Chapter 5
Involving the Human via Interaction

> A graphic is not "drawn" once and for all; it is "constructed" and reconstructed until it reveals all the relationships constituted by the interplay of the data. The best graphic operations are those carried out by the decision-maker himself.
>
> Bertin (1981, p. 16)

The previous chapter discussed diverse options for designing visual representations that help people understand time and time-oriented data. 'Seeing' trends, correlations, and patterns in a visual representation is indeed a powerful way for people to extract knowledge from data. Yet, 'seeing' alone is not sufficient, or as Thomas and Cook (2005) put it:

> Visual representations alone cannot satisfy analytical needs. Interaction techniques are required to support the dialogue between the analyst and the data.
>
> Thomas and Cook (2005, p. 30)

From the previous chapter, we know that various aspects are involved when creating a visual representation: the characteristics of time and data, the user tasks, as well as the choice and the parametrization of visualization techniques. As a consequence, a generated visual representation might not yield the desired outcome, particularly when feeding unknown data into a visualization method. A related problem is that we sometimes do not know exactly what to expect from a visual representation or whether it is effective with regard to the task to be accomplished. One way to deal with this problem is to include the human user into the loop. So, visual exploration and analysis is not a one-way street where data are transformed into images, but it is in fact a human-in-the-loop process controlled and manipulated by one or more users.

Having said that, it becomes clear that in addition to visual methods, a high degree of interactivity and advanced interaction techniques for working with time-oriented data are important. Interaction helps users not only see the data but also understand them. By interacting, users can comprehend the visual mapping, realize the effect of visualization parameters, carve out hidden patterns, and become confident about

W. Aigner et al., *Visualization of Time-Oriented Data*, Human–Computer Interaction Series, https://doi.org/10.1007/978-1-4471-7527-8_5

the visualization and its underlying data. Users want and many times need to get their hands on their data – which is particularly true when engaging in exploratory data analyses. The importance of interaction is nicely summarized in the following statement:

> While visual representations may provoke curiosity,
> interaction provides the means to satisfy it.
>
> Tominski and Schumann (2020, p. 132)

While visualization research is naturally more focused on the visual output, the interactive operations involved in carrying out data analyses must also be considered. This chapter provides an overview of how interactivity can support the exploration of time and time-oriented data. For a deeper discussion of interaction for visualization in general, the interested reader is referred to Tominski (2015).

5.1 Motivation & User Intents

The constantly increasing size and complexity of today's datasets are major challenges for interactive visualization. Large datasets cannot simply be loaded to limited computer memory and then be mapped to an even smaller display. Users are only able to digest a fraction of the available information at a time. Complex data contain many different aspects and may stem from heterogeneous sources. As complexity increases, so does the number of questions that one might ask about the data and to which visual representations should help us find answers.

In our particular case, we need to account for the specific aspects of time and time-oriented data in the context of what, why, and how they are visualized (see Chapters 3 and 4). Any attempt to indiscriminately encode all facets of a complex time-oriented dataset into a single visual representation is condemned to failure, as this would lead to a confusing and overloaded display that users can hardly interpret.

Instead, the big problem has to be split into smaller pieces by focusing on relevant data aspects and particular tasks per visual representation. Several benefits can be gained: computational costs are reduced in a kind of divide-and-conquer way, the visual representations become more effective because they are tailored to emphasize a particular point, and users find it easier to explore and analyze the data since they can concentrate on important and task-relevant questions.

Dividing the visualization problem and separating different aspects into individual views raise the question of how users can visually access and mentally combine these. The answer is *interaction*. In an iterative process, the user will focus on different parts of the data, look at them from alternative perspectives, and actively construct answers to diverse questions. Typically, this process follows the *visual information seeking mantra*:

> "Overview first,
> zoom and filter,
> then details-on-demand."
>
> Shneiderman (1996, p. 2)

Starting with an overview, the user will first identify interesting parts of the time domain to focus on for a more detailed examination. From there, it might make sense to move on to data that are related or similar, or it might be better to return to the overview and investigate the data from a different point of view, or with regard to a different question. In other words, the user forms a mental model of the data by interactively navigating from one focus to the next, where the focus may be any part of the time domain, a certain data aspect, or a specific analysis task. While exploring data in this way, users develop understanding and insight.

The general motivation for interaction is clear now. But what specifically motivates a user to interact? An answer to this question is given in a study by Yi et al. (2007), who worked toward a deeper understanding of interaction in visualization. As already briefly mentioned in Section 1.1, they identified several user intents for interaction and introduced a list of categories that describe on a high level why users want to or need to interact. In the following, we make use of these categories and adapt them to the case of interacting with time and time-oriented data:

Select – Mark something as interesting When users spot something interesting in the visual representation, they want to mark and visually highlight it as such, be it to temporarily tag an intriguing finding or to permanently memorize important analysis results. The pieces to be marked can be manifold: interesting points in time, an entire time-dependent variable, a particular temporal pattern, or certain identified events.

Explore – Show me something else Time-oriented data are often large and can be visualized only partially. That is, only a subset of time and a subset of the time-dependent variables are visible at a time. To arrive at a full view of the data, users have to explore different subsets of the data. This includes interactively navigating the time domain to bring different parts of it to the display, and also constructing different subsets of variables to uncover multivariate temporal dependencies.

Reconfigure – Show me a different arrangement Different arrangements of time and associated data can communicate completely different aspects, a fact which becomes obvious when recalling the distinction between linear and cyclic representations of time. As users want to look at time from different angles, they need to be provided with facilities that allow them to interactively generate different spatial arrangements of time-oriented data.

Encode – Show me a different representation Similarly to what was said about the spatial arrangement, the visual encoding of data values has a major impact on what can be derived from a visual representation. Because data and tasks are versatile, users need to be able to adapt the visual encoding to suit their needs, be it to carry out location or comparison tasks, or to confirm a hypothesis generated from one visual encoding by checking it against an alternative one.

Abstract/Elaborate – Show me more or less detail During a visual analysis, users need to look at certain things in detail, while for other things schematic representations are sufficient. The hierarchically structured levels of granularity of

time are a natural match to drive such an interactive information drill-down into time-oriented data. Higher levels contain abstractions and provide aggregated overviews, whereas lower levels hold the increasingly elaborate details.

Filter – Show me something conditionally When users search for particular information in the data or evaluate a certain hypothesis about the data, it makes sense to restrict the visualization to show only those data items that match the conditions imposed by the search criteria or the hypothesis' constraints. Interactively filtering out or attenuating irrelevant data items clears the view for users to focus on those parts of the data being relevant to the task at hand.

Connect – Show me related items When users make a potentially interesting finding for one part of the data, they usually ask whether similar or related discoveries can be made in other parts of the data as well. So, users intend to interactively find, compare, and evaluate such similarities or relations. For example, for a trend discovered in one season of a certain year, it could be interesting to investigate if the trend is repeated at the same time in subsequent years.

These seven intents apply to interactive visualization, and we linked them specifically to interacting with time-oriented data. On top of that, Yi et al. (2007) mention two general interaction intents that are also relevant when exploring time.

Undo/Redo – Let me go to where I have already been Users have to navigate in time and study it at different levels of granularity, they have to try different arrangements and visual encodings, and they have to experiment with filtering conditions and similarity thresholds. A history mechanism for undoing and redoing interactions enables users to try out new views on the data and to return effortlessly to a previous visual representation if new ones did not work out as expected.

Change configuration – Let me adjust the interface In addition to adapting the visual representation to the data and the tasks at hand, it is also often necessary to configure the overall system that provides the visualization. This includes configuring not only the user interface (e.g., the arrangement of windows or the items in toolbars), but also the general management of system resources (e.g., the amount of memory to be used for undo and redo).

Taken together, the discussed intents represent on a high conceptual level what interactions a visualization system for time-oriented data should provide. For specific types of time-oriented data, additional interactions may be worth considering, such as faceting and warping for multivariate longitudinal data (see Cheng et al., 2016).

Many of the visualization approaches we describe in Appendix A support interaction of one kind or another. While marking (or selecting) interesting data items and navigation in time are quasi-mandatory, facilities for other intents are often rudimentary or not considered at all. This is often due to the extra effort one has to expend for designing and implementing effective interaction techniques. But in fact, all of the outlined user intents are equally important and corresponding techniques should be provided in order to take full advantage of the synergy of the human's skills in creative problem-solving and the machine's computational capabilities.

5.2 Interaction Fundamentals

Now that we know about the general motivation and the specific user intents behind interaction, we can move on and take a look at how interaction is actually performed. We will next describe fundamental aspects of interaction, which naturally are more general and less specific to interacting with time-oriented data.

5.2.1 Conceptual Background

Let us first look at aspects that concern interaction on a conceptual level, including how interaction can be modeled as a loop, what costs are involved when interacting, how interaction can be performed in a discrete or continues manner, and what the role of interaction latency is.

The interaction loop When users interact they express their intent to change what they see on the display, and they expect the visual representation to reflect the intended change. Consequently, Norman (2013) models interaction as a loop of two phases: an execution phase and an evaluation phase. The first phase subsumes steps that are related to the execution of interaction, including the intention to interact, the mental construction of an interaction plan, and the physical actions (e.g., pressing a button) to actually execute the plan. The second phase is related to understanding the system-generated visual feedback and involves perceiving and interpreting the feedback as well as evaluating the success of the interaction. Figure 5.1 illustrates Norman's conceptual model.

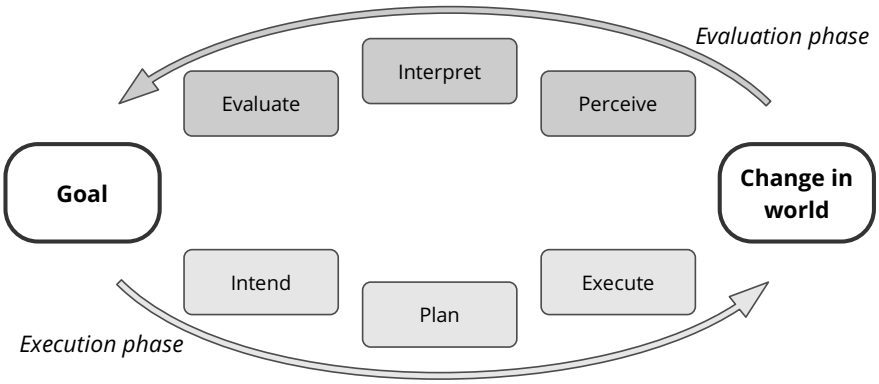


Fig. 5.1: Norman's model of interaction comprised of the execution and the evaluation phases. *The authors. Adapted from Norman (2013).*

Interaction costs The individual steps of both phases of the loop incur costs (see Lam, 2008). These costs can be physical or mental. Physical costs relate to flexing one's muscles, for example, when moving the hands to press a button or when moving the eyes to perceive the system response. Mental costs pertain to brain activities when thinking about how to achieve a goal or when interpreting the visual feedback. In a sense, the costs are associated with building bridges between the human and the system. Therefore, Norman (2013) calls the loop's phases the *gulf of execution* and the *gulf of evaluation*.

A primary goal of interaction design should be to narrow the gulfs by keeping the interaction costs low. On the execution side, this involves, for example, making interactions easy to discover and avoiding longer pointer movements through cascades of settings. On the evaluation side, it is important to let the visual response stand out clearly so that users can understand the effects of their actions easily.

Modes of interaction Technically, Jankun-Kelly et al. (2007) model the loop of user interaction as adjustments of visualization parameters, where concrete parameters can be manifold, e.g., the rotation angle of a 3D helix glyph, the focus point of a fisheye-transformed time axis, thresholds of a filter operation, or parameters that control a clustering algorithm.

Different modes of interaction can be identified depending on how parameter changes are performed. Spence (2007) distinguishes two modes of active user interaction:

- stepped interaction and
- continuous interaction.

For *stepped interaction*, a parameter change is executed in a discrete fashion. That is, the user performs one interaction step and evaluates the visual feedback. Much later, the user might perform another step of interaction. As an example, one can imagine a user looking at a visualization of the data at the granularity of years. If more details are required, the user might take an interaction step to switch to a finer granularity of months.

The term *continuous interaction* is used to describe interaction for which a visualization parameter is changed at a higher frequency. The user continuously performs an action and evaluates the generated feedback for a sustained period of time. This enables the user to quickly scan a larger range of parameter values and their corresponding visual representations. As such, continuous interaction is particularly useful in the context of exploratory 'what if' analyses of time-oriented data.

An example would be the adjustment of the cycle length for a spiral visualization in order to find out if and if so, which cyclic patterns exist in the data. For stepped interaction, the user has to explicitly specify different cycle lengths in a successive manner (e.g., by entering a numeric value). The stepped approach is quite time-consuming already when exploring only a moderate number of possible parameter values. Moreover, the discrete stepping does not allow cyclic patterns to emerge naturally as different cycle lengths are tried out. With continuous interaction (e.g., by dragging a slider), the user can explore any parameter range at any speed with a single

continuous action. The risk of missing interesting patterns is reduced because cyclic patterns would crystallize gradually as suitable parameter values are approached. An important requirement though is to keep the interaction latency low.

Interaction latency For smooth and efficient interaction, the ensemble of visual and interaction methods has to generate feedback in a timely manner (within 50 - 100 ms according to Shneiderman (1994) and Spence (2007)). However, time-oriented data tend to be large and can pose considerable computational challenges. On the one hand, mapping and rendering the visual representation takes time, particularly if complex visual abstractions have to be displayed. On the other hand, computational methods (see Chapter 6) involved in the visualization process consume processing time before generating results. The adverse implication for interaction is that visual feedback might lag, disrupting the interaction loop (see Liu and Heer, 2014).

Another aspect adds to the time costs for presenting visual feedback. As interaction involves change, we want users to understand what is happening. However, abrupt changes in the visual display will hurt the mental model that users are developing while exploring unknown data. Pulo (2007) and Heer and Robertson (2007) provide evidence that smoothly interpolating the parameter change and applying animation to present the visual feedback can be a better solution. However, animation consumes time as well, not to mention the possibly costly calculations for interpolating parameter changes.

Thus, there are two conflicting requirements. On the one hand, interaction needs synchronicity. An interactive system has to be responsive at all times and should provide visual feedback immediately. From the interaction perspective, a system that is blocked and unresponsive while computing is the worst scenario. On the other hand, interaction needs asynchronicity – for both generating the feedback (i.e., computation) and presenting the feedback (i.e., animation). The difficulty is to integrate synchronicity and asynchronicity. One option to address this difficulty is to take a progressive approach.

Progressive visualization The goal of progressive visualization is to facilitate a smooth interaction cycle by generating visual feedback as quickly as possible (see Stolper et al., 2014; Angelini et al., 2018). This is achieved by a *divide & conquer* approach: Long-running computations are subdivided into smaller steps, and these operate on smaller data chunks rather than the whole dataset. For time-oriented data, data chunks can be obtained simply by sampling with respect to the dimension of time, by considering the semantics of time (e.g., workdays vs. weekends or day vs. night), or based on the increasingly detailed granularities of time (e.g., yearly, monthly, or daily data). The subdivision of computations into smaller steps depends very much on the concrete algorithms involved in the analytical and visual transformation of the data.

Working in smaller steps and on smaller data, progressive visualization generates a series of preliminary or partial results of increasing quality until a complete final image of the entire data is rendered. The quick and incremental generation of partial results has several advantages. First of all, the system is responsive at all times, and the interaction loop can run smoothly, even if there are still some computations running in

the background. Second, users can observe the system computing the visualization. This makes otherwise hidden calculations more transparent and understandable. Third, as partial results arrive, users can early on develop an idea of the data and, if necessary, can steer the running computations to more fruitful results. For example, if partial results do not show the expected outcome, the computations can be canceled early to stop wasting time. If partial results already show promising features in the data, these parts can be prioritized to further crystallize interesting patterns early on.

Overall, we can see that interaction is a human-in-the-loop process during which a diverse set of user intents have to be satisfied. For the user, costs should be kept low, which requires interactions that are easy to carry out and visual feedback that is easy to understand. From a technical perspective, the execution and evaluation phases of the interaction loop must run smoothly, which can be supported by progressive visualization. What ultimately counts is that both user concerns and technical aspects are addressed under the umbrella of an effective and cost-efficient user interface.

5.2.2 User Interface

The user interface is the channel through which a human and a machine exchange information (i.e., interaction input and visual feedback). This interface is the linchpin of interactive visual exploration and analysis of time-oriented data. Any visual representation is useless if the user interface fails to present it to the user in an appropriate way, and the diversity of available visualization techniques lies idle if the user interface fails to provide interactive access to them. In order to succeed, the user interface has to bridge the gap between the technical aspects of a visualization approach and the users' mental models of the problems to be solved. In this regard, Cooper et al. state:

> [...] user interfaces that are consistent with users' mental models are vastly superior to those that are merely reflections of the implementation model.
>
> Cooper et al. (2007, p. 30)

The user interface is responsible for numerous tasks. It has to provide visual access to time-oriented data and to information about the visualization process itself at different levels of graphical and semantic detail. Appropriate controls need to be integrated to allow users to steer exploration and analysis with regard to the interaction intents mentioned before, including marking interesting points in time, navigating in time at different levels of granularity, rearranging data items and elements of the visual representation, or filtering for relevant conditions. Moreover, the user interface has to support bookkeeping in terms of the annotation of findings, storage of results, and management of the working history (undo/redo).

In general, the user interface has to offer facilities to present information to the user and to accept interaction input from the user. This separation is reflected in the *model-view-controller* (MVC) architecture by Krasner and Pope (1988), where views provide visual representations of some model (in our case time, time-oriented data,

and visualization parameters) and controllers serve for interactive (or automatic) manipulation of the model. Next, we look at visualization views and interaction controls in more detail.

Visualization views Especially the different temporal granularities make it necessary to present the data at different levels of graphical and semantic detail. Overview+detail, focus+context, and multiple coordinated views are key strategies to address this demand.

Overview+detail methods present overview and detail separately. The separation can be either spatial or temporal. Spatial separation means that separate views show detail and overview. For example, on the bottom of Figure 5.2, an overview shows the entire time domain at a high level of abstraction. On top of the overview, there is a separate detail view, which shows the data in full detail (i.e., detailed planning information), but only for a narrow time interval. Temporal separation means a view is capable of showing any level of detail, but only one at a time. This is usually referred to as *zooming*, where the user can interactively zoom into details or return to an overview. *Geometric zooming* operates solely in the presentation space to scale a visual representation, whereas *semantic zooming* denotes zooming that can go beyond purely geometrical scaling and may involve recoding the data in the presentation space as well as in the data space depending on the zoom level.

Contrary to overview+detail, *focus+context* methods smoothly integrate detail and overview. For the user-chosen focus, full detail is presented, and the focus is embedded into a less-detailed display of the context. Figure 5.3 shows the perspective wall technique (⟶ p. 256) as a prominent example of the focus+context approach. Cockburn et al. (2009) provide a comprehensive survey of overview+detail, zooming, and focus+context and discuss the advantages and disadvantages of these concepts.

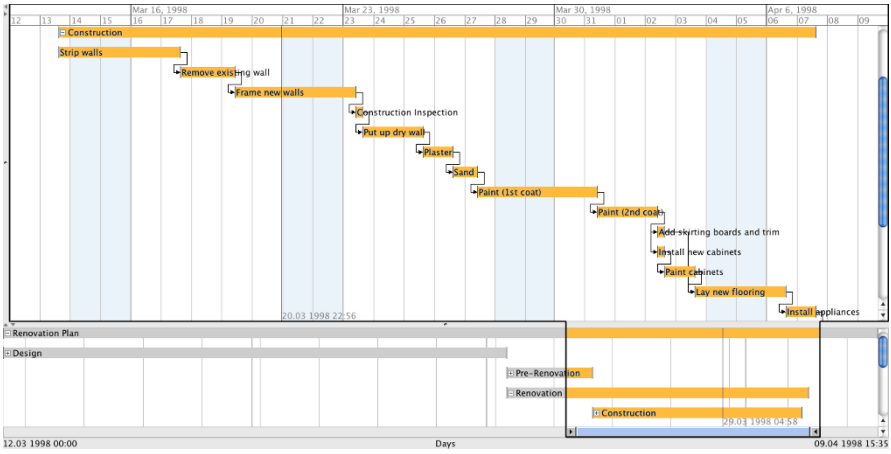


Fig. 5.2: Overview+detail. The detail view at the top shows individual steps of the construction phase of a renovation plan. In the overview at the bottom, the entire project is shown, including the design, pre-renovation, renovation, and construction phases.

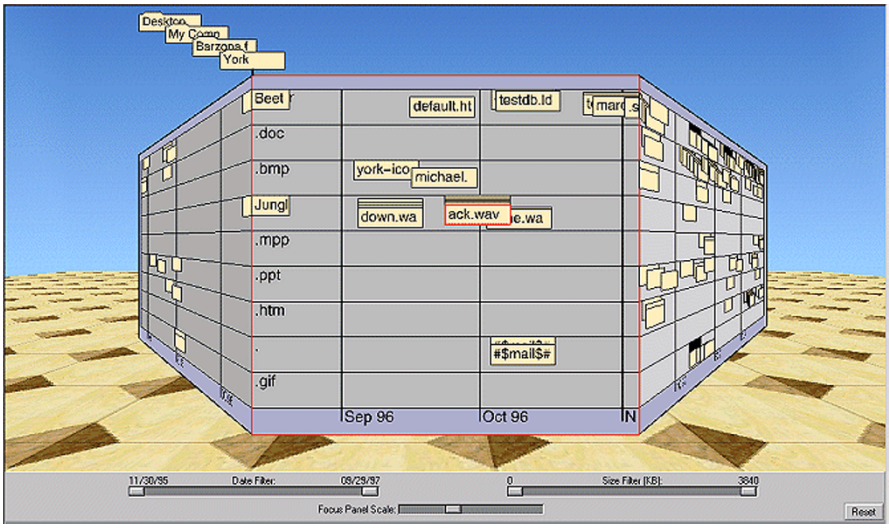


Fig. 5.3: Focus+context. The center of the perspective wall shows the focus in full detail. The focus is flanked on both sides by context regions. Due to perspective distortion, the context regions intentionally decrease in size and show less detail. © *Inxight Federal Systems. Used with permission.*

When visualizing time-oriented data, it is also often helpful to provide *multiple coordinated views*,[1] each of which is dedicated to particular aspects of time, certain data subsets, or specific visualization tasks. When there are multiple views, the user interface obviously needs a strategy for arranging them. One option is to use a fixed arrangement that has been designed by an expert and has proved to be efficient. It is also possible to provide users with the full flexibility of windowing systems, allowing them to move and resize views arbitrarily. Both options have their advantages and disadvantages and both are actually applied. An interesting third alternative is to maintain the flexibility of user-controlled arrangements, but to impose certain constraints in terms of what arrangements are possible (e.g., disallow partially overlapping views or enforce adjacency of related views). Irrespective of the strategy applied, the visualization should be *responsive* in the sense that it automatically adjusts itself to match the size and the aspect ratio of views (see Hoffswell et al., 2020).

In addition to arranging multiple views, coordinating the views plays an important role. Views are coordinated to help develop and maintain a consistent overall image of the visualized data. This means that an interaction which is initiated in one view is automatically propagated to all coordinated views, which in turn update themselves to reflect the change visually. A practical example is browsing in time. When the user navigates to a particular range of the time axis in one view, all other views (that are coordinated) follow the navigation automatically, which otherwise would be a cumbersome task to be manually accomplished by the user on a per-view

[1] Baldonado et al. (2000) provide general guidelines for when to use multiple coordinated views.

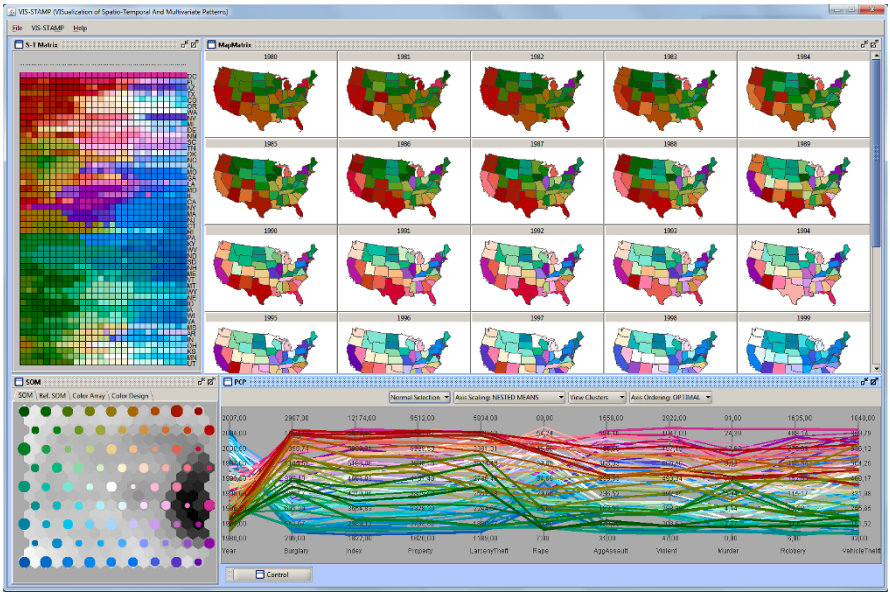


Fig. 5.4: Multiple coordinated views. Analysts can look at the data from different perspectives. The views are coordinated, which means selecting objects in one view will automatically highlight them in all other views as well. *The authors. Generated with the VIS-STAMP system by Guo et al. (2006).*

basis. Figure 5.4 shows an example where multiple coordinated views are applied to visualize spatio-temporal data in the VIS-STAMP system (↪ p. 380).

Interaction controls In addition to one or several visualization views, the user interface also consists of various interaction controls to enable users to tune the visualization process to the data and task at hand. Figure 5.5 shows a simple example with a single spiral view to its left (see Tominski and Schumann (2008) and ↪ p. 274). Already this single view depends on a number of parameters for which a corresponding number of controls must be provided in the control panel to the right. The control panel contains sliders for continuous adjustments of parameters such as *segments per cycle*, *spiral width*, and *center offset*. Buttons, drop boxes, and custom controls are provided for selecting different modes of encoding (e.g., adjusting individual colors or choosing different color scales).

In this example, user input (e.g., pressing a button or dragging a slider) is immediately committed to the system, which is a requirement for *continuous interaction*. However, this puts high demands on the system in terms of generating visual feedback quickly at interactive rates (see Piringer et al., 2009). Therefore, a commonly applied alternative is to allow users to perform a number of adjustments and to commit the adjustments as a single transaction only when the user presses an "Apply" button, which corresponds to *stepped interaction*.

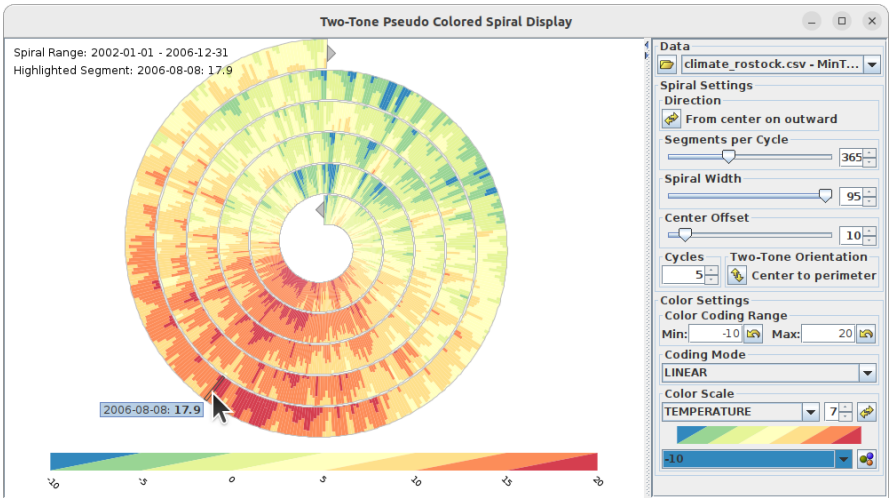


Fig. 5.5: User interface for a spiral visualization. The interface consists of one spiral view and one control panel, which in turn consists of various controls to adjust visualization parameters. *The authors.*

Certainly, there are visualization parameters that are adjusted more often than others during interactive visual exploration. Resources should preferably be spent on facilitating continuous interaction for important parameters. Moreover, Gajos et al. (2006) provide evidence that duplicating important functionality from an all-encompassing control panel to an exposed position is a useful way to drive adaptable user interfaces. For example, toolbars allow for interaction that is most frequently used, whereas rarely applied tools have to be selected from an otherwise collapsed menu structure.

5.3 Basic Interaction with Time-Oriented Data

It is clear now that we need visualization views on the one hand, and interaction controls on the other hand. Views are usually equipped with visual data representations, of which we described many examples for time and time-oriented data in the previous chapters. Let us now take a closer look at interactive means of controlling the visualization beyond standard graphical user interface controls. To this end, we briefly describe navigation in time, direct manipulation, brushing & linking, and dynamic queries as key methods for the interactive exploration and analysis of time-oriented data.

5.3.1 Navigation in Time

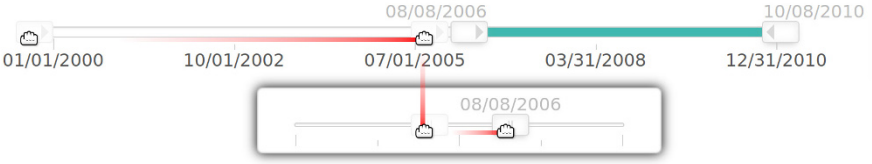
Time-oriented data typically contain very many time primitives, often too many to be displayed in a single visual representation. As a consequence, usually only a part of the time axis is visible at a time, and users have to navigate in time in order to develop a comprehensive understanding of the data. This navigation in time is essential.

Interactive sliders are control elements commonly found in user interfaces facilitating the exploration of data. For the case of time-oriented data, standard sliders are usually not enough for two reasons. First, standard sliders only have one handle to set a single value. For navigating in time, however, often two handles are required for defining the time interval to be visualized. One handle is for adjusting the interval's start, and the other handle sets the interval's end. Second, a standard slider cannot provide precise access to the time domain when the number of time primitives exceeds the interaction resolution. What is needed is a slider that can operate on different scales to facilitate quick and still precise access to all parts of time.

Figure 5.6 illustrates how such a slider may work for a time axis that extends from January 1, 2000 to December 31, 2010. In Figure 5.6b, the right handle has already been set to October 8, 2010. The figure further shows how the user can easily and accurately adjust the left handle to August 8, 2006. The interaction starts by horizontally dragging the handle roughly toward the desired date. Then the cursor is dragged in a downward movement to trigger the dynamic appearance of a higher-resolution on-demand slider. The interaction continues there horizontally, and thanks to the higher precision, the desired start date can be set exactly, which would not have been possible with the main slider alone.

Navigation in time via dedicated sliders is a widely applied approach. In the following, we will learn that interaction can also be performed directly on the visual representation of the data.

(a) Two-handle slider for navigating time.



(b) Interaction gesture for dual-scale interval adjustment.

Fig. 5.6: Navigation in time with a two-handle slider. (a) The slider's handles define the start and end of the time interval to be visualized. (b) Using a continuous interaction gesture, the interval start is adjusted coarsely on the main slider and fine-tuned precisely on a higher-resolution on-demand slider. *The authors. Adapted from Tominski and Schumann (2020).*

5.3.2 Direct Manipulation

Graphical controls in user interfaces often have the advantage of being standardized components (e.g., buttons, single-handle sliders, and value spinners), which are easy to integrate and use. However, a disadvantage is that visual feedback usually does not appear where the interaction is performed. Recall the example from Figure 5.5 where the interaction takes place in the control panel to the right, whereas visual feedback is displayed in the visualization view to the left. Direct manipulation as introduced by Shneiderman (1983) is the classic means to address this disadvantage.

The goal is to enable users to manipulate the visual representation directly without a detour. To this end, a visualization view or its graphical elements are implemented so as to be responsive to user input. A visualization may for instance allow zooming into details under the mouse cursor simply by rotating the mouse wheel, or visiting different parts of the visual representation simply by dragging the view. Such functionality is often present in zoomable user interfaces (see Cockburn et al., 2009). Virtual trackballs (see Henriksen et al., 2004) are more object-centric in that they allow users to grab and rotate virtual objects to view them from different angles.

In terms of interacting with visual representations of time-oriented data, we just learned that navigating time is of particular importance. To support navigation, many tools rely on standard slider or calendar controls in the user interface. However, for direct manipulation, the interaction has to be tightly coupled with the display of the data. We explain what this means by two examples.

First, we take a look at DimpVis (↪ p. 305), which facilitates navigation to points in time (see Kondo and Collins, 2014). Figure 5.7 shows DimpVis in action on a basic point plot. The interaction works as follows. When the user grabs a dot, a path shows up indicating the selected data item's trajectory through time. In order to navigate, the user can now drag the dot along the path, where intermediate labels help the user find the desired moment in time. In a sense, DimpVis works like a slider, only the sliding operates on a curved path, rather than a straight line.

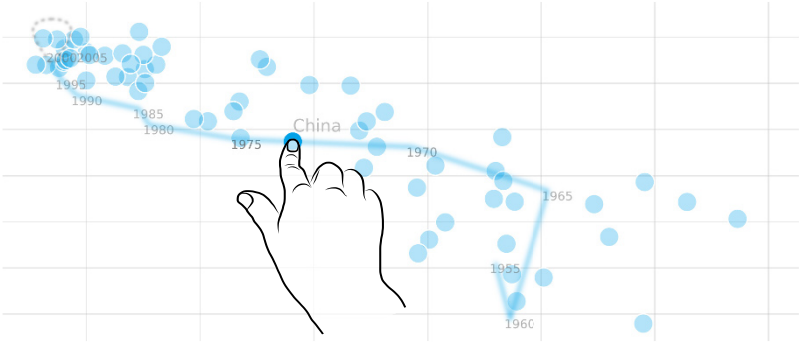
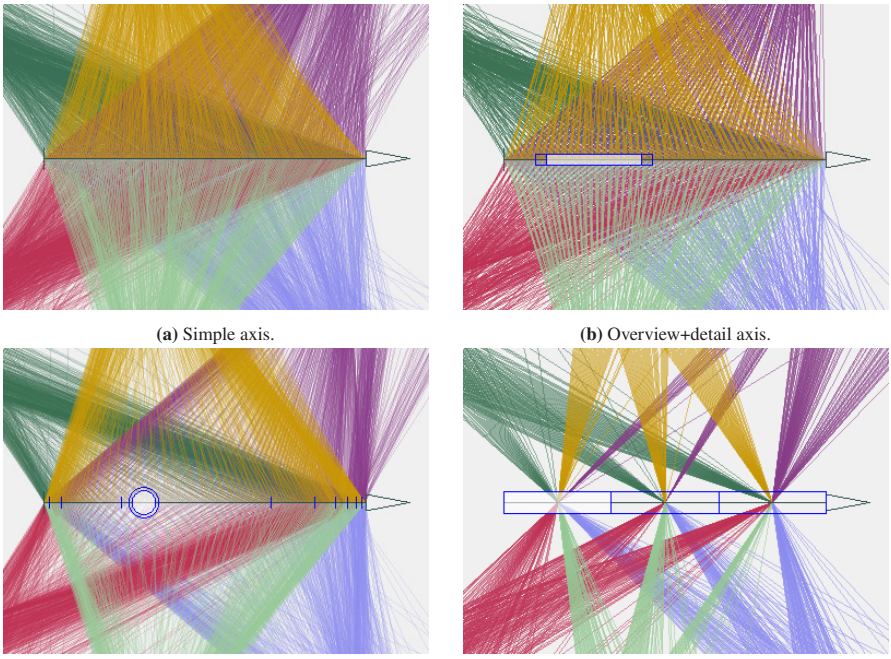


Fig. 5.7: Navigation in time via dragging a data item along its trajectory through time. *The authors. Generated with the DimpVis software by Kondo and Collins (2014).*



(a) Simple axis.

(b) Overview+detail axis.

(c) Focus+context axis.

(d) Hierarchical axis.

Fig. 5.8: The TimeWheel's mapping of time along the time axis can be manipulated directly in different ways. The simple axis uses a fixed linear mapping of time. The overview+detail axis allows users to select any particular range of the time domain to be mapped linearly to the time axis. The focus+context axis can be used to untangle dense parts of the time domain by applying a non-linear mapping. The hierarchical axis represents time at different levels of granularity, where individual axis segments can be expanded and collapsed.

For a second example of direct manipulation, we refer to the TimeWheel (↪ p. 298), in particular to its interactive axes (see Tominski et al., 2004). As Figure 5.8 illustrates, the TimeWheel provides (a) a simple non-interactive axis and three types of interactive axes: (b) overview+detail axis, (c) focus+context axis, and (d) hierarchical axis. Each of the axes displays time and the interactive ones offer different options for direct manipulation. The overview+detail axis basically extends the simple axis with three interactive handles to control the position and extent of the time interval to be displayed in the TimeWheel, effectively allowing users to zoom and scroll into any particular part of the data. The focus+context axis applies a non-linear distortion to the time axis in order to provide more drawing space for the user's focus and less space for the context. This allows users to untangle dense parts of the data. Finally, for the hierarchical axis, the display is hierarchically subdivided into segments according to the different granularities of time (e.g., years, quarters, months, and days). Users can expand or collapse these segments interactively to view the data at different levels of abstraction.

The advantage of directly manipulating the visual representation is, as indicated, that interaction and visual feedback take place at the very same location. However, direct manipulation always involves some learning and training of the interaction facilities provided (see Schwab et al., 2019a). This is necessary because most of the time the interaction is not standardized but custom-made to fit the visual mapping.

5.3.3 Brushing & Linking

Brushing & linking is a classic interaction concept that takes up the idea of direct manipulation. Becker and Cleveland (1987) describe brushing as a technique that enables users to select interesting data items directly from a data display. There are various options for selecting data items. We will often find brushing being implemented as point and click interaction to select individual data items. Rubber-band and lasso interaction serve the purpose of brushing subranges in the data or multiple data items at once. Hauser et al. (2002) introduce brushing based on angles between data items, and Doleisch and Hauser (2002) go beyond binary selection to allow for smooth brushing (i.e., data items can be partially selected).

After brushing, selected data items are highlighted in order to make them stand out against the rest of the data. The key benefit of the *linking* part of brushing & linking is that data brushed in one view are automatically highlighted in all other views. In this sense, brushing & linking is a form of coordination among multiple views. This is especially useful when visualizing the variables of a multivariate time-oriented dataset individually in separate views: Brushing a temporal interval of interest in one view will highlight the same interval and corresponding data values in all views. This makes it easy for users to compare how the individual variables develop during the brushed time period.

For complex data, using a single brush is often unsatisfactory. Instead, users need to perform multiple brushes on different time-dependent variables or in different views. Compound brushing as explained by Chen (2004) allows users to combine individual brushes into composite brushes by using various operators, including logical, analytical, data-centric, and visual operations. With such facilities, brushing is much like a visual query mechanism.

5.3.4 Dynamic Queries

Shneiderman (1994) describes dynamic queries as a concept for visual information seeking. It is strongly related to brushing & linking in that the goal is to focus on data of interest, which in the case of dynamic queries is often realized by filtering out irrelevant data. Because time-oriented data are often large, dynamically omitting data with respect to task-specific conditions can be very helpful.

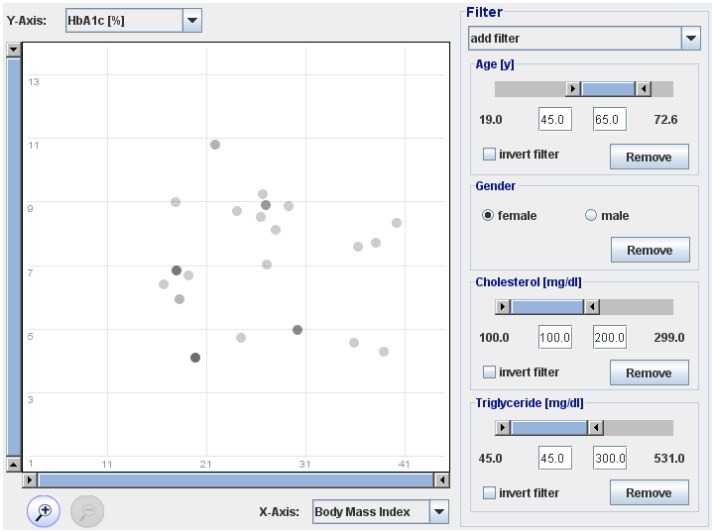


Fig. 5.9: Several filters can be adjusted in order to dynamically restrict the scatter plot visualization to data items that conform with the formulated conditions. *The authors.*

Depending on the view characteristics and visualization tasks, two alternatives can be applied to display filtering results: filtered objects can be displayed in less detail or they can be made invisible. Reducing detail is useful in views that maintain an overview, where all information needs to be displayed at all times, but filtered objects need only to be indicated. Making objects invisible is useful in views that notoriously suffer from cluttering.

Filter conditions are usually specified using dedicated mechanisms. Threshold or range sliders are effective for filtering time or any particular numerical variable; textual filters are useful for extracting objects with specific labels (e.g., data tagged by season). Similar to what has been said for brushing & linking, the next logical step is to combine filters to provide some form of multidimensional data reduction. For instance, a logical And combination generates a filter that can be passed only if an object obeys all filter conditions; an object can pass a logical Or filter if it satisfies any of the involved filter conditions. Figure 5.9 shows an example of a dynamic query interface.

While some systems offer only fixed filter combinations or require users to enter syntactic constructs of some filter language, others implement a visual interface where the user can interactively specify filter conditions. Examples of querying time-oriented data that are visualized as line plot (↪ p. 233) are timeboxes and relaxed selection techniques.

Timeboxes (↪ p. 290) by Hochheiser and Shneiderman (2004) are used to filter out variables of a multivariate line plot. To this end, the user marks regions in the visual display by creating one or more elastic rectangles that specify particular value ranges and time intervals. The system then filters out all variables whose plots do not

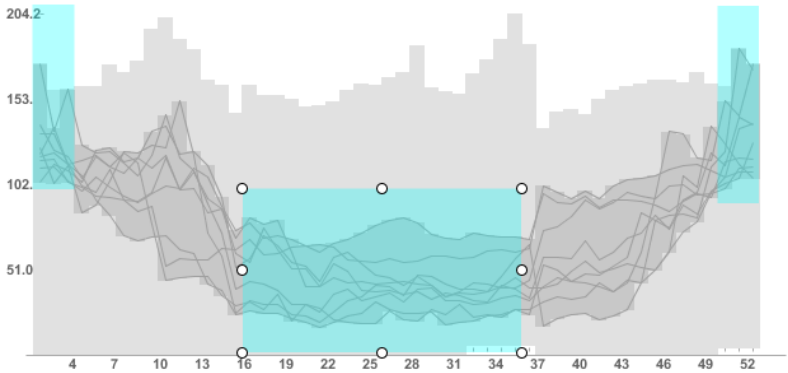


Fig. 5.10: Three timeboxes are used to dynamically query stock data. Only those stocks are displayed that are high at the beginning, but low in the middle, and again high at the end of the year. *The authors. Generated with the TimeSearcher software by Hochheiser and Shneiderman (2004).*

overlap with the rectangles, effectively performing multiple AND-combined range queries on the data. Figure 5.10 depicts a query that combines three timeboxes to restrict the display to stocks that performed well in the first and the last weeks of the year, but had a bad performance in the middle of the year.

The relaxed selection techniques by Holz and Feiner (2009) are useful for finding specific patterns in the data. For that purpose, the user specifies a query pattern by sketching it directly on the display. When the user is performing the sketching, either the distance of the sketch to the line plot or the user's sketching speed is taken into consideration in order to locally relax the query pattern. This relaxation is necessary to allow for a certain tolerance when matching the pattern in the data. An interactive display of the query sketch can be used to fine-tune the query pattern. Once the query pattern is specified, the system computes corresponding pattern matches and displays them in the line plot as depicted in Figure 5.11.

We should acknowledge that carrying out interactions directly on the visual representation as illustrated in this section is definitely useful, but the user can mark only those things that are already in the data and are actually displayed. Formulating queries with regard to potential but not yet existing patterns in the data beyond some tolerance requires additional formal query languages, and their utility hinges on the interface exposed to the user (see Monroe et al., 2013a).

Overall, navigating in time, direct manipulation, brushing & linking, and dynamic queries form an interaction vocabulary that any visualization of time-oriented data should support. Despite the advantages of being able to dynamically focus on data that are relevant to the task at hand, this vocabulary has still not yet become standard. While virtually all visualization tools for time-oriented data offer navigation in time, many do so using only rudimentary means that require users to take discrete steps rather than allowing them to browse the data in a continuous manner. Brushing the data directly in the visual representation and constructing more complex dynamic

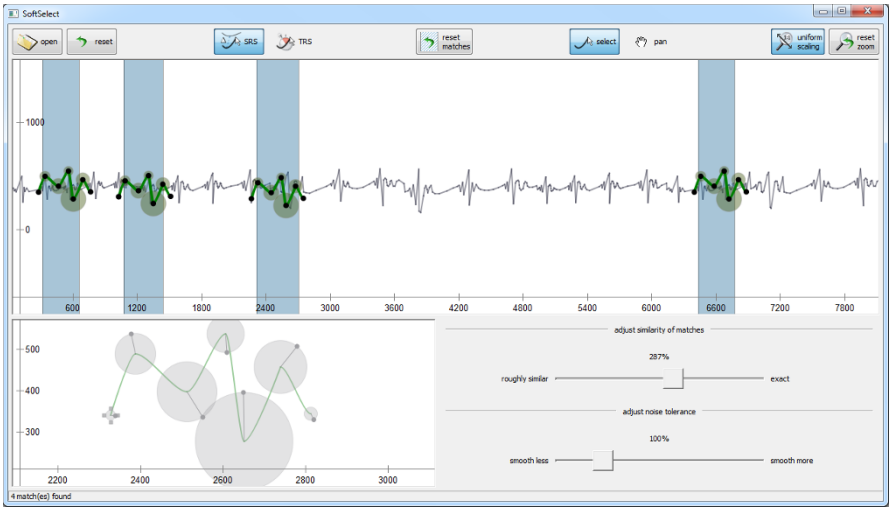


Fig. 5.11: The user can sketch a query pattern directly in the line plot and optionally refine it locally in a dedicated query view. The line plot then shows where in time the query matches with a certain tolerance. © *Courtesy of Christian Holz.*

queries are typically reserved for professional visualization systems. Again one can find a reason for that in the higher development costs for designing and implementing efficient interaction methods, particularly when direct manipulation and sketching are involved (see Mannino and Abouzied, 2018). Moreover, because visualization and interaction must be coupled tightly, it is typically difficult to develop interaction components that can be interchanged among the different visualization techniques for time-oriented data. One rare exception is the EazyPZ library (see Schwab et al., 2019b) whose zoom and pan functionality can be used as a basis for flexible navigation in time. Finding generally applicable solutions to other interaction problems is an open research question.

5.4 Advanced Interaction Methods

The previous section was concerned with basic interaction methods. In this section, we shed some light on advanced ways of interacting with time-oriented data. We start with interactive lenses as versatile tools for data exploration. When interesting data portions have been spotted, it is often necessary to compare them. This section will illustrate how visual comparison can be supported with naturally inspired interaction techniques. In order to help users make analytical progress, further advanced support can be offered in the form of guidance or by integrating automatic event-based methods. Finally, this section will consider advanced interaction using modern interaction modalities beyond mouse and keyboard interaction.

5.4.1 Interactive Lenses

Interactive lenses, originally introduced as magic lenses by Bier et al. (1993), are related to the focus+context concept discussed on p. 137. Tominski et al. (2017) define interactive lenses as lightweight tools that provide alternative visual representations for selected parts of the data on demand. Once activated, working with a lens is as easy as moving it across the visualization to specify where the lens is to take effect. The lens effect is automatically computed and merged with the base visualization to generate a locally enhanced visual representation. When the lens is no longer needed, it can simply be dismissed and the original visualization is restored.

As such, interactive lenses support scrutinizing the visualized data similar to using a magnifying glass. The difference though is that an interactive lens is not limited to enlarging selected parts of the visual representation. Conceptually, the effect generated by an interactive lens can include (i) the alternation of existing visualization content (e.g., change the coloring of selected time points), (ii) the omission of content (e.g., filter out less relevant data), or (iii) the addition of new content (e.g., add textual labels for clarification).

According to Tominski et al. (2017), more than 50 lens techniques for different data analysis scenarios are known in the literature, and eight of them are suited for time-oriented data. An additional example is the *regression lens* by Shao et al. (2017) shown in Figure 5.12. It is particularly useful for analyzing temporal trends. The lens' primary purpose is to enhance point plots (↪ p. 232) by adding locally computed regression curves for the data points within the perimeter of the lens. Our example shows two regression curves calculated by different algorithms. Additionally, the left and top borders of the lens are enhanced with histograms of the selected data. By

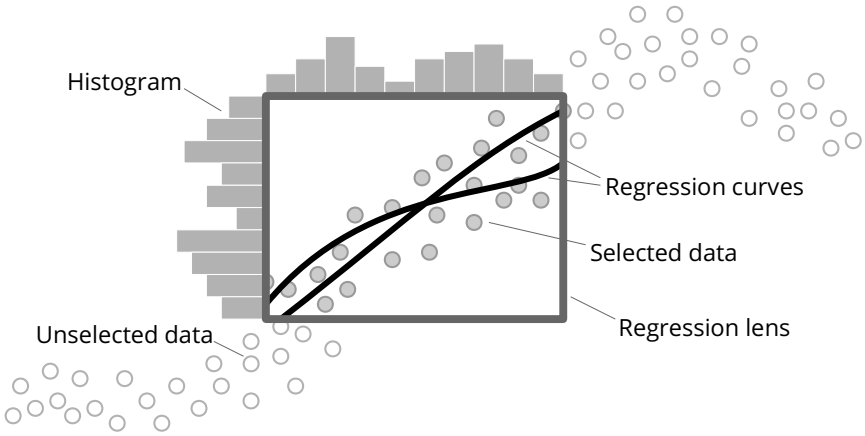


Fig. 5.12: The regression lens computes regression curves of its underlying data points and shows them as line plots on top of the base visualization. Additional histograms indicate the data distribution at the lens borders. *The authors. Adapted from Shao et al. (2017).*

moving and resizing the lens, the user can quickly explore the regression in local parts of the data without changing the original visualization globally.

While our example of the regression lens is focused on time-oriented data, interactive lenses are highly versatile tools in general. The swiftness and naturalness with which lenses can be operated are their key advantages. How natural interaction can also benefit the comparison of time-oriented data will be discussed next.

5.4.2 Interactive Visual Comparison

Comparing data is a ubiquitous data analysis activity (see Gleicher et al., 2011; Gleicher, 2018; L'Yi et al., 2021). It is particularly relevant in the context of time-oriented data. For example, the detection of temporal trends requires the comparison of individual data values along the time axis in the first place. Once promising trends have been identified, it is usually also of interest to compare them with each other: Which trend has the steeper slope or which trend peaks at the global maximum?

Without dedicated support, visual comparison can be a demanding task. In Chapter 4, we already discussed visual color-coding specifically to support visual comparison tasks. But still it may be necessary to move the eyes back and forth between the data to be compared, which is costly and error-prone. In the following, we discuss interaction techniques that allow users to dynamically re-arrange parts of a visual representation to facilitate visual comparison.

The interaction techniques to be presented are inspired by natural human behavior (see Tominski et al., 2012a). When people compare information printed on paper they usually carry out three steps:

1. Select comparison candidates
2. Arrange candidates for comparison
3. Carry out the actual comparison

In the first step, people specify *what* they want to compare. The comparison candidates can be individual data values or data items at different points in time or sub-ranges of the time axis showing interesting behavior such as trends or recurring patterns. In the second step, the comparison candidates are arranged so as to enable or ease their comparison. Finally, the actual comparison is conducted to figure out what relationships might exist between the compared data. Two requirements should be fulfilled in this regard. First, the properties of the individual data being compared should be clearly visible. Second, the similarities and differences between the data need to be communicated as well. The degree to which both requirements are met depends largely on the arrangement generated in step two, so let us look at this aspect in more detail.

Assume two comparison candidates A and B have been selected. When A and B are printed on paper, people would naturally arrange them as juxtaposition or superposition. For juxtaposition, A and B are arranged side by side. This allows us to see the individual data properties of A and B clearly, but in order to detect

(a) Side-by-side. **(b)** Shine-through. **(c)** Folding.

Fig. 5.13: Natural comparison behavior when comparing information printed on paper. *Tominski and Schumann (2020).*

similarities or differences, the eyes have to switch between both sides frequently. For superposition, A and B are stacked on top of each other. As A and B are now co-located, similarities and differences are potentially easier to see, but either A occludes B or the other way around, which hinders the comparison and also deteriorates the visibility of either A or B. For real-world comparison on paper, the occlusion can be resolved in two ways. Either the stacked A and B are held against the light to let the occluded information shine through and generate a merged representation of A and B. Or the occlusion is resolved by folding the occluding piece of paper back and forth to reveal A and B in quick succession. Figure 5.13 illustrates these natural comparison behaviors: side-by-side, shine-through, and folding.

On the computer, this natural comparison between A and B can be replicated via advanced interaction techniques, as schematically depicted in Figure 5.14. Via simple drag gestures, side-by-side and overlapping arrangements can be created. For resolving occlusions, shine-through comparison can be implemented via alpha-blending, where the occluding view is made partially transparent. The folding technique makes it possible to peel off the occluding view very much like for real paper. To keep the interaction costs low, the folding can simply be triggered by clicking at the location where the occlusion between the views is to be resolved. Based on a heuristic, a natural fold is calculated and presented via a smooth animation.

Let us take a closer look at Figure 5.14 to understand the advantages and drawbacks of the different interactions. In the side-by-side variant, the user drags comparison candidate B next to A. This shows both subsets of the data clearly, however, determining which trend is steeper might not be so easy to figure out. The shine-through technique makes the direct comparison of the trends easier by superimposing A and B and allowing the user to manipulate the degree of occlusion via a vertical drag gesture or slider. Yet it is no longer clear which line plot belongs to which subset. The folding variant is a compromise. It clearly separates the superimposed line plots, and by quickly folding back and forth, the peaks can be compared reasonably well. Yet, the collateral occlusion caused by the folding need to be dealt with.

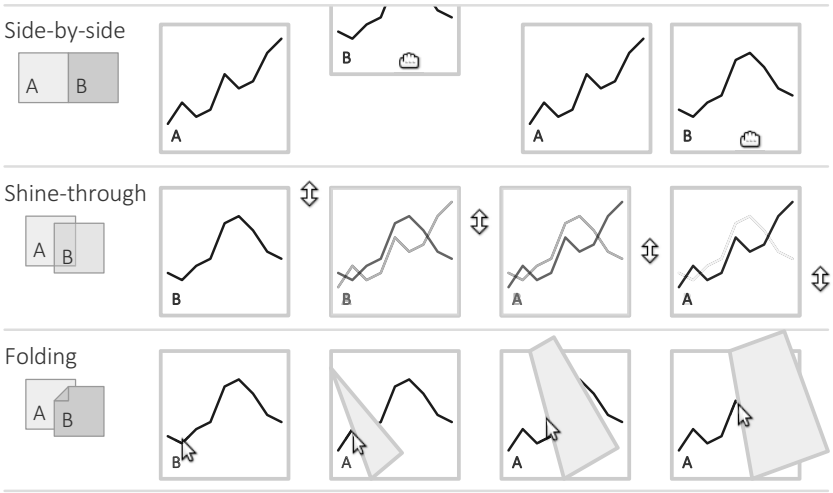


Fig. 5.14: Side-by-side, shine-through, and folding interaction. *The authors.*

In summary, this section illustrated different interaction techniques for supporting visual comparison tasks, which are so common when analyzing time-oriented data. The naturalness of the interactions makes them easy to learn and carry out. Moreover, the outlined techniques are not limited to comparing line plots, but are generally applicable to any visual representation.

5.4.3 Guiding the User

The interaction techniques described in this chapter so far provide many degrees of freedom to enable users to study time-oriented data from different perspectives and to develop a comprehensive understanding. However, the many degrees of freedom can also be a challenge. During the data exploration, many questions arise: Where should I move the lens to identify a local cluster? Which partial trends should I select for comparison? Where should I navigate to find interesting patterns? These questions become problematic when there are too many of them and when the user has too many difficulties answering them. If this is the case, the analytical progress stalls and the interactive exploration comes to a halt.

To ensure steady progress and to keep the data exploration going, it makes sense to provide users with guidance. *Guidance* has been defined as a means to help users resolve problems they may encounter during interactive data exploration (see Schulz et al., 2013b; Ceneda et al., 2017; Collins et al., 2018). The important aspect here is that guidance is there to help and to assist. It is not a means to provide answers to analytic questions, but to enable and support users to arrive at answers on their own, that is, the human remains in the loop.

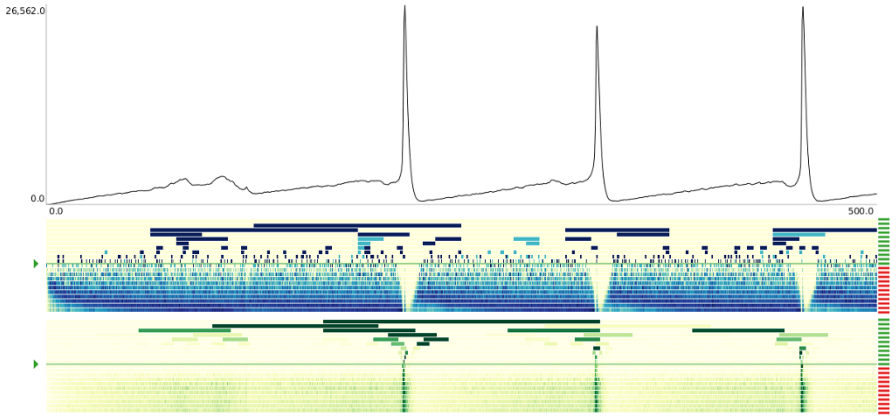


Fig. 5.15: Overview plot of a time series with 3.6 million time points (top) and color-coded difference bands (center: slope sign difference; bottom: absolute value difference) indicating where potentially interesting observations could be made. CC BY *Martin Luboschik. Also see Luboschik et al. (2012).*

In the following, we will demonstrate how large time-oriented data can be explored at multiple scales with the help of an appropriate guidance strategy. The starting point is a large time series with millions of data points from a simulation of the cell division cycle in fission yeast (see Luboschik et al., 2012). We are going to visualize these data as a classic line plot (↪ p. 233). The problem though is that about 3.6 million time points usually do not fit in a line plot. Therefore, the time series has been aggregated at several levels of granularity, leading to a multi-scale representation of the data. Such a representation lends itself to being explored via zooming. When the zoom level changes, the visualization shows the level of granularity that matches the resolution of the display.

An overview of the whole time series is depicted at the top of Figure 5.15. At this level of granularity, one can easily see three peaks. But what we are seeing is only a coarse representation, in fact, the coarsest of our multi-scale time series. We do not know what is going on at the finer scales on the slopes or at the top of the peaks. Zooming and panning will allow us to access the details we seek. However, where in time and at what temporal scale can we make interesting observations? The guidance approach we are about to demonstrate uses the data themselves as an input to compute visual cues that provide users with orientation to narrow down their search on promising parts of the data.

The assumption is that differences between adjacent scales might serve as an indication for users to look more closely into particular parts of the data. Various measures can be employed to calculate the differences. Luboschik et al. (2012) consider absolute value differences and slope sign differences. These measures are calculated for all pairs of adjacent scales. Aggregating the measures and color-coding them leads to so-called difference bands that can be attached below our line plot on demand as shown in Figure 5.15.

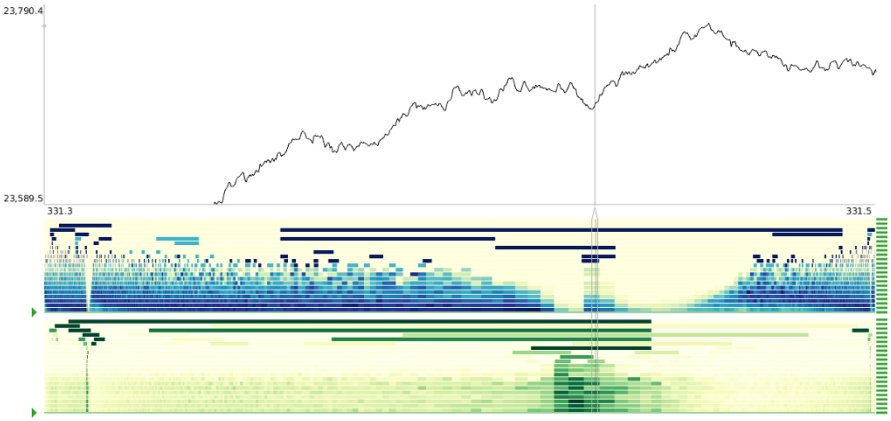


Fig. 5.16: Zoomed view of the tip of the second peak from Figure 5.15. The difference bands are magnified by means of a focus+context distortion. *Martin Luboschik. Also see Luboschik et al. (2012).*

Interestingly, in the bluish bands for slope sign difference (center), we can see three notches exactly where the three peaks are in the line plot. There are also three greenish spikes in the absolute value difference bands (bottom). So, both bands *guide* the user to the peaks for more detailed inspection. And in fact, some interesting behavior can be observed. Looking at the notches for slope sign difference in Figure 5.15 more closely, one can see thin spikes.

To understand what is going on, we study the second notch in more detail. We magnify the second notch and the tip of its associated peak as shown in Figure 5.16. From the magnified difference bands, we can see that greater differences, indicated by darker colors, exist between the temporal scales of finer granularity. The zoomed line plot confirms that the tip of the peak is not a smooth curve as we might have thought. There is in fact a rather rough up and down of the curve.

This example of multi-scale exploration of time-oriented data illustrates the benefit of providing guidance. The additional difference bands provide on-demand support to help users decide which parts of the data are promising to study in detail. Other examples of guidance exist, where the focus is less on navigation, but on guiding the configuration of visualization techniques, for example, to suggest suitable cycle lengths of spiral representations (↪ p. 274) to help users find cyclic patterns in time-oriented data (see Ceneda et al., 2018). For a broader view on guidance and more examples, the interested reader is referred to the survey by Ceneda et al. (2019).

5.4.4 Integrating Interaction and Automation via Events

With the increasing complexity of data and visualization methods alike, it is not always easy for users to set visualization parameters appropriately for the analysis task at hand. Particularly if parameters are not self-explanatory, they are not easily set manually. Guidance can provide a form of support to assist users in the parametrization process.

Another possible solution is to employ the concept of *event-based visualization*, which combines visualization with event methodology (see Reinders et al., 2001; Tominski, 2011). In diverse application fields, including active databases, software engineering, and modeling and simulation, events are considered happenings of interest that trigger some automatic actions. In the context of visualization, such an event-action-scheme is useful for complementing manual interaction with automatic parametrization of visual representations.

The basic idea of event-based visualization is (1) to let users specify their interests, (2) to detect if and where these interests match in the data, and (3) to consider detected matches when generating the visual representation. This general procedure requires three main components: (1) *event specification*, (2) *event detection*, and (3) *event representation*. Figure 5.17 illustrates how they are attached to the visualization pipeline. Next, we will look at each of these components in more detail.

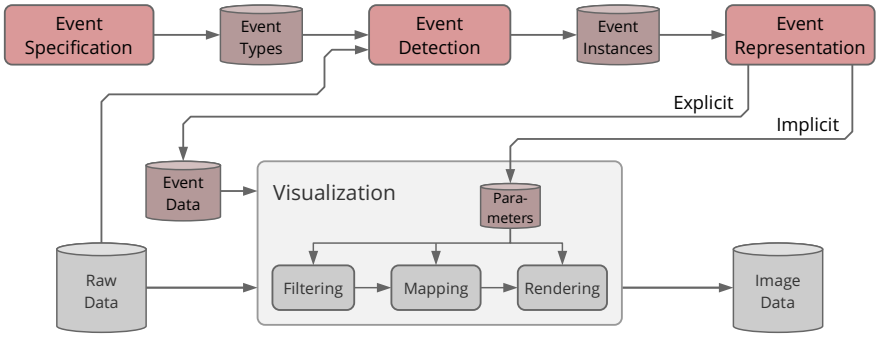


Fig. 5.17: The main ingredients of event-based visualization – event specification, event detection, and event representation – attached to the visualization pipeline. *The authors.*

Describing User Interests

The event specification is an interactive step where users describe their interests as *event types*. To be able to find actual matches of user interests in the data, the event specification must be based on formal descriptions. Tominski (2011) uses elements of predicate logic to create well-defined event formulas that express interests with

respect to relational datasets (e.g., data records whose values exceed a threshold or attribute with the highest average value). For an analysis of time-oriented data, sequence-related notations (for instance as introduced by Sadri et al. (2004)) enable users to specify conditions of interest regarding temporally ordered sequences (e.g., sequence of days with rising stock prices). A combination of event types to composite event types is possible via set operators.

As a simple example, we formulate our interest in *"Three successive days where the number of people diagnosed with influenza increases by more than 15% each day"* as the following event type:

$$\{(x, y, z)_{date} \mid z.flu \geq y.flu \cdot 1.15 \wedge y.flu \geq x.flu \cdot 1.15\}$$

The first part of the formula defines three variables $(x, y, z)_{date}$ that are sequenced by date. To express the condition of interest, these three variables are set into relation using predicates, functions, and logical connectors.

Certainly, casual users may find it difficult to describe their interests via event formulas. Therefore, sufficient specification support should consider dedicated means for experts, regular users, and visualization novices. In this regard, one can think of three levels of specification: *(i) direct specification*, *(ii) specification by parametrization*, and *(iii) specification by selection*. All levels are based on the aforementioned formalism, but the complete functionality is available only to expert users at the level of direct specification. The second level works with parametrizable templates that hide the complexity of event formulas from the user. Non-expert users can adjust the templates via easy-to-set parameters, but otherwise do not need to fiddle with the internals of event formulas. For example, exposing the increase rate (15% in our previous example) as a template parameter would be reasonable. At the third level, users simply select from a predefined collection of event types that are particularly tailored to the application context.

Finding Relevant Data Portions

The event detection is an automatic step that determines whether the interests defined interactively are present in the data. The outcome of the event detection is a set of *event instances*. They describe where in the data interesting information is located. That is, entities that match user interests are marked as event instances. For event detection, the variables used in event formulas are substituted with concrete data entities. In the second step, predicates, functions, and logical connections are evaluated, so that the event formula as a whole can be evaluated as either true or false. Because this procedure can be quite costly in terms of computation time, efficient methods must be utilized for the event detection. A combination of the capabilities of relational database management systems and efficient algorithms (e.g., the OPS algorithm by Sadri et al. (2004)) is useful for static data. When dynamic data (i.e., data that change over time, see Section 3.3) have to be considered, detection efficiency becomes even more crucial. Here, incremental detection methods can help. Such

methods operate on a differential dataset, rather than on the whole data. However, incremental methods also impose restrictions on possible event types, because they do not have access to the entire dataset.

Considering User Interests in Visual Representations

The last important step of event-based visualization is the event representation. The goal of this step is to incorporate detected event instances, which reflect the interests of the user, into visual representations. The three requirements that have to be considered are as follows:

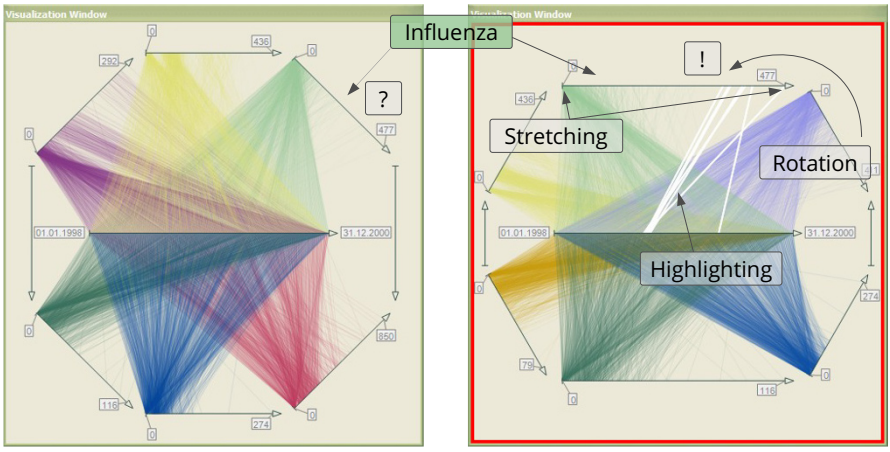
1. Communicate the fact that something interesting has been found.
2. Emphasize interesting data among the rest of the data.
3. Convey what makes the data interesting.

Most importantly, the visual representation must clearly express that something interesting is contained in the data. To meet this requirement, easy-to-perceive visual cues (e.g., a red frame around the visual representation, exclamation marks, or annotations) can be used. Alpha-blending can be applied to fade out past events. The second requirement aims at emphasizing those parts of the visual representation that are of interest. Additionally, the visualization should communicate what makes the highlighted parts interesting (i.e., what the particular event type is). However, when facing arbitrarily definable event formulas, this last requirement is difficult to fulfill.

We can distinguish two basic options for representing events: *explicit* and *implicit* event representation. For the explicit case, the focus is set exclusively on event instances, neglecting the raw data. Since the number of events is usually smaller than the number of data items, explicit event representation can grant insight even into very large datasets. For implicit event representation, the goal is to automatically adjust visualization parameters so as to highlight the points of interest detected in the data. Assuming that user interests relate to user tasks and vice versa, implicit event representation can help us obtain better-targeted visual representations. The big challenge though is to meet the aforesaid requirements solely by adapting visualization parameters. Apparently, the availability of adequate visualization parameters is a prerequisite for implicit event representation.

Let us illustrate the potential of event-based visualization with an example. Assume a user has to analyze multivariate time-dependent human health data for uncommonly high numbers of cases of influenza. The task at hand is to find out if and where in time these situations have occurred. A possible way to accomplish this task is to use the TimeWheel technique (↪ p. 298).

Figure 5.18a shows a TimeWheel that uses the standard parametrization, where time is encoded along the central axis and multiple diagnoses are mapped to the axes surrounding the time axis. In particular, influenza happens to be the diagnosis that is mapped to the upper right axis (light green). Alpha-blending is applied by default to reduce visual clutter. Looking at this TimeWheel, the user can only guess from the labels of the axis showing influenza that there are higher numbers of cases



(a) Default parametrization. **(b)** Targeted parametrization.

Fig. 5.18: Default vs. targeted parametrization of a TimeWheel. (a) TimeWheel representing a time-dependent health dataset using the default configuration, which aims at showing main trends, but does not consider the interests of the user. (b) TimeWheel representing the same data, but matches with the user's interests have been detected and corresponding data are emphasized via highlighted lines and automatic rotation and stretching; the presentation is better targeted to the user's task at hand.

because the alpha-blending made the particular lines almost invisible (see question mark). Several interaction steps are necessary to re-parametrize the TimeWheel to accomplish the task at hand.

In contrast to this, in an event-based visualization environment, the user can specify the interest in *"Days with a high number of cases of influenza"* as the event type ($\{x \mid x.flu \geq 300\}$). If the event detection step confirms the existence of such events in the data, visualization parameters are altered automatically so as to provide an individually adjusted TimeWheel that reflects the special situation. In our particular example in Figure 5.18b, we change the color and transparency of line segments representing event instances: Days with high numbers of influenza cases are excluded from alpha-blending and are drawn in white (see exclamation mark). Additionally, rotation and stretching are applied such that the axis representing influenza is moved gradually to an exposed position and is provided with more display space. The application of a gradual process is important in this case to support users in maintaining their mental map of the visual representation. In this automatically adjusted TimeWheel, the identification of days with higher numbers of influenza infections is easy.

5.4.5 Interaction Beyond Mouse and Keyboard

Most of the interaction techniques discussed in this chapter, and also most of the techniques described in the literature, are designed for the classic desktop computer workplace where the mouse and keyboard are the dominant input devices. Yet, technological advances have brought us to a point where new interaction modalities are becoming more and more commonplace. Interaction beyond mouse and keyboard brings new possibilities for exploring and analyzing data in various ways (see Lee et al., 2012; Keefe and Isenberg, 2013). In this section, we briefly look at what is possible in terms of modern interaction for time-oriented data. In particular, we consider touch interaction for exploring time-oriented data visualized as stacked graphs and tangible interaction for exploring space-time cube visualizations.

Touching Stacked Graphs

Touch interaction has become the primary input modality for mobile devices. It can also be found on laptop computers and larger display surfaces (see Voida et al., 2009). Touch interaction has the advantage that the action takes place directly on the display, exactly where the operation is to take effect. Yet, a difficulty with touch is that the input devices, our fingers, are rather imprecise making it harder to point at fine details in a visualization. Using the fingers for interaction can also cause the hand to occlude relevant information on the display. Nonetheless, the directness and intuitiveness of touch interaction are the key motivation for using it in the context of visualization.

The example we are looking at here is TouchWave by Baur et al. (2012). TouchWave is specifically designed for direct and fluid interaction with time-oriented data visualized as stacked graphs (↪ p. 286). For improving the legibility, comparability, and scalability of stacked graphs, several concrete touch interactions and corresponding visual feedback are offered. Legibility can be improved by touching the visualization background, which triggers the display of an on-demand vertical ruler showing the exact value distribution for the time point corresponding to the finger position. By using more than one finger, which is called multi-touch interaction, additional rulers can be activated to facilitate the visual comparison of several points in time.

As the order of individual streams in a stacked graph is important, reordering the streams is an essential operation. By long-pressing the stacked graph, its streams can be sorted so that the stream with the highest value for the time point being touched is at the top. Double tapping a stream will make it the baseline stream on top of which all other streams are stacked. Moreover, individual streams can be pulled out of the stacked graph via simple drag gestures. These interactive rearrangements are particularly useful for comparison, as we have already seen in Section 5.4.2.

To support multi-scale data exploration, the TouchWave utilizes pinch gestures. Pinching horizontally will create a focus+context distortion of the time line revealing details in the focus, while compressing the context. Vertical pinching can be used to

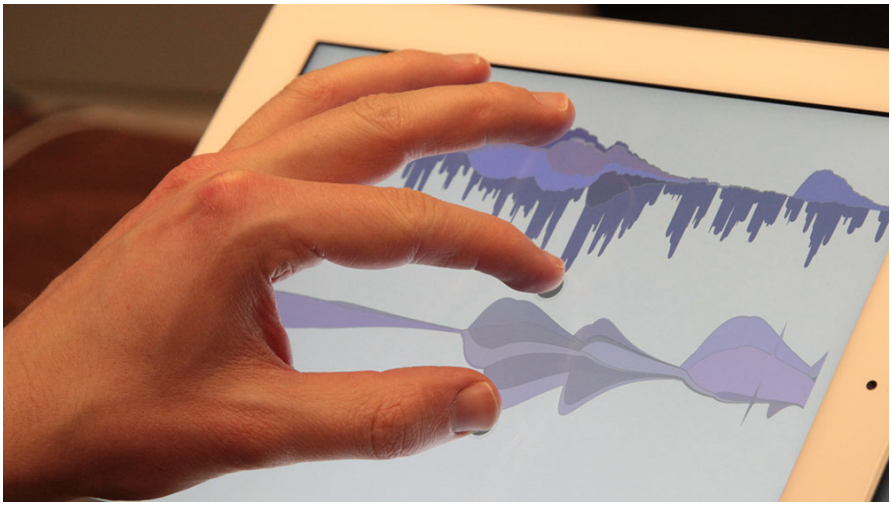


Fig. 5.19: Using a pinch gesture for scaling a stacked graph visualization vertically. © *Courtesy of Dominikus Baur.* https://do.minik.us/projects/touchwave

perform a hierarchical zoom with respect to the streams in a stacked graph. Such a vertical pinch gesture is illustrated in Figure 5.19.

TouchWave is designed particularly for stacked graphs. Yet, touch-based interaction also works for other visualizations of time-oriented data. For example, Riehmann et al. (2018) describe dedicated touch interactions for multiple time series depicted as horizon graphs (↪ p. 277). What all touch techniques have in common is that they facilitate the direct interaction *on* the display. Next, we will see how tangible interaction can support interaction *with* the display.

Exploring Space-Time Cubes with Tangible Interaction

Tangible interaction is a style of interaction where users interact by manipulating physical objects, so-called *tangibles* (see Shaer and Hornecker, 2009). This requires appropriate tracking equipment so that the system knows where the tangibles are located and how they are oriented in space. The spatial awareness can be utilized to define whole new interaction vocabularies. Basic interactions include horizontal and vertical translation and rotation, which in turn can be combined to gestures such as tilting, flipping, or shaking a tangible. These interactions can then be utilized to design new data exploration experiences.

In the context of exploring time-oriented data, tangible interaction opens up new possibilities for navigating the time axis and also for adjusting the visual representation depending on the user's tasks. To illustrate the usefulness of tangible interaction, we present two examples: tangible views and the Uplift system. In both cases, spatio-temporal data are visualized as a space-time cube (↪ p. 377) on a horizontal tabletop

display. The cube's base plane resides in the horizontal x-y plane of the tabletop and the dimension of time extends from the base plane along the vertical z-axis. It is important to realize that the space-time cube is a virtual one, meaning that the space above the horizontal tabletop defines the space-time cube, but its content is not yet visible. Initially, there is only a map on the tabletop, but via tangible interaction, one can access the space-time cube and make different parts of time and space visible.

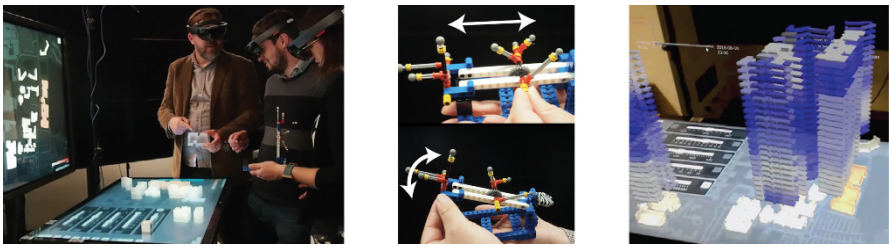
Tangible views The two terms *tangible* and *views* already hint at a duality between display and interaction: The views serve to show the visualization, and at the same time, the views are tangible and serve as an input device for interacting with the visualization. Conceptually, tangible views are spatially-aware lightweight displays. Spindler et al. (2010) describe an implementation where tangible views are made of cardboard onto which visual representations can be projected.

(a) Flipping the color-coding. **(b)** Side-by-side comparison.

Fig. 5.20: Using tangible views for exploring spatio-temporal data in a virtual space-time cube. *The authors.*

In order to interactively explore a virtual space-time cube and adjust its visualization, one or more tangible views can be held in the space above the base map as illustrated in Figure 5.20. Different parts of the map can be accessed by moving a tangible view horizontally (i.e., navigation in space). The tangible view's partial map is then updated according to the horizontal position above the base map. Similarly, by raising and lowering the tangible view along the vertical axis, one can select particular time points to be displayed (i.e., navigation in time). By flipping the tangible view, it is possible to switch between two different color-coding strategies, for example, for identification and location tasks as described in Section 4.2.2. Tangible views can also facilitate visual comparison. To this end, two tangible views are used in combination. First, each view is moved individually to select two map regions and two time points to be compared. Then a lock operation is performed, which makes both tangible views insensitive to further motion. This in turn allows the user to bring the two tangible views together forming a side-by-side arrangement for comparison.

Uplift Our second example of tangible interaction with a space-time cube visualization is the Uplift system by Ens et al. (2021). In this example, the space-time cube is also located in the space above a tabletop display, but it is displayed virtually as an augmented-reality representation. This allows several persons to look at the data simultaneously as shown in Figure 5.21. Several tangibles are used in concert to interact with the system in various ways. Particularly interesting is the navigation through time and the unfolding of the space-time cube. By placing a tangible token on the tabletop, slider widgets with different temporal granularity can be activated. A physical slider widget can then be used to select a particular point in time. By using a hinge of the physical widget, the space-time cube can be unfolded to show several data layers for comparing multiple time steps.



(a) (b) (c)

Fig. 5.21: Uplift: tangible and immersive tabletop system. (a) Collaborative exploration around a tabletop display using tangible objects. (b) Physical widget for navigating in time. (c) Unfolded space-time cube visualization above the tabletop surface.

What we can learn from tangible views, physical widgets, and TouchWave before is that there is more to interaction than just mouse and keyboard. Touch and tangible interaction are but two examples of modern ways of interacting with data. Further examples are gaze-based interaction (see Duchowski, 2018), where the eyes perform actions, and proxemic interaction (see Jakobsen et al., 2013), where the distance of the user to the display is considered. Natural language is another channel to be utilized for interaction, where combining language with other input modalities seems to be a quite promising approach (see Srinivasan and Stasko, 2018). Yet, further research needs to be conducted to take full advantage of these new interaction modalities and their combination for the particular case of visually exploring and analyzing time-oriented data.

5.5 Summary

The focus of this chapter was on interaction. We started with a brief overview of intents that motivate users to interact with the visualization. The most notable intent

in the context of time-oriented data is the intent to navigate in time in order to visit different parts of the data. Users also need to view time-oriented data at different levels of detail, because the data are often given at multiple granularities. Further intents are related to interactively adjusting the visual mapping according to data and tasks at hand, and to managing the exploration process.

We explained that interactive visualization is an iterative loop where the user plans and carries out an interaction, and the computer generates feedback in order to visually reflect the change that resulted from the user's actions. This human-in-the-loop process brings together the computational power of the machine and the intellectual power of human beings. In order to take full advantage of this synergy, we need an efficient user interface that bridges the gap between the algorithmic structures being used for visualizing time and time-oriented data, and the mental models and analytic workflows of users. This also includes tackling technical challenges to guarantee the smooth execution of the interaction loop.

This chapter also presented basic interaction concepts, including temporal navigation, direct manipulation, brushing & linking, and dynamic queries. These concepts are vital for data exploration tasks where the user performs an undirected search for potentially interesting data features. Going beyond basic interaction, we considered interactive lenses, natural visual comparison, guidance, event-based visualization, and interaction beyond mouse and keyboard. These advanced concepts can further enhance the visual exploration of time-oriented data. But still, the potential of advanced interaction methods has not been fully exploited by current visualization techniques. There is room for future work to better adapt existing interaction methods or to develop new ones according to the specific needs of time-oriented data. Moreover, the examples of guidance and event-based visualization indicate that a combination of visual, interactive, and automatic methods can be quite useful. In the next chapter, we will take a closer look at computational analysis methods for supporting the visual analysis of time-oriented data.

References

Angelini, M., G. Santucci, H. Schumann, and H.-J. Schulz (2018). "A Review and Characterization of Progressive Visual Analytics". In: *Informatics* 5.3, p. 31. DOI: 10.3390/informatics5030031.

Baldonado, M. Q. W., A. Woodruff, and A. Kuchinsky (2000). "Guidelines for Using Multiple Views in Information Visualization". In: *Proceedings of the Conference on Advanced Visual Interfaces (AVI)*. ACM Press, pp. 110–119. DOI: 10.1145/345513.345271.

Baur, D., B. Lee, and S. Carpendale (2012). "TouchWave: Kinetic Multi-touch Manipulation for Hierarchical Stacked Graphs". In: *Proceedings of the International Conference on Interactive Tabletops and Surfaces (ITS)*. ACM Press, pp. 255–264. DOI: 10.1145/2396636.2396675.

Becker, R. A. and W. S. Cleveland (1987). "Brushing Scatterplots". In: *Technometrics* 29.2, pp. 127–142. DOI: 10.2307/1269768.

Bertin, J. (1981). *Graphics and Graphic Information-Processing*. Translated by William J. Berg and Paul Scott. de Gruyter.

Bier, E. A., M. C. Stone, K. Pier, W. Buxton, and T. D. DeRose (1993). "Toolglass and Magic Lenses: The See-Through Interface". In: *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*. ACM Press, pp. 73–80. DOI: 10.1145/166117.166126.

Ceneda, D., T. Gschwandtner, T. May, S. Miksch, H.-J. Schulz, M. Streit, and C. Tominski (2017). "Characterizing Guidance in Visual Analytics". In: *IEEE Transactions on Visualization and Computer Graphics* 23.1, pp. 111–120. DOI: 10.1109/TVCG.2016.2598468.

Ceneda, D., T. Gschwandtner, and S. Miksch (2019). "A Review of Guidance Approaches in Visual Data Analysis: A Multifocal Perspective". In: *Computer Graphics Forum* 38.3, pp. 861–879. DOI: 10.1111/cgf.13730.

Ceneda, D., T. Gschwandtner, S. Miksch, and C. Tominski (2018). "Guided Visual Exploration of Cyclical Patterns in Time-series". In: *Proceedings of the IEEE Symposium on Visualization in Data Science (VDS)*. IEEE Computer Society.

Chen, H. (2004). "Compound Brushing Explained". In: *Information Visualization* 3.2, pp. 96–108. DOI: 10.1057/palgrave.ivs.9500068.

Cheng, X., D. Cook, and H. Hofmann (2016). "Enabling Interactivity on Displays of Multivariate Time Series and Longitudinal Data". In: *Journal of Computational and Graphical Statistics* 25.4, pp. 1057–1076. DOI: 10.1080/10618600.2015.1105749.

Cockburn, A., A. Karlson, and B. B. Bederson (2009). "A Review of Overview+Detail, Zooming, and Focus+Context Interfaces". In: *ACM Computing Surveys* 41.1, 2:1–2:31. DOI: 10.1145/1456650.1456652.

Collins, C., N. Andrienko, T. Schreck, J. Yang, J. Choo, U. Engelke, A. Jena, and T. Dwyer (2018). "Guidance in the Human-Machine Analytics Process". In: *Visual Informatics* 2.3. DOI: 10.1016/j.visinf.2018.09.003.

Cooper, A., R. Reimann, and D. Cronin (2007). *About Face 3: The Essentials of Interaction Design*. Wiley Publishing, Inc.

Doleisch, H. and H. Hauser (2002). "Smooth Brushing for Focus+Context Visualization of Simulation Data in 3D". In: *Proceedings of the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*. University of West Bohemia, pp. 147–154. URL: http://wscg.zcu.cz/wscg2002/Papers_2002/E71.pdf.

Duchowski, A. T. (2018). "Gaze-based Interaction: A 30 Year Retrospective". In: *Computers & Graphics* 73, pp. 59–69. DOI: 10.1016/j.cag.2018.04.002.

Ens, B., S. Goodwin, A. Prouzeau, F. Anderson, F. Y. Wang, S. Gratzl, Z. Lucarelli, B. Moyle, J. Smiley, and T. Dwyer (2021). "Uplift: A Tangible and Immersive Tabletop System for Casual Collaborative Visual Analytics". In: *IEEE Transactions on Visualization and Computer Graphics* 27.2, pp. 1193–1203. DOI: 10.1109/TVCG.2020.3030334.

Gajos, K. Z., M. Czerwinski, D. S. Tan, and D. S. Weld (2006). "Exploring the Design Space for Adaptive Graphical User Interfaces". In: *Proceedings of the Conference on Advanced Visual Interfaces (AVI)*. ACM Press, pp. 201–208. DOI: 10.1145/1133265.1133306.

Gleicher, M. (2018). "Considerations for Visualizing Comparison". In: *IEEE Transactions on Visualization and Computer Graphics* 24.1, pp. 413–423. DOI: 10.1109/TVCG.2017.2744199.

Gleicher, M., D. Albers, R. Walker, I. Jusufi, C. D. Hansen, and J. C. Roberts (2011). "Visual Comparison for Information Visualization". In: *Information Visualization* 10.4, pp. 289–309. DOI: 10.1177/1473871611416549.

Guo, D., J. Chen, A. M. MacEachren, and K. Liao (2006). "A Visualization System for Space-Time and Multivariate Patterns (VIS-STAMP)". In: *IEEE Transactions on Visualization and Computer Graphics* 12.6, pp. 1461–1474. DOI: 10.1109/TVCG.2006.84.

Hauser, H., F. Ledermann, and H. Doleisch (2002). "Angular Brushing of Extended Parallel Coordinates". In: *Proceedings of the IEEE Symposium Information Visualization (InfoVis)*. IEEE Computer Society, pp. 127–130. DOI: 10.1109/INFVIS.2002.1173157.

Heer, J. and G. Robertson (2007). "Animated Transitions in Statistical Data Graphics". In: *IEEE Transactions on Visualization and Computer Graphics* 13.6, pp. 1240–1247. DOI: 10.1109/tvcg.2007.70539.

Henriksen, K., J. Sporring, and K. Hornbæk (2004). "Virtual Trackballs Revisited". In: *IEEE Transactions on Visualization and Computer Graphics* 10.2, pp. 206–216. DOI: 10.1109/tvcg.2004.1260772.

Hochheiser, H. and B. Shneiderman (2004). "Dynamic Query Tools for Time Series Data Sets: Timebox Widgets for Interactive Exploration". In: *Information Visualization* 3.1, pp. 1–18. DOI: 10.1057/palgrave.ivs.9500061.

Hoffswell, J., W. Li, and Z. Liu (2020). "Techniques for Flexible Responsive Visualization Design". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*. ACM Press, pp. 1–13. DOI: 10.1145/3313831.3376777.

Holz, C. and S. Feiner (2009). "Relaxed Selection Techniques for Querying Time-Series Graphs". In: *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*. ACM Press, pp. 213–222. DOI: 10.1145/1622176.1622217.

Jakobsen, M. R., Y. S. Haile, S. Knudsen, and K. Hornbæk (2013). "Information Visualization and Proxemics: Design Opportunities and Empirical Findings". In: *IEEE Transactions on Visualization and Computer Graphics* 19.12, pp. 2386–2395. DOI: 10.1109/TVCG.2013.166.

Jankun-Kelly, T. J., K.-L. Ma, and M. Gertz (2007). "A Model and Framework for Visualization Exploration". In: *IEEE Transactions on Visualization and Computer Graphics* 13.2, pp. 357–369. DOI: 10.1109/tvcg.2007.28.

Keefe, D. F. and T. Isenberg (2013). "Reimagining the Scientific Visualization Interaction Paradigm". In: *Computer* 46.5, pp. 51–57. DOI: 10.1109/MC.2013.178.

Kondo, B. and C. Collins (2014). "DimpVis: Exploring Time-varying Information Visualizations by Direct Manipulation". In: *IEEE Transactions on Visualization and Computer Graphics* 20.12, pp. 2003–2012. DOI: 10.1109/TVCG.2014.2346250.

Krasner, G. E. and S. T. Pope (1988). "A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80". In: *Journal of Object-Oriented Programming* 1.3, pp. 26–49.

L'Yi, S., J. Jo, and J. Seo (2021). "Comparative Layouts Revisited: Design Space, Guidelines, and Future Directions". In: *IEEE Transactions on Visualization and Computer Graphics* 27.2, pp. 1525–1535. DOI: 10.1109/TVCG.2020.3030419.

Lam, H. (2008). "A Framework of Interaction Costs in Information Visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 14.6, pp. 1149–1156. DOI: 10.1109/TVCG.2008.109.

Lee, B., P. Isenberg, N. H. Riche, and S. Carpendale (2012). "Beyond Mouse and Keyboard: Expanding Design Considerations for Information Visualization Interactions". In: *IEEE Transactions on Visualization and Computer Graphics* 18.12, pp. 2689–2698. DOI: 10.1109/TVCG.2012.204.

Liu, Z. and J. Heer (2014). "The Effects of Interactive Latency on Exploratory Visual Analysis". In: *IEEE Transactions on Visualization and Computer Graphics* 20.12, pp. 2122–2131. DOI: 10.1109/TVCG.2014.2346452.

Luboschik, M., C. Maus, H.-J. Schulz, H. Schumann, and A. Uhrmacher (2012). "Heterogeneity-Based Guidance for Exploring Multiscale Data in Systems Biology". In: *Proceedings of the IEEE Symposium on Biological Data Visualization (BioVis)*. IEEE Computer Society, pp. 33–40. DOI: 10.1109/BioVis.2012.6378590.

Mannino, M. and A. Abouzied (2018). "Expressive Time Series Querying with Hand-Drawn Scale-Free Sketches". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*. ACM Press. DOI: 10.1145/3173574.3173962.

Monroe, M., R. Lan, J. M. del Olmo, B. Shneiderman, C. Plaisant, and J. Millstein (2013a). "The Challenges of Specifying Intervals and Absences in Temporal Queries: A Graphical Language Approach". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*. ACM Press, pp. 2349–2358. DOI: 10.1145/2470654.2481325.

Norman, D. A. (2013). *The Design of Everyday Things*. Revised and expanded edition. Basic Books.

Piringer, H., C. Tominski, P. Muigg, and W. Berger (2009). "A Multi-Threading Architecture to Support Interactive Visual Exploration". In: *IEEE Transactions on Visualization and Computer Graphics* 15.6, pp. 1113–1120. DOI: 10.1109/TVCG.2009.110.

Pulo, K. (2007). "Navani: Navigating Large-Scale Visualisations with Animated Transitions". In: *Proceedings of the International Conference Information Visualisation (IV)*. IEEE Computer Society, pp. 271–276. DOI: 10.1109/iv.2007.82.

Reinders, F., F. H. Post, and H. J. W. Spoelder (2001). "Visualization of Time-Dependent Data with Feature Tracking and Event Detection". In: *The Visual Computer* 17.1, pp. 55–71. DOI: 10.1007/pl00013399.

Riehmann, P., J. Reibert, J. Opolka, and B. Fröhlich (2018). "Touch the Time: Touch-Centered Interaction Paradigms for Time-Oriented Data". In: *Proceedings of the Eurographics / IEEE Conference on Visualization (EuroVis) - Short Papers*. Eurographics Association, pp. 113–117. DOI: 10.2312/eurovisshort.20181088.

Sadri, R., C. Zaniolo, A. Zarkesh, and J. Adibi (2004). "Expressing and Optimizing Sequence Queries in Database Systems". In: *ACM Transactions on Database Systems* 29.2, pp. 282–318. DOI: 10.1145/1005566.1005568.

Schulz, H.-J., M. Streit, T. May, and C. Tominski (2013b). *Towards a Characterization of Guidance in Visualization*. Poster at IEEE Conference on Information Visualization (InfoVis). Atlanta, USA.

Schwab, M., S. Hao, O. Vitek, J. Tompkin, J. Huang, and M. A. Borkin (2019a). "Evaluating Pan and Zoom Timelines and Sliders". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*. ACM Press, pp. 1–12. DOI: 10.1145/3290605.3300786.

Schwab, M., J. Tompkin, J. Huang, and M. A. Borkin (2019b). "EasyPZ.js: Interaction Binding for Pan and Zoom Visualizations". In: *IEEE Visualization Conference, IEEE VIS 2019 - Short Papers*. IEEE Computer Society, pp. 31–35. DOI: 10.1109/VISUAL.2019.8933747.

Shaer, O. and E. Hornecker (2009). "Tangible User Interfaces: Past, Present and Future Directions". In: *Foundations and Trends in Human-Computer Interaction* 3.1-2, pp. 1–137. DOI: 10.1561/1100000026.

Shao, L., A. Mahajan, T. Schreck, and D. J. Lehmann (2017). "Interactive Regression Lens for Exploring Scatter Plots". In: *Computer Graphics Forum* 36.3, pp. 157–166. DOI: 10.1111/cgf.13176.

Shneiderman, B. (1983). "Direct Manipulation: A Step Beyond Programming Languages". In: *IEEE Computer* 16.8, pp. 57–69. DOI: 10.1109/mc.1983.1654471.

Shneiderman, B. (1994). "Dynamic Queries for Visual Information Seeking". In: *IEEE Software* 11.6, pp. 70–77. DOI: 10.1109/52.329404.

Shneiderman, B. (1996). "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations". In: *Proceedings of the IEEE Symposium on Visual Languages*. IEEE Computer Society, pp. 336–343. DOI: 10.1109/VL.1996.545307.

Spence, R. (2007). *Information Visualization: Design for Interaction*. 2nd edition. Prentice-Hall.

Spindler, M., C. Tominski, H. Schumann, and R. Dachselt (2010). "Tangible Views for Information Visualization". In: *Proceedings of the International Conference on Interactive Tabletops and Surfaces (ITS)*. ACM Press, pp. 157–166. DOI: 10.1145/1936652.1936684.

Srinivasan, A. and J. T. Stasko (2018). "Orko: Facilitating Multimodal Interaction for Visual Exploration and Analysis of Networks". In: *IEEE Transactions on Visualization and Computer Graphics* 24.1, pp. 511–521. DOI: 10.1109/TVCG.2017.2745219.

Stolper, C. D., A. Perer, and D. Gotz (2014). "Progressive Visual Analytics: User-Driven Visual Exploration of In-Progress Analytics". In: *IEEE Transactions on Visualization and Computer Graphics* 20.12, pp. 1653–1662. DOI: 10.1109/TVCG.2014.2346574.

Thomas, J. J. and K. A. Cook (2005). *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Computer Society.

Tominski, C. (2011). "Event-Based Concepts for User-Driven Visualization". In: *Information Visualization* 10.1, pp. 65–81. DOI: 10.1057/ivs.2009.32.

Tominski, C. (2015). *Interaction for Visualization*. Synthesis Lectures on Visualization 3. Morgan & Claypool. DOI: 10.2200/S00651ED1V01Y201506VIS003.

Tominski, C., J. Abello, and H. Schumann (2004). "Axes-Based Visualizations with Radial Layouts". In: *Proceedings of the ACM Symposium on Applied Computing (SAC)*. ACM Press, pp. 1242–1247. DOI: 10.1145/967900.968153.

Tominski, C., C. Forsell, and J. Johansson (2012a). "Interaction Support for Visual Comparison Inspired by Natural Behavior". In: *IEEE Transactions on Visualization and Computer Graphics* 18.12, pp. 2719–2728. DOI: 10.1109/TVCG.2012.237.

Tominski, C., S. Gladisch, U. Kister, R. Dachselt, and H. Schumann (2017). "Interactive Lenses for Visualization: An Extended Survey". In: *Computer Graphics Forum* 36.6, pp. 173–200. DOI: 10.1111/cgf.12871.

Tominski, C. and H. Schumann (2008). "Enhanced Interactive Spiral Display". In: *Proceedings of the Annual Conference of the Swedish Computer Graphics Association (SIGRAD)*. Linköping University Electronic Press, pp. 53–56. URL: https://www.ep.liu.se/ecp/034/013/ecp083413.pdf.

Tominski, C. and H. Schumann (2020). *Interactive Visual Data Analysis*. AK Peters Visualization Series. CRC Press. DOI: 10.1201/9781315152707.

Voida, S., M. Tobiasz, J. Stromer, P. Isenberg, and S. Carpendale (2009). "Getting Practical with Interactive Tabletop Displays: Designing for Dense Data, Fat Fingers, Diverse Interactions, and Face-to-face Collaboration". In: *Proceedings of the International Conference on Interactive Tabletops and Surfaces (ITS)*. ACM Press, pp. 109–116. DOI: 10.1145/1731903.1731926.

Yi, J. S., Y. ah Kang, J. T. Stasko, and J. A. Jacko (2007). "Toward a Deeper Understanding of the Role of Interaction in Information Visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 13.6, pp. 1224–1231. DOI: 10.1109/TVCG.2007.70515.