zooming and panning allow users to see the overview of the analysis process and navigate to the part of interest [46]. Collapsing and expanding branches of the tree on demand can also help reduce its size and avoid distraction [13]. Distortion techniques may help users to quickly navigate and focus on more relevant states and actions [112]. Another technique for compressing the provenance tree and improving user understanding is *action chunking* [74]: a sequence of related actions may be better represented as a single higher-level action. For example, a quick succession of sort or filter actions likely indicate a multi-step configuration of the view and can be chunked together.

Typically, provenance data is shown in a separate view, linked with other data views of the visualization system [74, 88, 132, 158]. However the system can also be used as a single item of the provenance view. For instance, in GraphTrail [46] – a single-view visualization system, when the visualization is changed (e.g., changing attribute mapping in a bar chart), it creates a new view containing that visualization and links with the current view. This is similar to how normally the provenance view is developed; however, GraphTrail makes the entire exploration space become the provenance view. Moreover, each history item is not a thumbnail, but a fully interactive visualization. By allowing zooming and panning, users can choose between a close examination of individual system states (Figure 2.31a) and an overview of the analysis structure (Figure 2.31b). An extra overview window as in

overview-and-detail technique [32] could be useful in search and navigation in a large space.



(a) Zooming in to work on the active visualization.



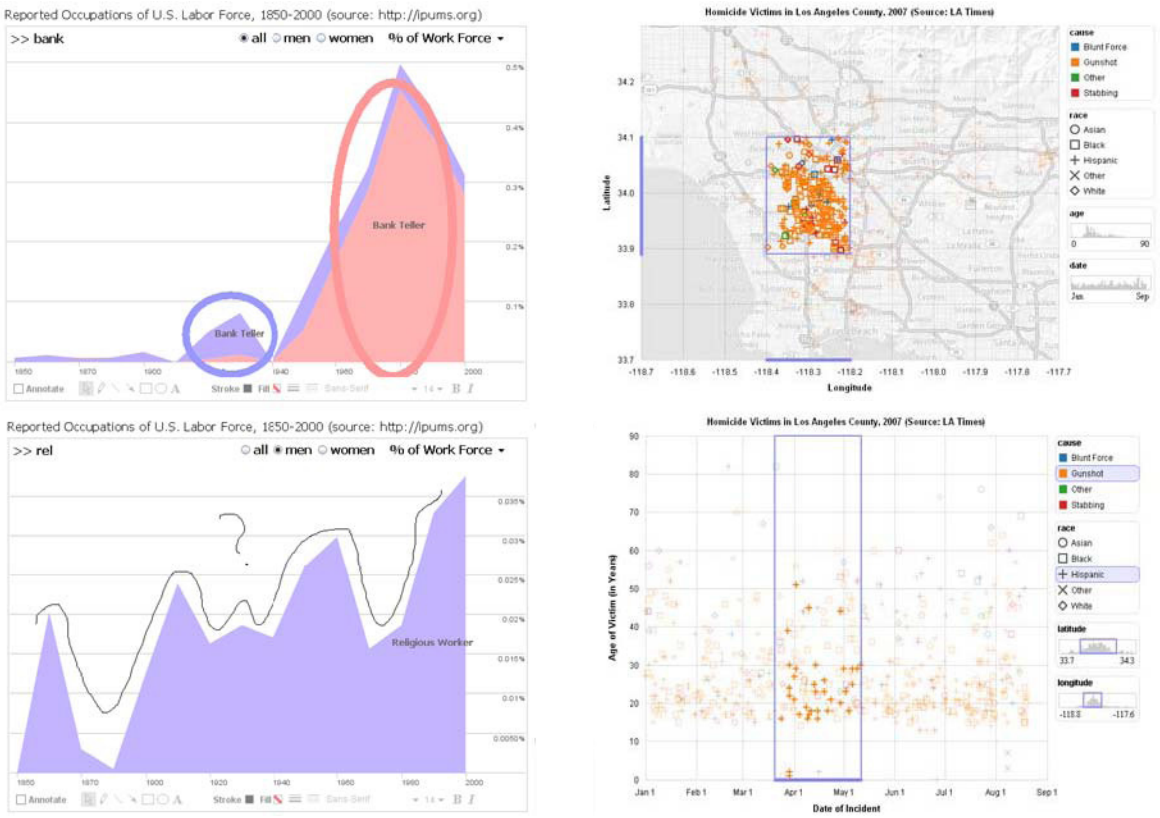
(b) Zooming out to see an overview as the history of the exploration process.

Figure 2.31: Unification of the exploration view and the history view. *Image source: [46].*

2.4.2 Visualizing the Reasoning Process

The reasoning process reflects user findings, methods and strategies in performing a task. This process mainly happens in the user's mind, and some sensemaking systems allow the user to manually externalize it through user bookmark [178], annotation [76], and manipulation of those externalized items [192]. A common form of reasoning externalization is to allow the user to write down their thinking. It could be a free note [158] or a description of a user bookmarked visualization [178]. Alternatively, users can annotate directly on the visual bookmark using standard graphical editing features such as circling on interesting patterns (Figure 2.32a – Top) or hand drawing on a suspicious trend (Figure 2.32a – Bottom). To make the annotation meaningful across multiple views, the selection should be aware of the data involved in it [71]. For example, in Figure 2.32b, the annotation in the top view also makes the data items in the bottom view get annotated using the same selection

query. Data-aware annotations allow users to see the data of interest remained across different views.

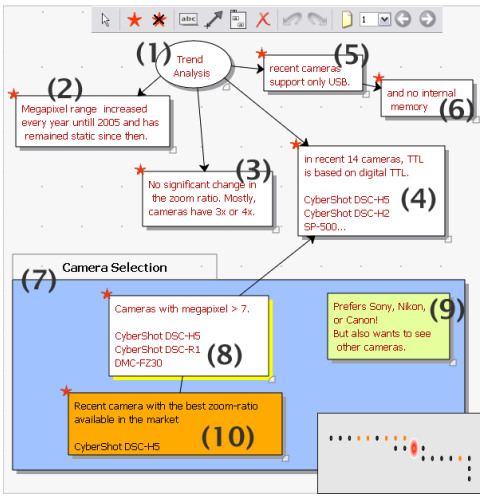


(a) Geometric annotation. **(b)** Data-aware annotation.

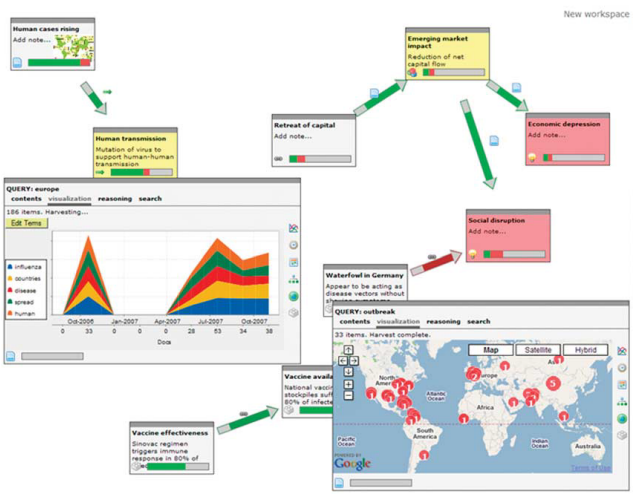
Figure 2.32: User annotation of bookmarked visualization.*Image source: [75].*

A note can simply be any user thought, but it can also take on different roles such as *evidence*, *assumption* and *hypothesis* [132]. They have different visual representations, enabling users create and manage their complex reasoning processes. Typically, users are allowed to move notes freely and draw edges to connect them, enabling spatial grouping of related notes and relationships establishment. These interactive features are known to help users produce more critical thinking in their analyses [153]. For example, users can show connections between a hypothesis and its evidence by drawing links, and spatially separate the evidence into a supporting group and a counter group, facilitating further comparison. Figure 2.33a shows such a diagram produced with user notes and Figure 2.33b shows such a more formal diagram with different reasoning roles.

More analytical reasoning methods have also been applied on the provenance data. The SRS system [132] first allows the user to construct a reasoning diagram



(a) A diagram of user notes showing their thoughts. *Image source: [158].*

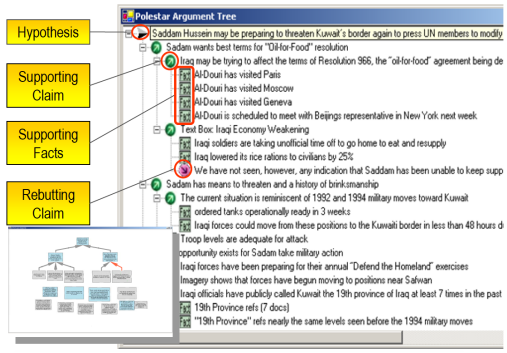


(b) A formal reasoning diagram with nodes having different roles: evidence, casual relationship, assumption and hypothesis. *Image source: [133].*

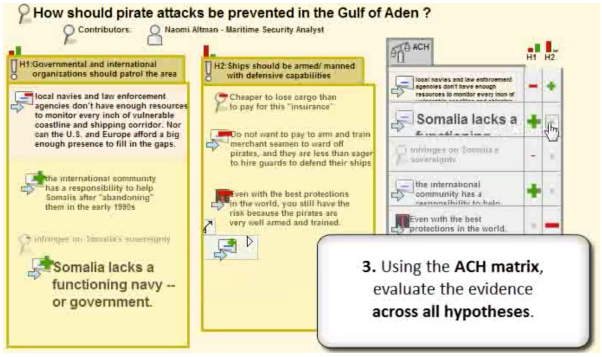
Figure 2.33: Visualization of reasoning process.

with bookmarked visualizations as evidence nodes and causal relationships as links (Figure 2.33b). These evidence nodes and links may lead to a hypothesis. The user then specifies their confidence about the evidence before the system computes the likelihood of the hypothesis using the Dempster–Shafer belief network [152]. The result provides analytical support to the user.

POLESTAR [134] supports argument structuring, based on methods for analyzing legal documents such as the Toulmin model of argument [171]. To help validate a hypothesis, it organizes arguments as a tree structure of claims, each supported by at least one piece of evidence. A claim can have its supporting sub-claims and can also have rebuttals that weaken or restrict their force. Figure 2.34a shows such a argument tree. Sandbox [190] supports analysis of competing hypotheses – a judgment method that requires careful weighing of alternative explanations [80]. It allows users to add multiple hypotheses, each with supporting or contradictory evidence. These pieces evidence are weighted by the users and summarized in a visual matrix, enabling the users to effectively make a decision without having to remember and analyze in their minds. Figure 2.34b shows an example of this support.



(a) Argument tree. Analysts elaborate arguments via a tree structure of claims, sub-claims, and facts. *Image source: [134].*



(b) Analysis of competing hypotheses. The interface allows adding multiple hypotheses, each with weighted supporting/counter evidence, and producing a summary matrix to evaluate them. *Image source: a video snapshot of [190].*

Figure 2.34: Analytical support for the reasoning process.

2.5 Visualization of Time-Oriented Data

Time is an essential aspect of life; everything contains inherent temporal attributes such as the time when a person was born and the time when an event happens. Long before computers were invented, information graphics have been used to represent temporal relationship of data. One of the oldest documented timelines was created back in 1765 entitled *Chart of Biography* by Joseph Priestley (Figure 2.35). It shows the lifespans of two thousand famous names along a horizontal time axis, spanning from 1200 BC to 1800 AD. He uses a horizontal line segment to depict a lifespan, and adds dots to either ends to indicate the uncertainty of the reported values.

Since then, many visualization techniques have been developed to effectively reveal the temporal relationship of data. The book by Aigner et al. [6] provides a comprehensive review of this topic. In this section, we focus on different visual mappings of time.

2.5.1 Horizontal

The most common representation of time is mapping it to a horizontal axis as in the aforementioned *Chart of Biography*. Data items are positioned along the axis as either a point mark for point-based time or a line mark for interval-based time. Given a two-dimensional space, the vertical axis is available for encoding additional information.

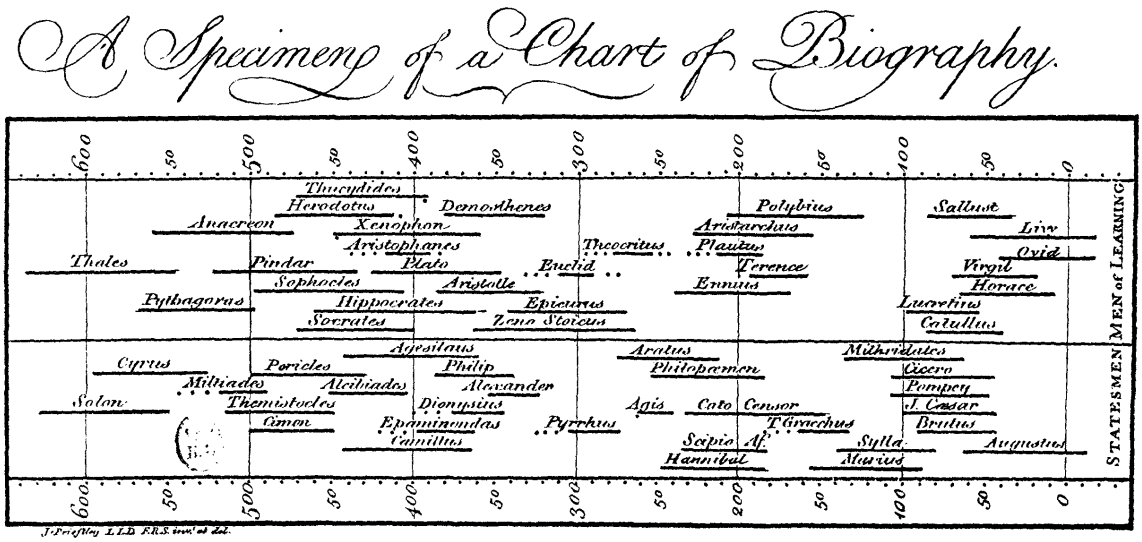
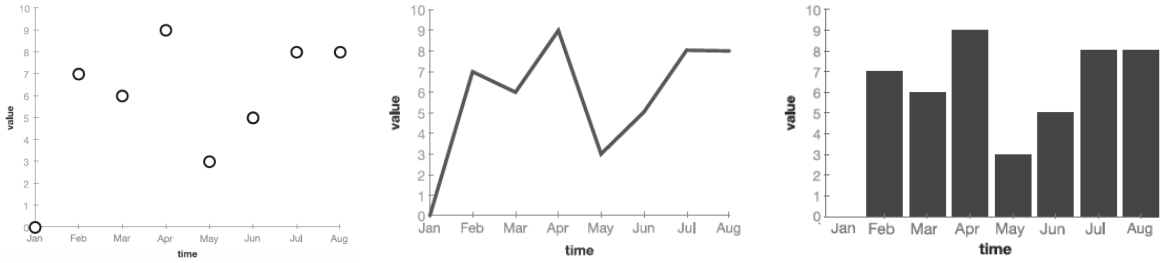


Figure 2.35: Joseph Priestley's Chart of Biography portraying the lifespans of famous historical persons. *Image source: [140].*

2.5.1.1 Scaled Vertical Axis

Time-series data is a sequence of data points collected at uniform intervals such as population of a country for every year and stock market value for every hour. A time-dependent variable in time-series data is often mapped to a vertical axis. Classic charts such as scatter plot, line chart and bar chart and are all commonly used for this purpose. Scatter plot shows each data point as a point mark, with the x-coordinate mapping to the temporal value and the y-coordinate mapping to the value of the time-dependent variable. Line chart further connects these data points to form a line. Bar chart also shows data points individually like scatter plot, but each data point is represented by a bar, with its height corresponding to its time-dependent value. Line chart is suitable for showing trends of the series, whereas scatter plot and bar chart are good at emphasizing individual data points. With the advantage of visual alignment, bar chart is more effective than scatter plot at comparison of time-dependent values [6]. Figure 2.36 shows an example for each of these three charts.

Horizon graph [142] is a recent improvement of line chart for visualizing time-series data, designed for a more space-efficient representation in order to facilitate comparison of different series, such as daily prices for one year of multiple stocks. To make a fair comparison, it shows a derived percentage changes from the earliest data point instead of raw values. Starting from a line chart, the value range is divided into



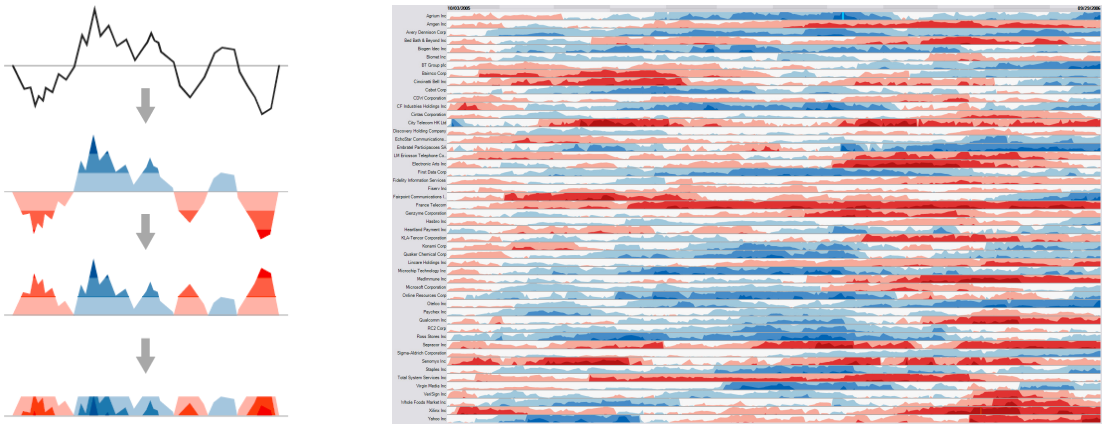
(a) Scatter plot: showing data points as dots.

(b) Line chart: connecting data points with lines.

(c) Bar chart: showing data points as aligned bars.

Figure 2.36: Three charts showing the same time-series dataset with the horizontal axis representing time and the vertical axis representing a numerical value.

equal bands, such as 10% for each band, and color coded using a diverging colormap for positive/negative values with increasing color intensity for greater band values. The colored bands allow more precise value reading. Then, the negative values are mirrored into the positive side, reduced the chart height by half. To save more space, those bands are layered with increasing values atop using the two-tone pseudo coloring technique [151]. Figure 2.37a illustrates these steps, and Figure 2.37b shows prices of 50 stocks over a year.



(a) The construction steps: color → mirror → layer.

(b) A horizon graph showing prices of 50 stocks over a year, each row for a stock.

Figure 2.37: Horizon Graph: a space-efficient visualization of time-series data. *Image source: [142].*

Horizon graph uses space more efficiently than line chart in visualizing time-series data. A study by Heer, Kong and Agrawala [73] investigates their performances in value comparison tasks. The results show that mirroring a chart (flipping negative values) does not have negative effects; it neither slowed completion time

nor hurt accuracy. Moreover, for charts with small sizes, layered bands are even more effective than line chart.

Stacking multiple area charts on top of each other is a suitable approach to visualize multiple time-dependent variables. ThemeRiver [69] can be considered as a smooth and symmetric version of stacked graph, designed to show thematic variations over time within a large collection of documents. Each theme is displayed as a colored current flowing through the time, and at any point, the width of the current maps to the strength of the associated theme. The overall river consists of multiple colored currents, providing a good overview of the themes that were important at certain points in time.

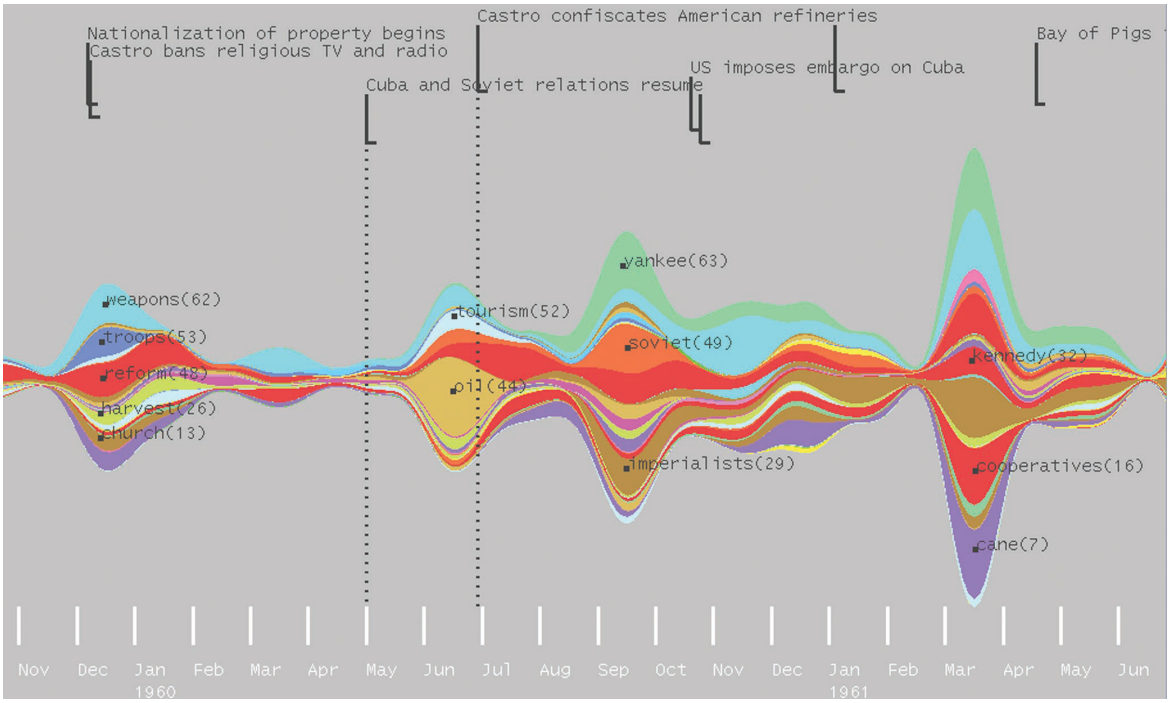
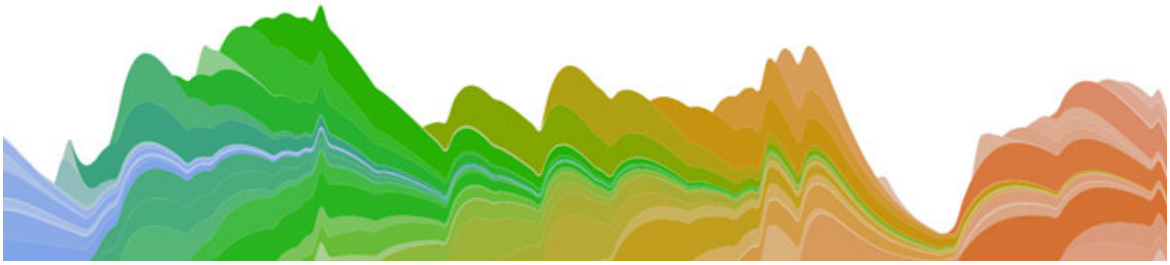
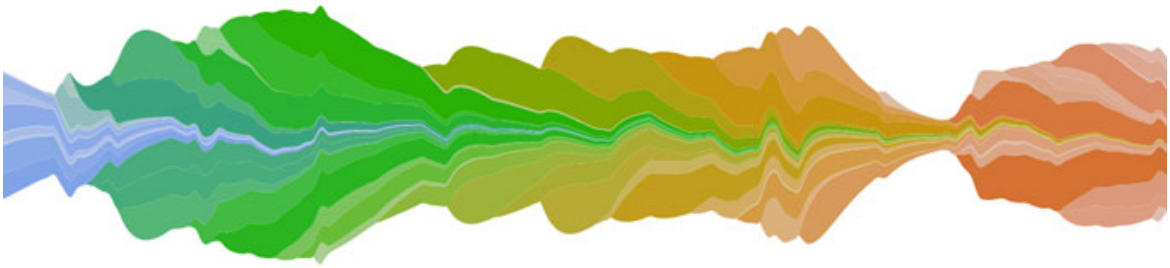


Figure 2.38: ThemeRiver uses a river metaphor to represent themes in a collection of Fidel Castro's speeches, interviews and articles from the end of 1959 to mid-1961. Each colored current corresponds a theme. *Image source: [69].*

Byron and Wattenberg discuss considerations of aesthetics and legibility for designing such stacked graphs. Figure 2.39a shows the traditional stacked area chart, where the bottom of the lowest layer is a horizontal line at 0. Figure 2.39b shows the ThemeRiver version, which is optimized for the symmetry of the entire layout. Figure 2.39c shows the Stream Graph [24] version, which minimizes the number of wiggles of layers.



(a) Traditional stacked graph.



(b) ThemeRiver.

(c) Stream Graph.

Figure 2.39: Stacked graphs with different design considerations. *Image source: [6].*

2.5.1.2 Non-Scaled Vertical Axis

The vertical dimension may be used to avoid clutter in producing a compact layout. LifeLines [137, 138], a visualization system of patient records, is a good example. It groups these records into different facets such as problems, allergies, diagnosis and medications, and vertically stack them on top of each other. Within each facet, interval-based records are represented as horizontal bars covering their timespans. Because their timespans may overlap, the layout adjusts their vertical coordinates to avoid intersection. Figure 2.40 shows an example of LifeLines.

Another example is Continuum [10], a timeline visualization of hierarchically structured temporal data such as the relationship of era, composers and pieces.

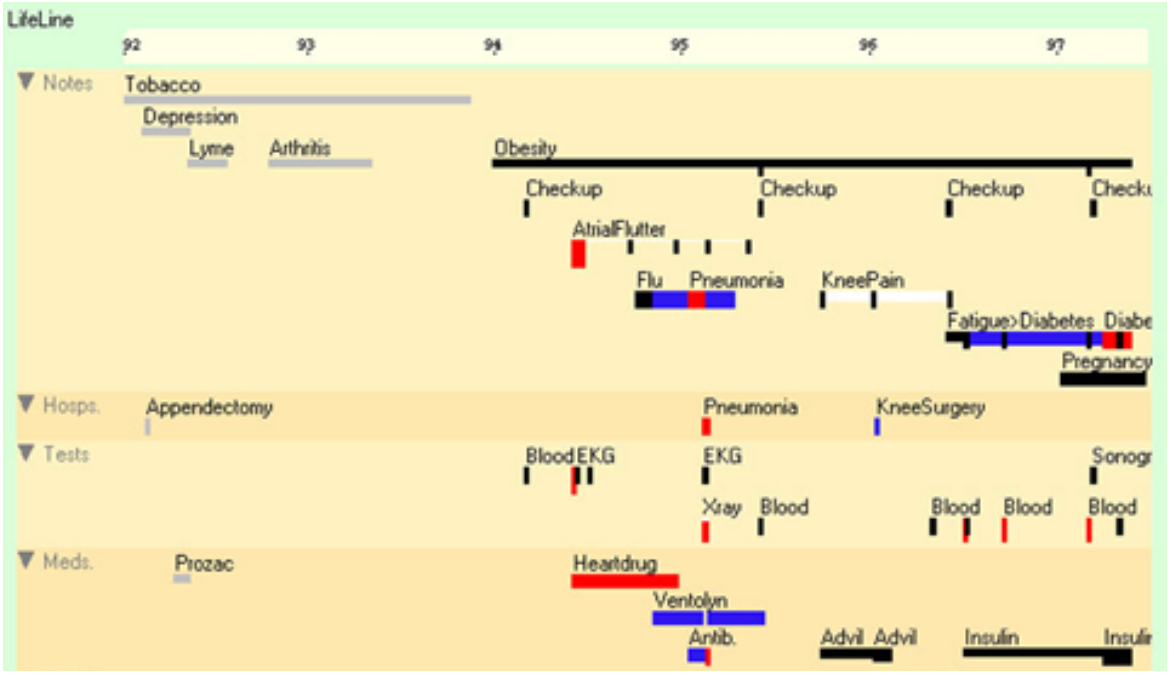


Figure 2.40: LifeLines. The timeline consists of several facets, stacked vertically. Each facet includes health-related records shown as horizontal bars, which may be located in different rows to avoid overlapping. *Image source: [138].*

The timeline shows the lifespans of composers as rectangles, where the width represents temporal information, and the height depends on the number of pieces they composed. It also uses vertical position to produce an overlap-free layout.

Temporal visualization often encode additional relationships of data items. Gantt chart [55] is a classic method for displaying planning activities in project management. Each activity is shown as a horizontal bar covering the planning time, and text from the left part of the chart shows activity names. Dependency is a common relationship in planning and can be shown as an arrow. For instance, an arrow pointing from the end of activity A to the beginning of activity B can indicate that B must happen after A. Activities are often organized in hierarchical structure such as tasks and sub-tasks. They can be ordered and arranged with indentation to reflect this structure. Timeline tree [21] draws an explicit node-link tree on the left part of the timeline to show the hierarchy. Figure 2.41 shows a Gantt chart with hierarchically structured tasks.

Spatial proximity is another method to represent relationships. This method is applied in storyline visualizations to illustrate the dynamic relationships between characters in a movie, which was first introduced by Munroe with his hand-drawn charts [117]. The visualization depicts each character as a curved line and each scene

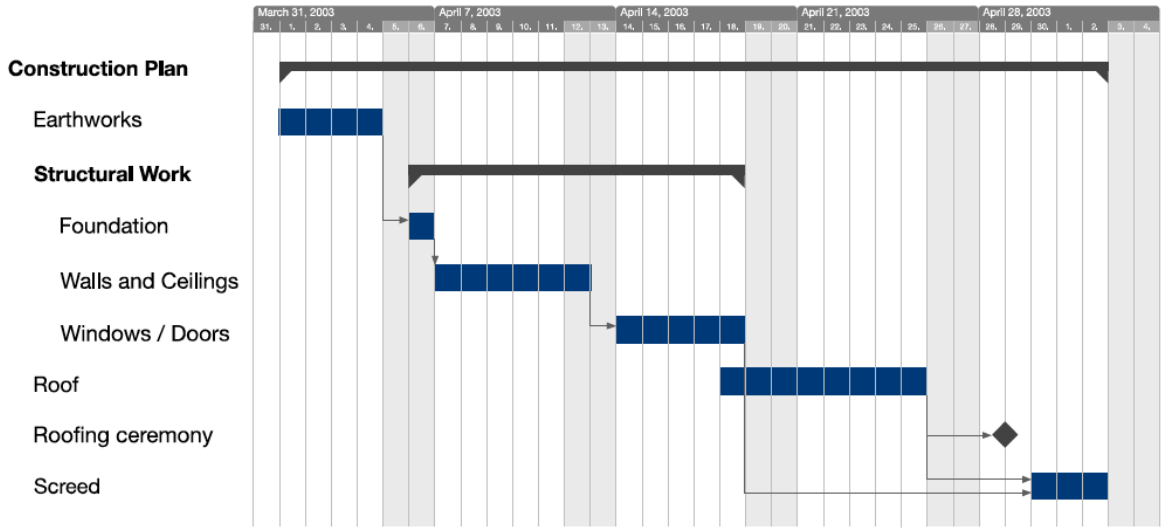


Figure 2.41: Gantt chart. Each task is shown in an individual row with task name on the left and horizontal bar on the right spanning task time. Arrows show dependency, and label indentation indicates the hierarchy of tasks. *Image source: [6].*

as a bundle of those character lines. Ideally, all the lines within a scene should be horizontally parallel. A line diverges from its bundle if the character leaves the scene, and conversely, a line converges into a bundle if the character joins that scene. Algorithms have been introduced to automate the drawing process, including work by Tanahashi and Ma [165] and Liu et al. [104]. Figure 2.42 shows the storyline visualization of the *Star Wars* movie.

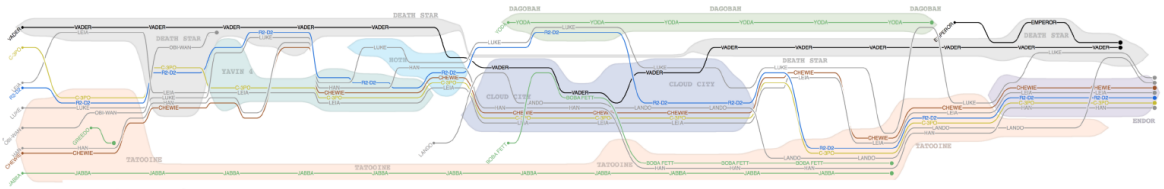


Figure 2.42: Storyline visualization of the movie *Star Wars*. Each line represents a character, and a bundle of lines represents an interaction of those characters. *Image source: [165].*

Similarly, TimeNets [93] also applies *proximity* to visualize relationships of genealogical data. It uses a curved line to represent the lifespan of a person. These lines are located spatially distant if the persons they represent are unrelated. Two lines converge if the represented persons marry and diverge when they divorce. Child lines stay close to their parent lines, and dotted vertical lines are drawn from the parents to the beginning of their child lines to indicate the parent-child relationship. Figure 2.43 shows a TimeNets example depicting these relationships.

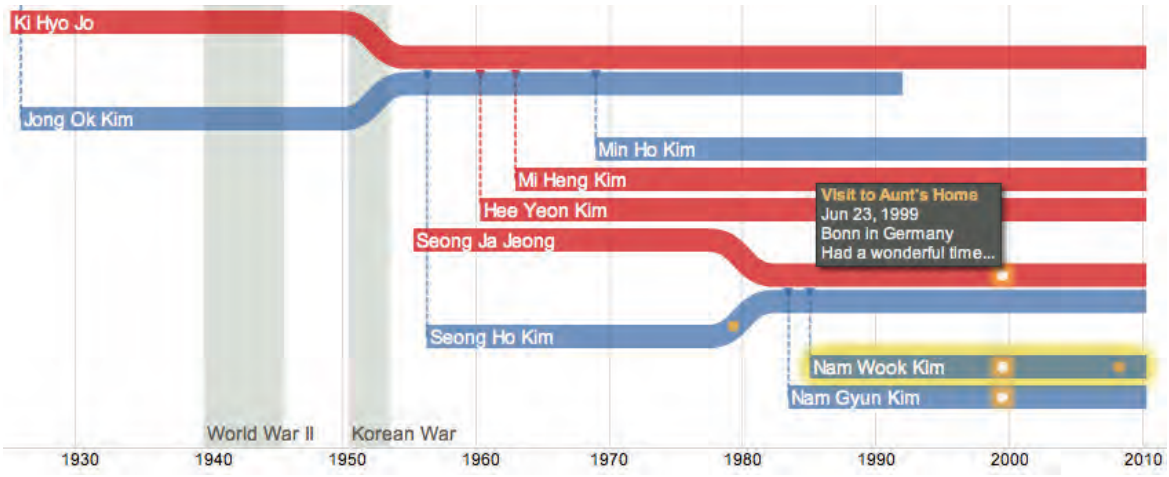


Figure 2.43: TimeNet visualization of genealogical data. Lifelines represent people, converging lines signify marriage, and drop lines indicate children. *Image source: [93].*

2.5.2 Spiral

Another representation of time is mapping it to a *spiral* axis, and data items are positioned along that spiral [183]. Color intensity and line thickness are suitable for encoding an additional quantitative variable. For interval-based data, filled curved segments are aligned with the spiral to indicate the two ends of intervals [25]. Spirals can also be intertwined to show multiple variables. Figure 2.44 shows a spiral graph comparing two variables over time with different color hues.

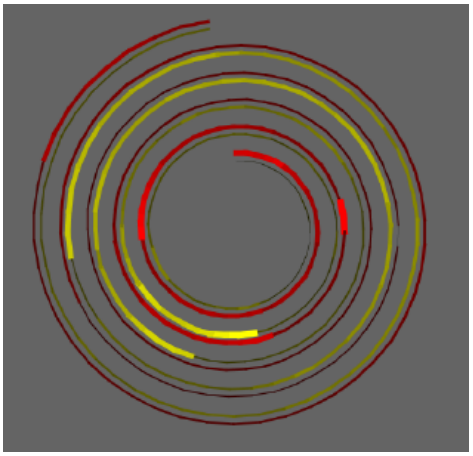
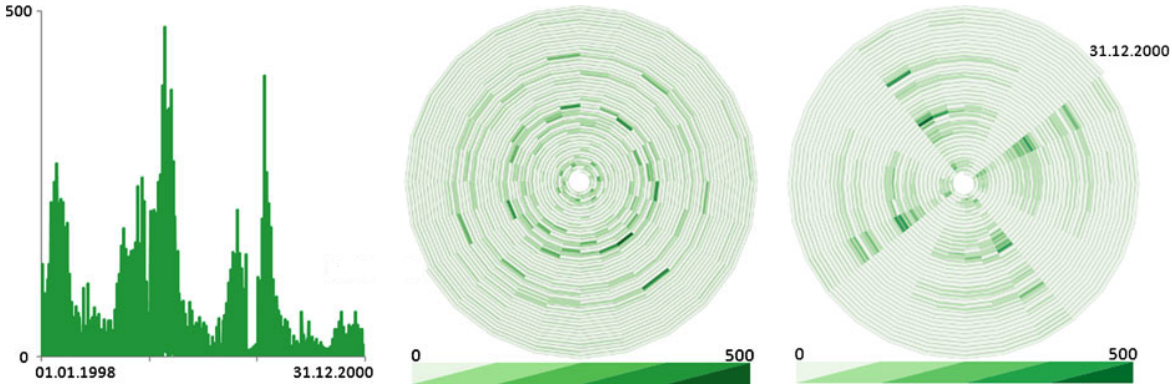


Figure 2.44: Spiral graph. Time is represented along a spiral with color intensity and line thickness are used to indicate time-dependent, numerical values. Two variables are distinguished using different color hue: yellow and red. *Image source: [183].*

Spiral graph is effective at spotting periodic patterns of the data, but it highly depends on the cycle length; i.e., the number of time steps per cycle. Figure 2.45 shows three charts using the same time-series dataset. Figure 2.45a uses a bar chart and clearly reveals trends and extreme values. The other two charts use spiral graphs; however, a cyclic pattern is only visible in Figure 2.45c. The difference between them is the cycle length: 24 days for Figure 2.45b but 28 days for Figure 2.45c, which clearly shows a pattern of four weeks. This pattern can also be revealed if the cycle length is set to a small multiple of 7 days. Interaction has been proposed to facilitate users in identifying the right cycle length [169, 183]. Users can manually adjust the cycle length. Alternatively, users can watch an animation of the visualization through different cycle lengths and stop the animation when they find the pattern of interest.



(a) Bar chart: revealing trends and extreme values.

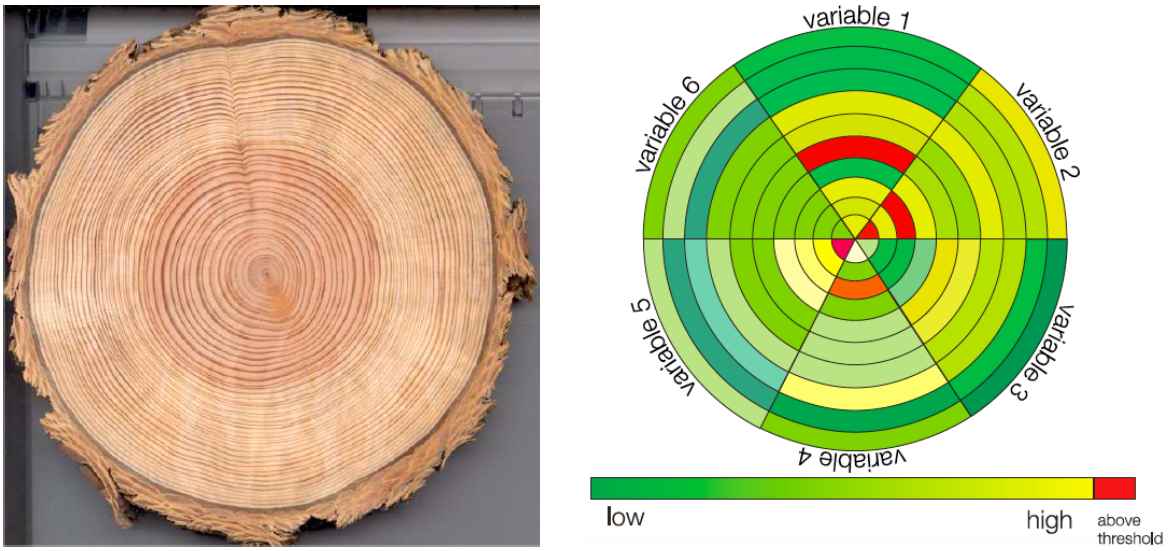
(b) Line chart: connecting data points with lines.

(c) Bar chart: showing data points as aligned bars.

Figure 2.45: Different insights can be gained from visual representations depending on whether the linear or cyclic character of the data is emphasized. *Image source: [6].*

2.5.3 Circle

Time can also be represented using a *tree-ring* metaphor. In a tree, a new layer of wood cells is produced every year, growing out from the center (Figure 2.46a). Inspired from this phenomenon, Keim, Schneidewind and Sips [91] introduce Circle View – an approach to visualize time-related multidimensional datasets. It splits a circle into multiple concentric rings, each for a time step. The circle is divided into a number of segments, each representing a variable. Figure 2.46b shows such a view with six variables. Color is used to show the (aggregated) data value for the corresponding interval.



(a) Cross section of Douglas Fir tree showing almost perfect tree-rings. *Image source: [168].*

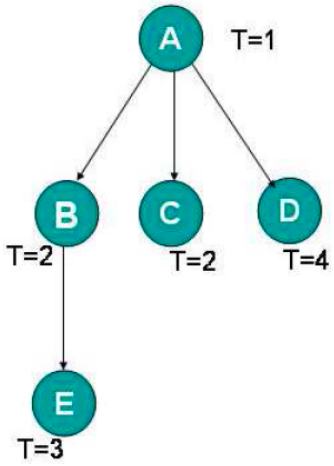
(b) Circle view showing six variables over ten time steps as concentric rings. *Image source: [91].*

Figure 2.46: Tree ring. Concentric rings expanding from the center, each indicating a time step.

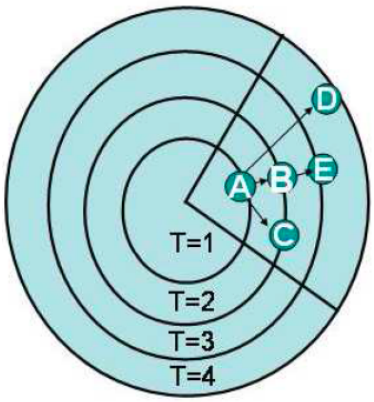
Based on tree-rings, Therón [168] develops a technique to show both temporal and hierarchical information. Figure 2.47a shows a simple example of such a dataset as tree with five nodes, each is associated with a timestamp. These nodes are assigned to the rings based on their temporal values before being positioned along the ring in such a way that arrows can be drawn from parent nodes to child nodes to reflect the hierarchy (Figure 2.47b).

2.5.4 Calendar

Another method to represent time is using a *calendar*. A day is usually color coded based on its value to reveal patterns in the data. A calendar visualization allows users to identify patterns at different temporal granularities such as daily, weekly and monthly. Figure 2.48 visualizes the daily power consumption over a year with color indicating the result of a clustering algorithm. Several patterns can be observed in this figure. First, the energy consumption is low in the summer and on Fridays. Second, it is even lower at weekends and holidays such as Christmas and New Year. Third, this figure shows the energy consumed in Netherlands, thus some patterns are clear for Dutch people such as on December 5th, employees can leave their office earlier to celebrate Santa Claus.



(a) Hierarchy is shown as a tree with temporal values are annotated next to the nodes.



(b) Hierarchy is shown as a tree embedded on a circle view with rings indicating temporal values.

Figure 2.47: Visualizing both temporal and hierarchical information using tree rings. *Image source: [168].*

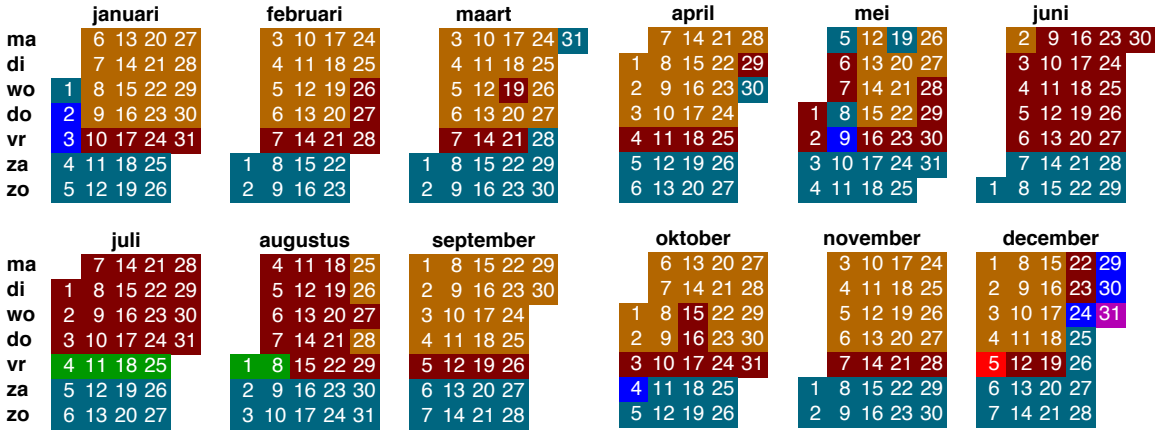


Figure 2.48: Calendar visualization of a one-year data of daily power consumption. A cell for each day is color coded to reveal data patterns. *Image source: [177].*

2.5.5 Small Multiples

Another method to depict changes over time is *small multiples* – a set of miniature visual representations placed next to each other with each showing the visualization at a selected time step [172]. Small multiples provide an overview of the data and allow users to visually compare it at different time points. The concept is general and can be applied to virtually all static visualization techniques because only

thumbnails of the visualization for each time step are required. Figure 2.49 shows an example of small multiples of bar charts.

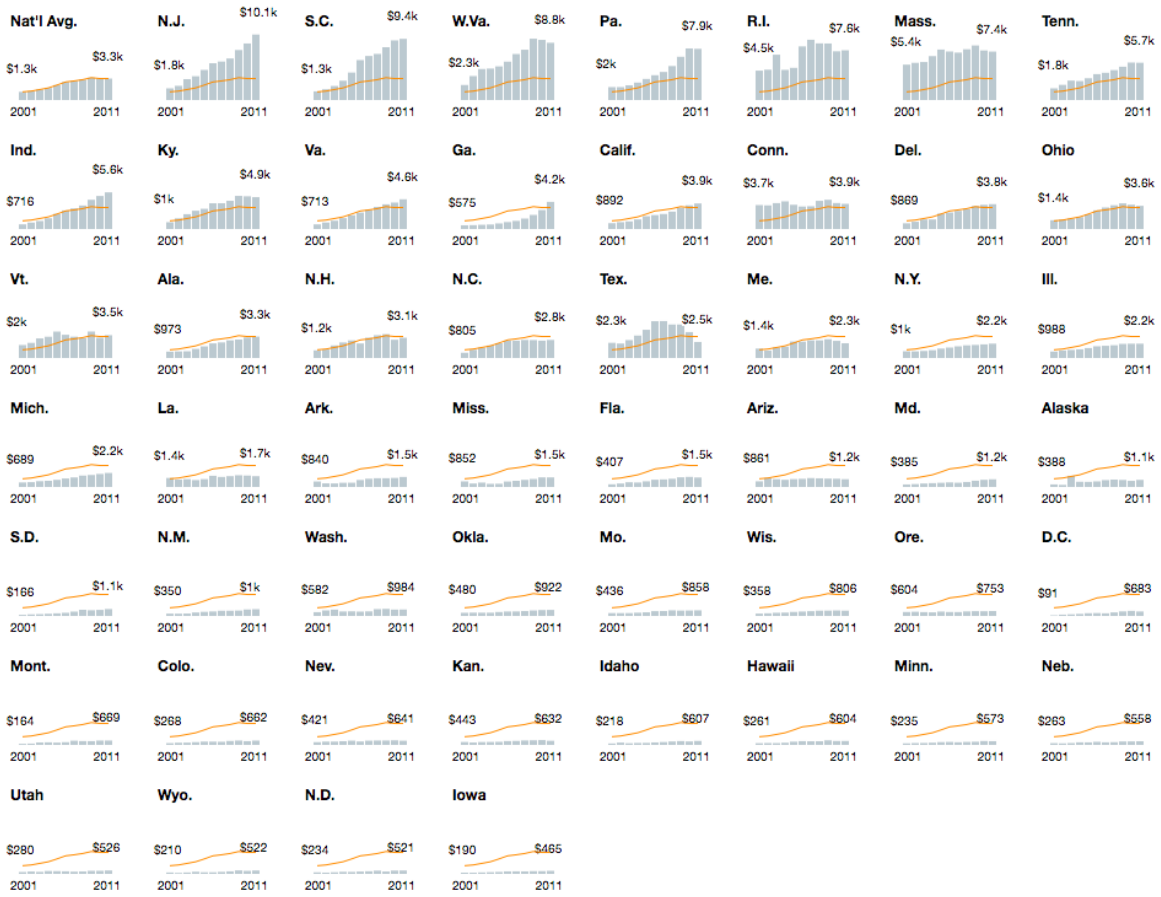


Figure 2.49: Small multiples of bar charts showing average annual Medicare spending on ambulance services per dialysis patient by U.S. state from 2001 to 2011. *Image source: [3].*

To facilitate the exploration of relationship in the data at multiple time steps, the miniatures should be interactive and linked together, rather than just static thumbnails. Figure 2.49 shows small interactive bar charts sorted decreasingly by the value spend on the last year. Alternatively, they can be ordered alphabetically by state names to facilitate searching. Standard brushing and linking interaction technique also helps compare the subsets of interest. A limitation of this method is scalability. The number of representable time steps are relatively small because of the thumbnail size.

2.5.6 Animation

Animation is another technique to convey time that can be applied to virtually all static visualization techniques. It relies on human perception in perceiving changes when a visualization smoothly updates from one time step to another. The most notable example is Trendalyzer by Gapminder Foundation [2] – an interactive visualization and presentation tool based on scatter plots. The animation is controlled via a time slider, a play/pause button, and a speed slider. Only a few data items should be animated, and they are often highlighted so that the user can keep track of the changes. Trails may also be displayed to maintain the path of a data item through time. A study by Robertson et al. [144] shows that animation is both slower and less accurate than small multiples in conveying trends over time.

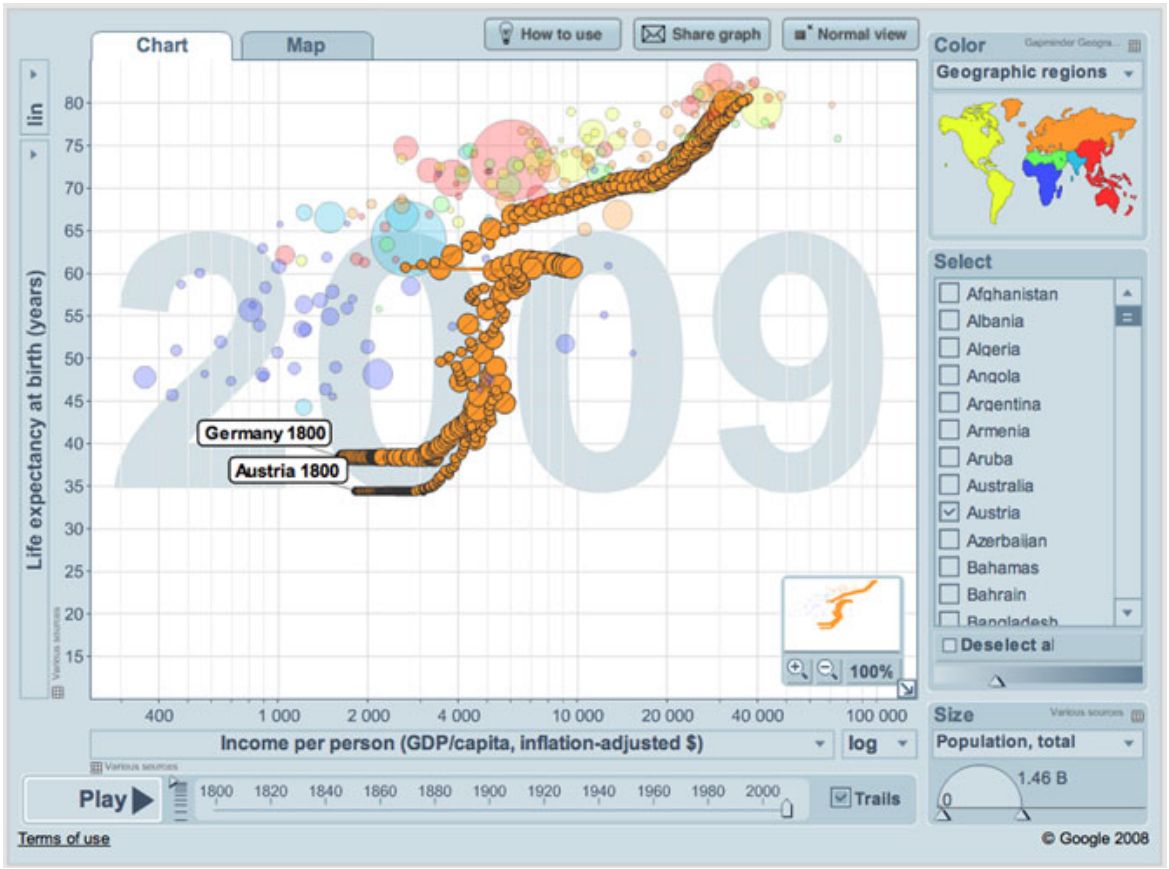


Figure 2.50: Trendalyzer interface. A scatter plot with an animation controller to traverse through time. Additionally, trails are activated for the selected countries, Austria and Germany, which help to preserve the path of a variable in animation. *Image source: [6].*

Besides showing trends of time-series data, animation has also been applied in other temporal datasets. Animation is a powerful and appealing technique in

presentation of a known story [56]. It helps illustrate computer algorithms step by step and motivate students in approaching complex problems [90]. Animation also allows reproduction of a data exploration process by interpolating visual parameters of key saved visualization steps [107].

2.6 Visualization of Network and Tree Data

2.6.1 Node-Link Diagrams

The most common visual encoding for network and tree data is *node-link diagrams*, where nodes are represented as point marks and links connecting these nodes are represented as *connection* marks. We further discuss layouts using this representation for network data with different constraints: rooted tree, directed graph and general graph.

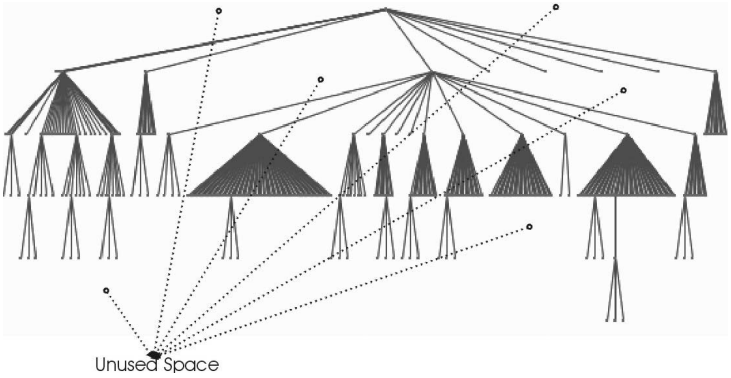
2.6.1.1 Tree Layout

A classic algorithm by Reingold and Tilford [143] produces a tidy tree layout satisfying the following aesthetic rules:

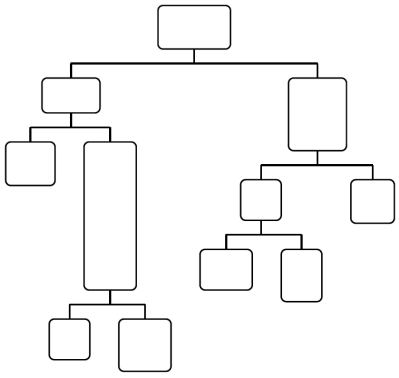
1. Nodes at the same level of the tree should lie along a straight line, and the straight lines defining the levels should be parallel.
2. A left son should be positioned to the left of its father, and a right son should be positioned to the right of its father.
3. A father should be centered over its sons.

Even though the layout produced by Reingold and Tilford's algorithm is tidy, it still leaves plenty of void space at the root side of the tree as shown in Figure 2.51a. Marriott and Sbarski [109] relax the rule that a parent must be placed at the center of its children by slightly shifting branches of nodes to produce a narrower tree. In practice, nodes have their sizes rather than just single points, and their heights can also be different. Van der Ploeg [175] relaxes the layering requirement to produce a shorter tree as shown in Figure 2.51b.

Conventionally, tree layouts are rectilinear with children branching from the parent node either from left to right or top to bottom. However, the children nodes can also be arranged radially, along a circular arc of their parent. The depth of a



(a) Produced by standard Reingold–Tilford algorithm. *Image source: [125].*



(b) Relaxed layering for shorter layout. *Image source: [175].*

Figure 2.51: Tree layouts.

circular tree is encoded as distance away from the center of the circle. Reingold–Tilford algorithm can be modified to show a radial tree. Figure 2.52 shows the class hierarchy of the *Flare* visualization toolkit [70] using both a conventional tree and a radial tree. The toolkit consists of 10 top-level classes, which determine the color of nodes in these figures.

2.6.1.2 Hierarchical Graph Layout

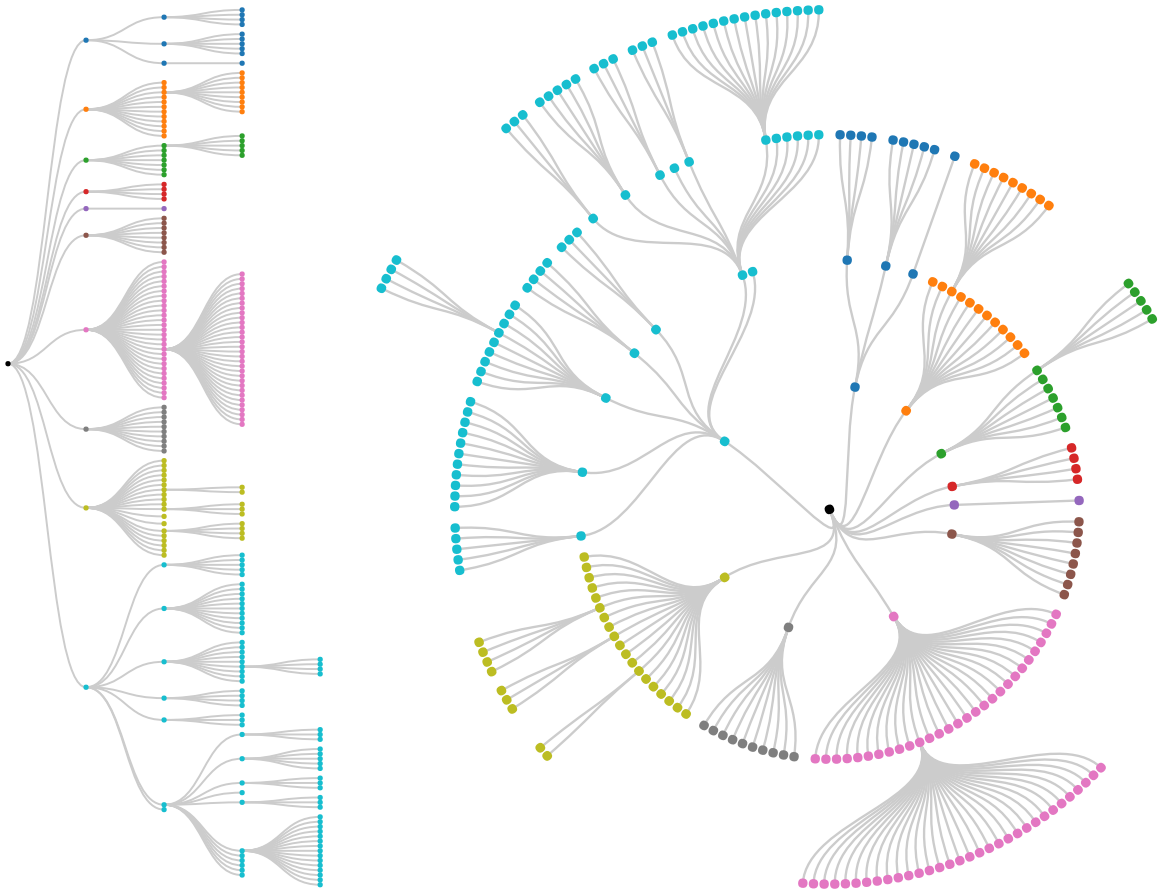
The most popular method of drawing directed graphs is the Sugiyama framework [164], separating nodes into layers to show the hierarchy effectively. Figure 2.53 shows an example of layered graphs. The framework consists of four steps.

1. *Cycle removal*. If the input graph contains directed cycles, temporarily reverse the direction of some edges to make the graph acyclic.

2. *Layer assignment*. Nodes are assigned to horizontal layers, which determines their y-coordinate.

3. *Vertex ordering*. Within each layer, the nodes are ordered to minimize edge crossings between adjacent layers.

4. *Horizontal coordinate assignment*. The x-coordinate of each node is determined, typically aiming to make edges straight.



(a) Conventional tree, growing direction as left to right.

(b) Radial tree. Depth is encoded as distance away from the center.

Figure 2.52: Tree layouts with different orientations. Data is the class hierarchy of the Flare visualization toolkit [70]. Color represents the top-level classes.

2.6.1.3 Force-Directed Layout

Force-directed layout is a popular method to visualize general graphs using node-link metaphor [47]. It positions nodes based on a simulation of physical forces: *spring-like attractive* forces attract nodes, and *repulsive* forces like those of electrically charged particles push them away from each other. The layout usually starts with a random arrangement of nodes and then iteratively refines their locations according to the behavior of the simulation. The layout stops when the simulation reaches a stable state or a maximum number of iterations. Figure 2.54a illustrates the idea of physical simulation, and Figure 2.54b shows an example with color indicating node type.

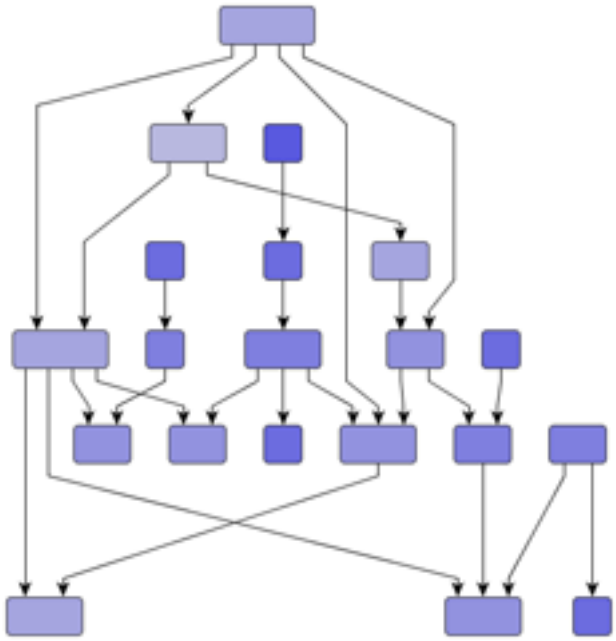
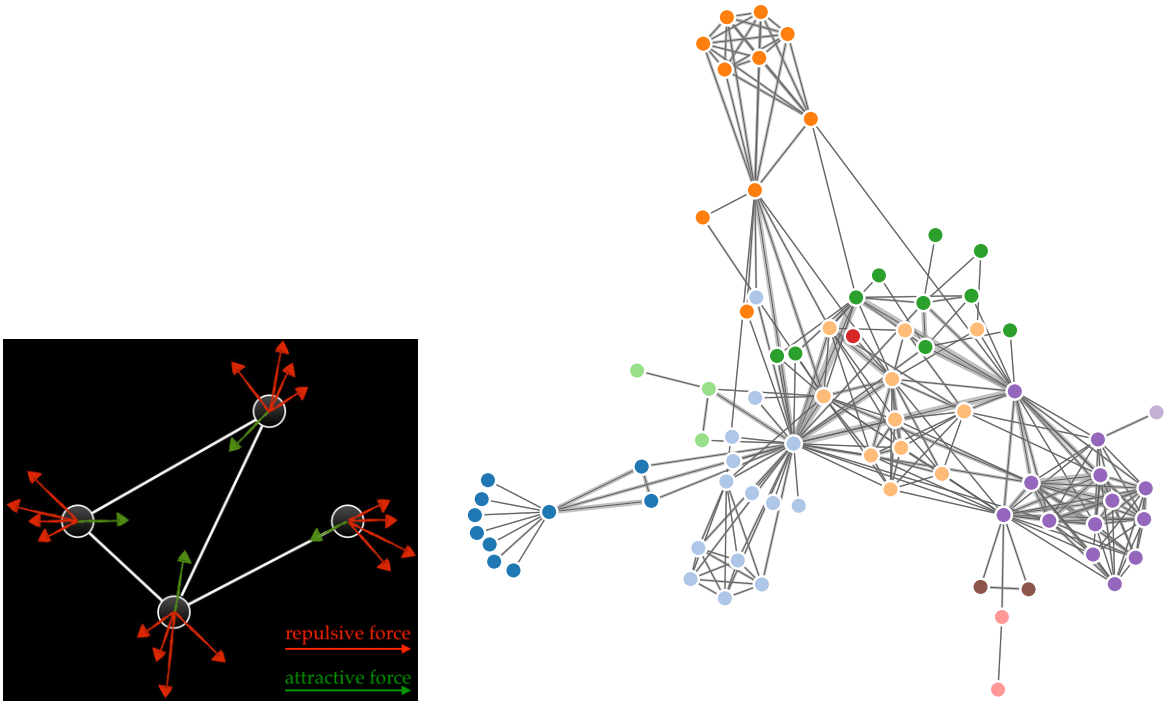


Figure 2.53: A layered graph. Nodes are assigned to horizontal layers. Within each layer, nodes are ordered to minimize edge crossings and assigned x-coordinate to make edges as straight as possible. *Image source: yWorks.*

Force-directed layout is aesthetically pleasing, aiming to produce uniform edge length, symmetry and even node distribution [52]. However, its major weakness is scalability, in terms of both the visual complexity and the running time [118]. The layout quickly degenerates into a hairball of visual clutter with even a few hundred nodes. Another limitation is its nondeterministic output: the layout looks different each time it runs, breaking user mental model.

2.6.2 Matrix Views

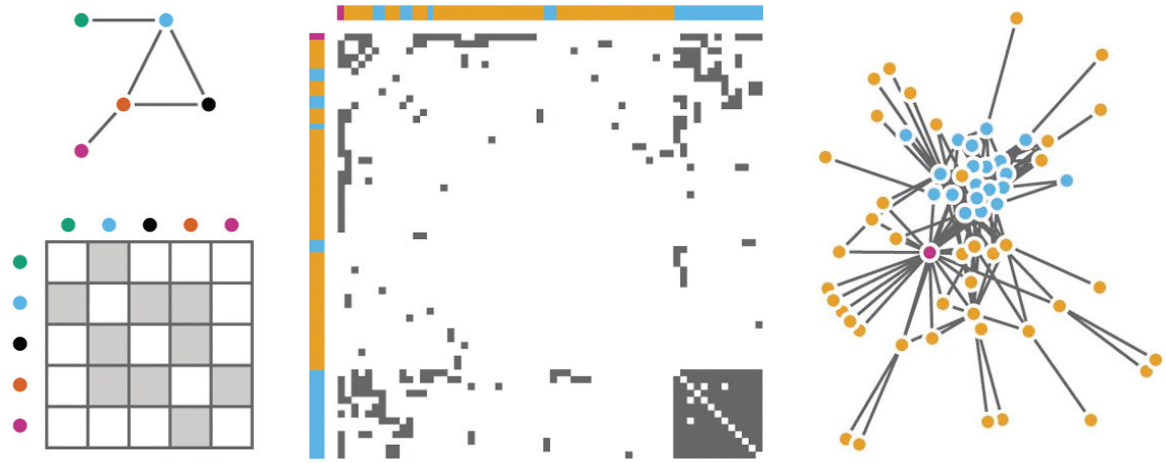
A network can be represented by an adjacency matrix. Each row and column of the matrix corresponds to a node, and a cell indicates whether the pair of corresponding nodes is connected in the network. Additional information about edges are often encoded to the visual representation of cells using color, shape and size. Matrix views can also show weighted networks, where each link associates with a quantitative value attribute, by encoding cells with an ordered visual channel such as color luminance or size. Figure 2.55 shows examples of matrix views, compared with node-link diagrams of the same dataset.



(a) A physical simulation.

(b) An example with color showing node type.

Figure 2.54: Force-directed layout.



(a) Comparison with a small network.

(b) Matrix view of larger network.

(c) Force-directed layout of larger network.

Figure 2.55: Comparison of node-link diagrams and matrix views. Gray cells indicate edge connectivity. *Image source: [118].*

A major strength of matrix views is the scalability. It can easily show a networks with thousands of nodes and millions of edges without suffering from the cluttering issue as in node-link diagrams. Matrix views are stable: adding a new node or edge

will only cause a small visual change. Whereas, adding a new item in a force-directed view may lead to a major change [118]. Nodes, as columns and rows in a matrix view, can be reordered, allowing users to reveal outliers, clusters, and patterns of the network [77].

A major weakness of matrix views is their difficulty in exploring the topological structure of the network, such as path tracing. This because they show links in a more indirect way than the direct connections of node-link diagrams – a trade-off for their strength in avoiding clutter. Another weakness of matrix views is unfamiliarity: users easily interpret small node-link diagrams but require training to understand matrix views [118]. A study shows that for many low-level abstract network tasks, node-link diagrams are best for small networks, whereas matrix views are best for large networks [57].

2.6.3 Space-Filling Techniques

Space-filling techniques only apply to tree data and uses *containment* marks to represent hierarchical relationship, placing child nodes within their parent node. Treemap [156] represents a node as a rectangle, which is recursively subdivided into smaller rectangles, each represents a child of the node. The rectangle size is proportional to a quantitative attribute of the node. The original motivation of treemap is to analyze the utilization of storage space on a hard disk. Each leaf node represents a computer file, and the node size encodes the file size. The size of a parent node simply maps to the containing folder size. Color is also commonly used to encode additional information to nodes such as file type. Figure 2.56 shows such an example of treemap.

Treemap is very effective when size is the most important feature to be displayed. It easily spots outliers of very large attribute values such as large files. However, containment is not as effective as connection in node-link diagrams for tasks focused on topological structure. It is difficult to identify the path of a given node, thus suitable for hierarchies with just a few levels. Borders of nodes or shading [185] can be used to depict the tree structure more strongly.

Alternatively, other space-filling techniques have been proposed to better represent the hierarchical structure. Figure 2.57 shows the three techniques that we discuss next using the same dataset as in Figure 2.56 for treemap.

Circle packing [180] also employs containment to represent the hierarchy, but using circles to represent nodes. Circle sizes correspond to some node values. All child nodes are packed into their parent node so that they are tangent to some of

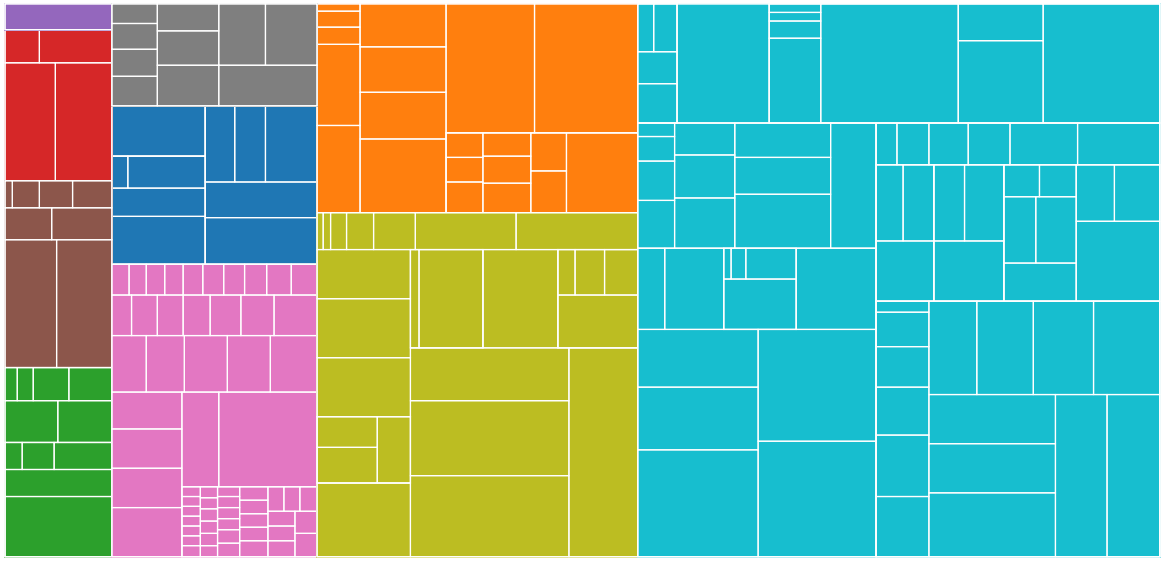
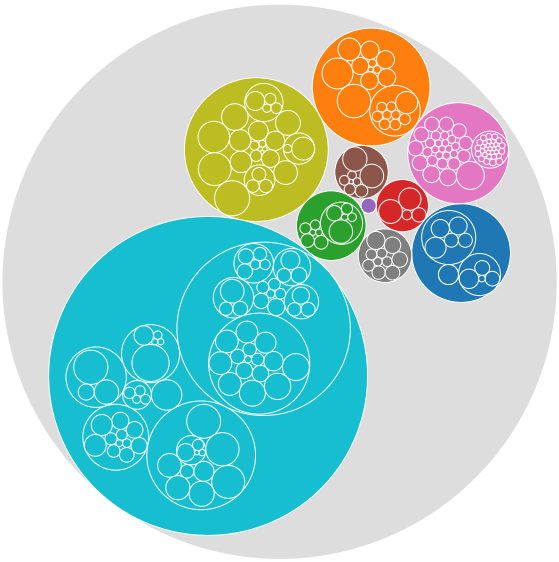


Figure 2.56: Treemap showing the class hierarchy of the Flare visualization toolkit [70]. Area represents class sizes and color represents the top-level classes.

their sibling nodes. This method uses more space than treemap but provides better hierarchy structure.

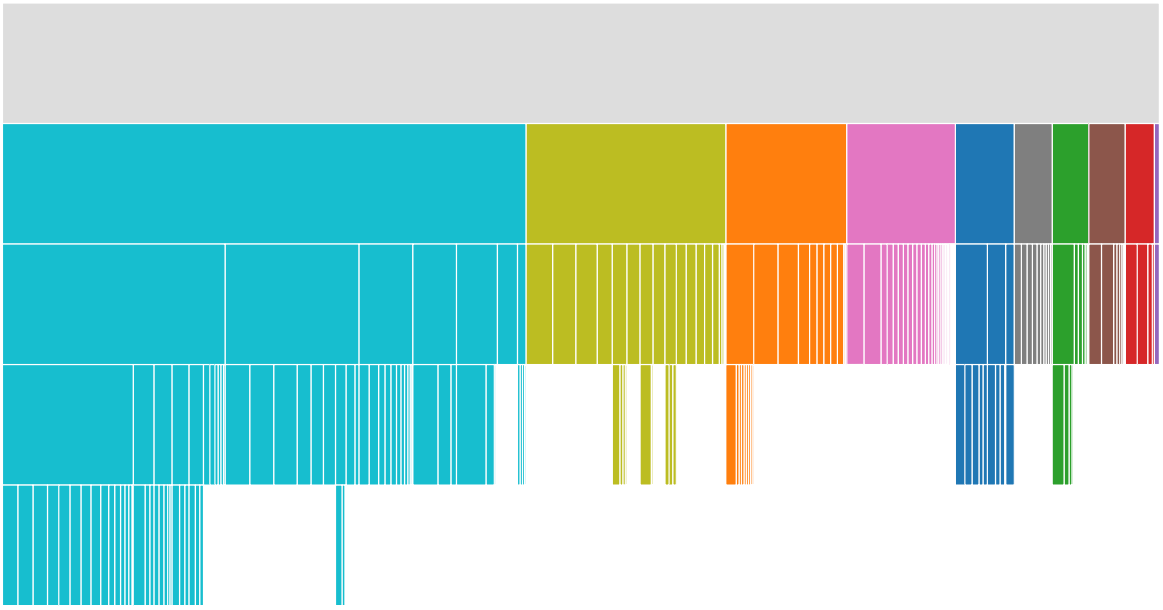
Icicle plot [98] does not strictly use containment; instead, it places child nodes *under* their parent node. Similar to treemap, icicle plot uses rectangles to represent nodes, but they all share the same height. Therefore, node width corresponds to some attribute value. Icicle plot shows parent nodes explicitly, making it more effective in tasks related to the parents such as comparing directories by size. The direct trade-off is space for showing those parent nodes. Icicle plot shows the tree structure and supports path tracing more effectively than treemap. However, small leaf nodes are very thin and difficult to read its label or interact with.

Sunburst [195] is a radial version of icicle plot. Child nodes are located under their parent node in a circular layout. The root node is at the center and deeper levels are further away from it. The angle swept out by a node, or a curved segment, corresponds to its value. Color is also commonly used to represent additional information.



(a) Circle packing.

(b) Sunburst.



(c) Icicle plot.

Figure 2.57: Other space-filling techniques. Data is the class hierarchy of the Flare visualization toolkit [70] as in Figure 2.56. Color also represents the top-level classes.