A Framework for Hierarchical Time-oriented Data Visualisation

Rafael Henkin

Department of Computer Science
City, University of London

A thesis submitted for the degree of
Doctor of Philosophy in Computer Science

May 2018

Declaration

I declare that this thesis titled and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- The University Librarian may exercise his powers of discretion to allow this thesis to be copied in whole or in part without further reference to the author.

Rafael Henkin
May 2018

Acknowledgements

I would like to thank my supervisors Aidan Slingsby and Jason Dykes for the support throughout the PhD journey, especially the short and sometimes very long meetings that at times left my head spinning while looking for the right path ahead. I also thank the rest of the giCentre, staff and students that helped me over the years and that I hope to have helped a bit too.

I thank my sponsor CAPES for the financial support.

The research presented in this thesis would not have happened without the complete support of my family, in particular my parents Hélio and Ida Mariza. Big thanks to my brother Marcelo and the rest of the family for the long distance and the occasional close distance support.

Finally, I thank Laura for all the love and the unfaltering motivation.

Abstract

The paradigm of exploratory data analysis advocates the use of multiple perspectives to formulate hypotheses on the data. This thesis presents a framework to support it through the use of interactive hierarchical visualisations for the exploration of temporal data. The research that leads to the framework involves investigating what are the conventional interactive techniques for temporal data, how they can be combined with hierarchical methods and which are the conceptual transformations that enable navigating between multiple perspectives.

The aim of the research is to facilitate the design of interactive visualisations based on the use of granularities or units of time, which hide or reveal processes at various scales and is a key aspect of temporal data. Characteristics of granularities are suitable for hierarchical visualisations as evidenced in the literature. However, current conceptual models and frameworks lack means to incorporate characteristics of granularities as an integral part of visualisation design. The research addresses this by combining features of hierarchical and time-oriented visualisations and enabling systematic re-configuration of visualisations.

Current techniques for visualising temporal data are analysed and specified at previously unsupported levels by breaking down visual encodings into decomposed layers, which can be arranged and recombined through hierarchical composition methods. Afterwards, the transformations of the properties of temporal data are defined by drawing from the interactions found in the literature and formalising them as a set of conceptual operators. The complete framework is introduced by combining the different components that form it and enable specifying visual encodings, hierarchical compositions and the temporal transformations. A case study then demonstrates how the framework can be used and its benefits for evaluating analysis strategies in visual exploration.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

AVO Abstract Visualisation Objects

EDA Exploratory Data Analysis

HiVE Hierarchical Visualisation Expression

Chapter 1

Introduction

1.1 Motivation

Exploratory data analysis (EDA) (Tukey, 1977) calls for the use of *multiple perspectives* on data to help formulate hypotheses before proceeding with other analytical methods. Interactive data visualisation supports this approach with the use of various visual encodings that form these perspectives and the interactions that enable navigating between them. The resulting visual exploration process is empowered by enhanced pattern recognition through abstraction and aggregation and the visual organisation of data based on structural relationships (Card et al., 1999).

These benefits of visualisations are particularly related to temporal or time-oriented data – that is, datasets containing records observed or measured over time. Recent advances have greatly increased the capability of recording and storing such type of data; analysing it requires appropriate visual methods that consider the various aspects of temporal data. One such aspect is the linearity or periodicity of time, which depends on the use of certain granularities or the units of time. Figure 1.1 shows an example of two visualisations of temporal data: on the left side, the bar chart emphasises the *linear* order for months of the year, whereas the chart on the right side emphasise the *cyclic* aspect of the months.

Additionally, granularities are related when formed into *calendars*, which are lattice structures (Bettini et al., 1998) that define the mappings between them, as seen in fig. 1.2. The relationship between the mappings can be explored to analyse events that happen at different scales, such as movement processes (Nathan et al., 2008).

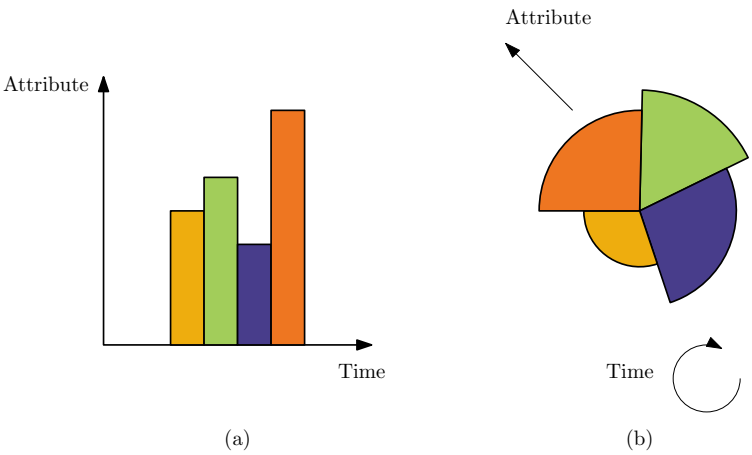


Fig. 1.1 Examples of visualisations emphasising: (a) linear aspect, (b) cyclic aspect.

This reflects the temporal resolution at which the data was captured and can be used in the visual organisation of the data. For example, data collected at a scale of *minutes* can also be visualised using coarser granularities such as *hours* – this, however, requires the use of aggregation methods to represent non-temporal data. More complex arrangements can be visualised by extracting parts of the granularity lattice and combining granularities.

The need for encodings and interactions for effective visual exploration, given these aspects of time, resulted in a number of techniques for interactive visualisation of temporal data from many domains (Aigner et al., 2011). Although many of the techniques used to show temporal data predate the use of computers, from a few centuries to thousands of years (Funkhouser, 1936), the design of modern visualisations is driven by the aspects of time. For example, Lammarsch et al. (2009) (fig. 1.3) and Van Wijk and Van Selow (1999) designed visualisations that use the hierarchical aspect of granularities. Others, such as Carlis and Konstan (1998), displayed temporal data in *spirals* that emphasise both the linear and cyclic aspects of time.

The development of these interactive techniques led, in turn, to the emergence of theories and models for time-oriented visualisation, helping to organise the existing literature and enabling further research through a common foundation for visualisation design. High level surveys (Silva and Catarci, 2000) and overviews (Aigner et al., 2007b; Muller and Schumann, 2003) mapped out the literature of visualisation methods, adapting terminology and concepts of time from other research areas such as artificial

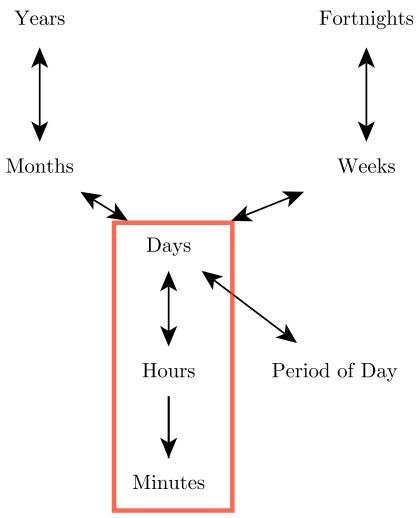


Fig. 1.2 Example of a calendar lattice. Although lattices are not intrinsically hierarchical, the paths between the time units in the calendar enables a hierarchical structure to be extracted from it, as highlighted in the figure.

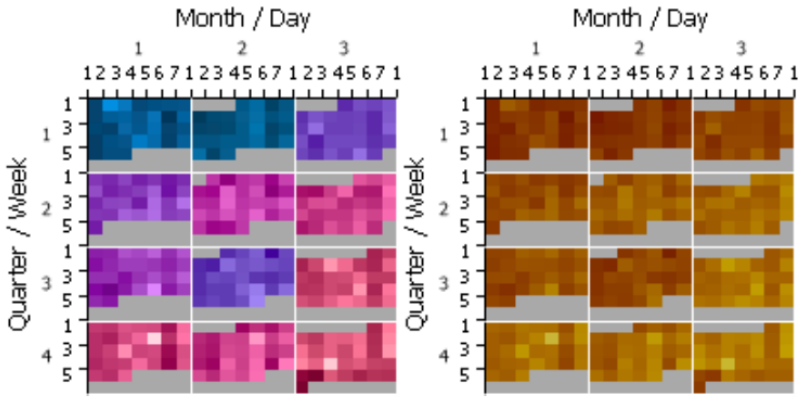


Fig. 1.3 GROOVE visualisation (Lammarsch et al., 2009), which enables users to reconfigure grid-based views based on calendar units.

intelligence (Allen and Hayes, 1985), databases (Dyreson et al., 2000) and geographic information systems (Peuquet, 1994). These concepts were later used to develop lower level, *generative* approaches that improve the *process* of visualisation design, such as taxonomies (Daassi et al., 2005), conceptual (Aigner et al., 2007a) and descriptive (Bach et al., 2016a) frameworks and design spaces (Brehmer et al., 2017).

Although there is a clear evolution regarding the ability of these works to include aspects of time as central parameters in interactive visualisation design, the concepts

and design spaces defined are limited to describing only data and visual aspects of interactive visualisation. For instance, although the conceptual framework by Aigner et al. (2007a) includes aspects such as the dimensionality of the visualisation (2D or 3D) and the related temporal data (such as the number of granularities represented in a visualisation), it does not discuss what are changes in the temporal data when users click on some visual object or drag a slider to manipulate the granularities. Another example is the design space by Brehmer et al. (2017), which describes visual arrangements based on temporal aspects but does not describe the conceptual meaning of the changes between the arrangements. Enumerating the transformations of time that can be triggered as interactions facilitates the analysis of exploration strategies and their replication (Ragan et al., 2015), through logging as *steps* that lead to the various perspectives in the visual exploration process. It also enables systematic exploration of the transformations and associated parameters.

At the same time, in these works, temporal granularities are included as simply an aspect to consider in the visualisation design process, rather than a central aspect that drives this process based on their characteristics. Examples from the literature (Lammarsch et al., 2009; Van Wijk and Van Selow, 1999) showed, in isolation, that various arrangements of temporal granularities can be effectively explored with *hierarchical visualisations*. In this category of visualisations, hierarchical structures are represented through visual encodings that highlight, explicitly or implicitly, the relationship between the different levels in the hierarchies. For time visualisation, this means exploring the relationship between the different granularities and also non-temporal dimensions contained in the data.

A considerable body of research in this area includes models for hierarchical visualisation design (Elmqvist and Fekete, 2010) and composite visualisations (Gleicher et al., 2011; Javed and Elmqvist, 2012), design spaces for implicit hierarchy visualisation (Schulz et al., 2011) and notations for describing visualisations (Slingsby et al., 2009). These theoretical works help design *general* hierarchical and composite visualisations, yet there is no theory or model linking them with the established theories for time-oriented visualisations that would facilitate including the multitude of combinations between temporal granularities in the visualisation design process.

These two identified gaps motivate the research presented in this thesis, which aims to investigate how aspects of interactive temporal data visualisation and hierarchical visualisations can be incorporated in the visualisation design process. The enhanced visualisation process enables systematic exploration of multiple perspectives by exploit-

ing the various characteristics of time through encodings and interactions, therefore facilitating visual exploration of temporal data.

1.2 Research question

In order to address the gaps identified in the previous section, the following question guides the research presented in this thesis:

- **Primary research question:** How can interactive hierarchical visualisations help explore temporal data?

In order to answer the question, three secondary questions are defined and drive the methodology applied in the research:

- **RQ1: What are the interactive visualisation methods used for temporal data?** This question is connected to the second part of the primary research question — visual exploration of temporal data, which defines the scope of the research. To answer this question, a survey of interactive visualisation methods is presented and organised following criteria that emphasise characteristics of encodings that are related to hierarchical visualisations.
- **RQ2: How can hierarchical composition techniques be combined with the surveyed visualisations?** This question connects RQ1 with the first part of the primary question — hierarchical visualisations, the area of information visualisation research that is investigated within the defined scope. Concepts of hierarchical visualisations and composite visualisations are defined and applied for use with the surveyed methods.
- **RQ3: What are the temporal interactions that facilitate exploring temporal data in hierarchical visualisations?** This question is related to the gap identified in the first section about the use of interactions in frameworks for temporal data visualisation. In addition to static specification of visualisations, this work proposes temporal transformations of visualisations in the form of operators that can be triggered by different types of interaction, such as direct manipulation or selecting from drop-down lists.

1.3 Research contributions

Combining the answers to each secondary question enables answering the primary research question and arriving at the following primary contribution of this thesis:

- **Primary contribution: Framework for Hierarchical Time-Oriented Interactive Visualisation**

The framework presented in this thesis addresses the research gaps identified in the first section. It comprises three components based on the enumerated research questions: *view*, *composition* and *transformation*. The view component defines properties for specifying visualisations based on the combination of the answers to RQ1 and RQ2. The composition component defines the visual composition methods for *structuring* the views into hierarchical visualisations, which are the answer to RQ2. Lastly, the *transformation* component defines temporal operators that transform the properties of time and alter the characteristics of time in visualisations – the effects of these transformations is explored further in the thesis.

Additionally, two of the components are also presented as *self-contained* research contributions:

- **Secondary contribution 1: Layered specifications of interactive visualisations for temporal data**

The specifications of surveyed visualisations techniques are presented as a novel contribution to the literature. In contrast with existing surveys and overviews (see chapter 2), the visualisations found in this thesis' survey are presented in the form of *decomposed* layers, where each layer forms a *view* that is hierarchically structured using any of the suggested visual composition methods. The decomposition into layers allows comparing the use of these basic blocks across visualisation techniques and the formation of a multi-dimensional *design space* for exploring *multiple perspectives*.

- **Secondary contribution 2: Temporal transformations for supporting visual exploration**

These are operators that are part of the *transformation* component of the framework. These operators are transformations of properties of the time domain, that is, the *times* that are used as references of items in the dataset that is being visualised. This component is presented as a contribution that can be used independently from the rest of the framework in non-hierarchical contexts. According to the review of

state-of-the-art literature presented in this thesis, it is the first complete specification of the temporal transformations that can be implemented as interactions in temporal data visualisations.

1.4 Scope

An ambitious goal would be to develop a framework that encompasses *all* visualisation methods and types of data that are related to time. However, there were time and technical constraints that prevented that goal from being achieved. Therefore, a realistic scope was set, being limited, first of all, to two-dimensional techniques that do not include animation. In these techniques, the visualisation itself only changes through user interaction (Muller and Schumann, 2003); the alternative would be visualisations where any aspect of the visualisation changes automatically with animation. Designing such visualisations includes a host of other design decisions, such as the speed of animation and interaction techniques for controlling animation, for which this thesis can serve as a starting point. The restriction to two-dimensional visualisations is done on a similar basis: including an additional dimension has different design challenges, such as dealing with object occlusion and deciding how to make transformations of 2D visualisations compatible with 3D visualisations and vice-versa. Additionally, there is evidence that viewers are better when comparing positions of objects in 2D and better when comparing shapes in 3D (St. John et al., 2001); as it will be seen throughout the thesis, position is the most common way of representing changes over time.

Within hierarchical visualisations, the scope of this project is limited to implicit hierarchical visualisations (Schulz et al., 2011). These are the visualisations where the *links* between different levels in the hierarchical structure are not displayed as visual marks. This is in contrast with explicit hierarchical visualisations, where connections between items in different levels are displayed through lines. Drawing the lines brings a whole range of design challenges such as the use of visual variables to modify them and finding the appropriate position of items in different levels to optimise the layouts; to keep a realistic timeframe for the research, explicit hierarchical visualisations were not considered. Finally, the framework only touches the surface of spatiotemporal visualisation design, such as allowing multiple geographic features to be included in a *map* visualisation or the visualisation of flows. Nevertheless, spatial dimensions can still be used with layouts that project spatial positions. These limitations are described throughout the thesis in the relevant chapters and revisited again in Chapter 9, in the

context of further work to extend the framework or applying the framework in other contexts.

1.5 Thesis outline

The rest of this thesis is organised as follows:

Chapter 2 introduces the main concepts and related works in the areas of time-oriented visualisation and hierarchical visualisations, as well as other related research areas. The characteristics of time that are explored in the are described in detail and the terminology used in the literature is clarified. A critical view of the theories and models mentioned in section 1.1 is introduced, contrasting the models and frameworks with the work presented in this thesis and describing how they inform the research. The major works in the areas that support this research, interaction for visualisation and notations and languages for describing visualisations, are summarised.

Chapter 3 describes the methodology used to answer the primary research question. An overview of the framework is given, along with justifications for the choices and methods used to answer each secondary question. The components that are the answers to each secondary question are then described in detail across three chapters. In chapter 4, the data and visualisation models that support the framework are introduced; visual composition methods are described in relation to hierarchical approaches and characteristics of time, resulting in the *composition* component. In chapter 5, the *view* component is presented as the answer to the question that enables specifying visual encodings for temporal data visualisation. Finally, in chapter 6, the temporal transformations included in the *transformation* component are described.

Chapter 7 combines the three components in the framework presented as the primary contribution of this thesis. An overview of the complete framework is presented, followed by a detailed description of the *interplay* between each component and their effects and uses for visual exploration.

Chapter 8 demonstrates how the framework can be applied in visual exploration through a case study in the context of a data visualisation challenge. The chapter describes how each part of the framework supports the challenge's visual exploration tasks, which are then answered with visualisations and interactions specified with the framework. The chapter concludes with an analysis of the exploration strategy that is used, demonstrating the benefits of the framework for exploratory analysis.

Chapter 9 concludes the thesis, reflecting on the research question outlined in the first chapter and placing the contributions of the thesis in this context. The limitations set by the scope are revisited and directions for further investigation of hierarchical time-oriented visualisations are explored. The gaps identified in the classification of visual encodings and interactions are also revisited to map future research directions.

1.6 Chapter summary

Two main lines of research drive this thesis: time-oriented or temporal data visualisation and hierarchical visualisations. Beginning with a survey of temporal data visualisation, each chapter of the thesis brings both fields one step closer, ending with a combined framework for hierarchical time-oriented visualisations that is the main contribution of this research. The next chapter lays out the groundwork for the rest of the thesis by introducing the concepts that permeate the field of theoretical works for time visualisation and hierarchical visualisations.

Chapter 2

Background and related work

As stated in the first chapter, the work presented in this thesis is positioned at the intersection of two areas of information visualisation research: temporal data visualisation and hierarchical visualisations. This chapter introduces the main concepts from these areas that support the work, the related works where the addressed research gaps were identified and other relevant supporting literature.

2.1 Background

2.1.1 Concepts and theories of temporal data

Exploring the temporal dimension enables understanding and explaining past events, analysing actions in the present and predicting future outcomes. Time has been thoroughly explored across many areas, with different ways of perceiving and representing time. For instance, *chronobiology* (Halberg, 1969) is concerned with the temporal characteristics of biologic phenomena, including sub-topics such as *chronophysiology*, *chronotoxicology* and *chronopathology*. In artificial intelligence, formalising the notion of time enables computational *reasoning* about time and relating non-temporal assertions to temporal assertions (Pani and Bhattacharjee, 2001). Yet another area of study is geography, where understanding *spatiotemporal* processes necessarily includes time in order to derive cause and effect relationships and represent *change* (Peuquet, 1994). In some of these areas, time is conceptualised according to their specific needs; higher level concepts and characteristics of time, however, can be used across multiple disciplines.

For example, describing the multiple *units of time* as granularities is relevant to both geography and artificial intelligence.

As temporal data visualisation is closely related to the lower level aspects of representation of temporal *information*, some areas of study are more relevant than others. Aigner et al. (2011) highlighted in particular the works of Goralwalla et al. (1998), in databases, and Frank (1998), in geographic information systems. Goralwalla et al. described an object-oriented model of time based on four aspects of temporal data: the structure of time, the representation of time, the order of time and the history of time. The following sections introduce these four aspects and relate them to data visualisation.

Temporal structure

The temporal structure contains the basic temporal features of a *model* of time. Three components are defined by the author: the type of time *primitive*, the type of time *domain* and the *determinacy* of time. The time primitives represent the different ways that observed time can be described: *anchored* primitives can have a *temporal location* (e.g. November 23rd), while *unanchored* primitives refer to *quantities* of time (e.g. 3 days). Anchored primitives are divided into *instants*, which are single anchored times, and *intervals*, which are pairs of anchored times. The only unanchored primitive is a *span* or duration.

The *domain* refers to the mathematical structure that is used to describe the primitives: in *discrete* domains, time primitives are modelled after integers: *0 second* is always followed by *1 second* ands this applies to every unit of time. In *continuous* domains, time can be infinitely divided: *0 second* can be *0.1 second*, *0.2 second* and so on. Bettini et al. (1998) also introduced the idea of *lower* and *upper bounds* of a time domain; as, in practice, very often data is collected over a certain period of time instead of *infinitely*, the bounds define the starting and ending times of the collection period.

Finally, *determinacy* refers to the *uncertainty* that is inherent to the relationship between the different time units and the non-temporal observations: an event that occurred on *November 23rd* is *indeterminate* in relation to the *hours* of that day.

Structure and visualisation: visualising temporal data requires deciding through which visual channels the temporal dimension will be displayed. The structure of time directly influences this decision; many times it is directly related to the data and thus not always under control of the visualisation designer. Most visual channels

are appropriate for representing instants. Intervals, on the other hand, require the representation of both its starting and ending times; durations can also be derived from intervals. This leads to the following design choices, illustrated in fig. 2.1:

- *Anchored visual mark*: the *length* property of visual marks can be used to display the temporal interval by mapping time to one positional visual channel and anchoring the visual mark at the appropriate starting and ending times;
- *Unanchored duration*: when there is no need to display time in a positional visual channels, the interval duration can be calculated instead. In this case, the *duration* ceases to be a *reference* and becomes a non-reference attribute;
- *Mixed approach*: a cartesian representation called *triangular model* (Qiang et al., 2012) uses both methods to display interval *data*. In this type of visualisation, the temporal domain is mapped to the horizontal axis, while a scale of duration is mapped to the vertical axis. In this case, point-based visual marks are used to represent intervals and anchored in the calculated *midpoint* of the interval. Vertically, the point is positioned in the corresponding value of the duration. This combination severely limits the choices of visual encodings for non-temporal attributes and is discussed again in later chapters. Another mixed approach is the representation of *set of possible occurrences* (SOPO) proposed by Rit (1986) which emphasises the uncertainty of events happening during an interval.

Temporal representation

This aspect is concerned with human readability and usability with regard to time; it relates to the multiple units of time (or granularities) that humans use to represent the passage of time and the temporal scales that result from the organisation of these units, primarily in the form of calendars. The relationship between different granularities and how to manage them has also been explored in areas such as databases; Bettini et al. (1998) introduced the formal concept of a *calendar*, while Dyreson et al. (2000) described a model to *efficiently* support granularities in databases based on these concepts. Goralwalla et al.'s concept of calendar is a set of granularities, with definitions of labels and number of time points in each granularity, and *conversion* functions between the granularities.

Representation and visualisation: granularities determine the number of time points that must be displayed; this primarily affects the choice of encodings that are

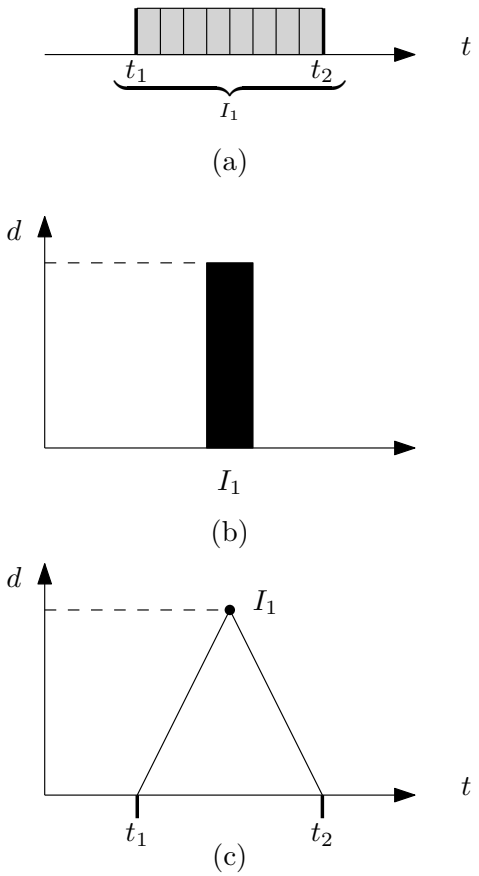


Fig. 2.1 Visual representations of intervals. In (a), an anchored visual mark is used to represent interval I_1 in a temporal horizontal axis. In (b), the duration of the interval is derived and displayed as the length of an unanchored bar; in this case, the horizontal axis may contain a distribution of intervals, for example, for comparing their duration. In (c), the mixed approach is used; the point corresponding to I_1 is placed on the *midpoint* of the interval in the horizontal axis and on the corresponding duration in the vertical axis.

used to visualise the data, which must be configured to facilitate addressing the tasks that are related to the granularity. In interactive visualisations, the support for *multiple* granularities must also be considered; this is further discussed in this chapter and is one of the direct motivations for this research.

Temporal order and history

Temporal order refers to the flowing of time in relation to the number *observers*. Goralwalla et al. define two types of order: linear and branching. In a linear domain, time flows from past to future in a *single* viewpoint. Branching domains contain points in

time where viewpoints on the data diverge. One example is a measurement that is consistent up to a certain date; from that date on, different measurements are taken and share the same temporal reference. Order is also closely related to the characteristics of granularities such as the cyclical nature of months, days, etc. In fact, Frank (1998) provided an alternative classification of time in which he considered *linear* and *cyclic* time separately from *branching time*: in this case, branching is considered as a characteristic of all attributes of the dataset instead of only the temporal domain.

Order and visualisation: the order is an *informing* characteristic of time that guides the design of visualisations. Visualisation can emphasise linear, cyclic or both aspects of time (see fig. 2.2). For instance, *circular* arrangements can be used with cyclical time, such as months of the year, where *January* and *December* are *contiguous* in the visualisation; months of the year can also be visualised through charts *from left to right*, where the cyclical aspect of months is not emphasised. Lastly, *spirals* are an example of visualisation that emphasise both aspects, as the spiral grows linearly and yet time points are visually aligned in cycles.

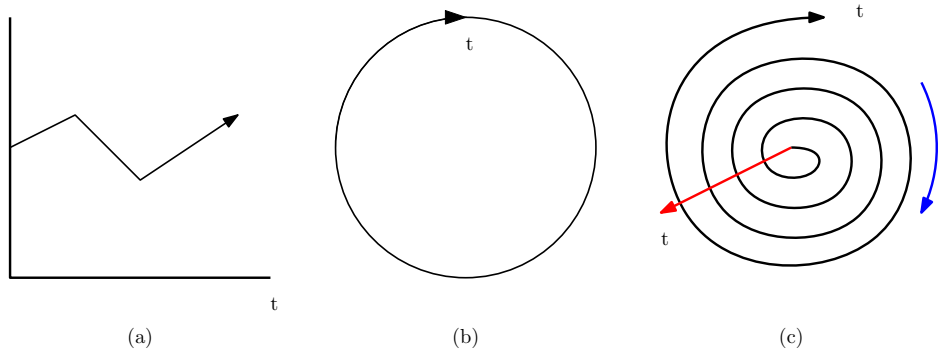


Fig. 2.2 Schematic representation of temporal order. In (a), *linearity* is emphasised. In (b), *cyclicity* is emphasised. In (c), the visualisation emphasises both linear (blue arrow) and cyclic (red arrow) aspects.

The *history* aspect refers to data management issues, such as deciding how non-temporal variables are related to temporal variables in observed records. As such, it does not have particular effects on the visualisations.

Summary

This section described the basic concepts of time primarily informed by Frank (1998); Goralwalla et al. (1998): how it can be *modelled* as *temporal data* and what are the primary characteristics that relate to *temporal data visualisation*. The following section

describes how frameworks and theories in the visualisation literature employ these concepts to facilitate visualisation design.

2.1.2 Temporal data visualisation

Evolution in temporal data visualisation resulted in the development of theories and models to lay out foundations for further efforts, allowing researchers to share knowledge through a common language, facilitating the application of existing techniques and identifying future research paths. In this section, the works that are directly related to the research presented in this thesis are critically analysed. These are the works that are most similar to this thesis regarding aim and content. The order used to describe them is based on the evolution in generative power, from more *descriptive* to more *generative*. These are not strict categories, but they convey the power of each work to describe existing literature or help generating new visualisations. In this survey, strong generative works may also be strong in their descriptive power, but, as it will be clear afterwards, the opposite is not true for the earlier works. This categorisation also helps to clarify the gaps that this research addresses.

Surveys and overviews

The first important work in the literature was the survey of visualisation of linear time-oriented data by Silva and Catarci (2000). The authors suggested a classification of methods following the framework specified by Goralwalla et al. (1998). The following four categories are used: *slice visualisation*, in which a single granularity of time is used in the representation; *periodic slice visualisation*, in which the multi-scale structure of granularities is used in the representation; *multi-slice visualisations*, in which multiple *versions of events* of a slice visualisation is represented — for example, simulations that produce different results beyond a certain point; and *snapshot visualisation*, in which only a single instant or interval of time is represented at any time. Within each category, the authors enumerate and describe sources from the literature that fit into that category.

There are also two tables summarising *visualisation features* and *interaction features* of the surveyed systems and tools. The first table contains a mix of the proposed classification, subjective categories, such as if the visualisation allows the identification of trends or patterns, and other general categories such as the domain-dependence of

the visualisation and use of the *focus+context* (Card et al., 1999) design principle. The second table contains categories derived from the general taxonomy of interactions by Shneiderman (1996) (see section 2.2.1) and temporal filtering, the only category of interaction *directly* related to time.

The classification was an initial investigation of the techniques used for representing temporal data and is an example of work that brings concepts developed in other areas - in this case databases - into information visualisation. It provides a general idea of how some characteristics of temporal data influence visualisation design, without discussing how this influence translates into the choices of visual encodings for each category of visualization.

Muller and Schumann (2003) also presented a general overview of visualisation methods for temporal data. While the authors use the taxonomy of *types of time* by Frank (1998), the criteria used to classify visualisations is based on aspects of representation rather than the taxonomy, which is only used to give context to the descriptions in each category. The classification contains three categories: static, dynamic and event-based representations. The first two are opposite categories, where static contains an explicit representation of the temporal domain. Dynamic representations, on the other hand, contain representations of time-oriented data that are similar to the *snapshot* category of visualisations suggested by Silva and Catarci (2000), where time is *implicitly* represented by animation (automatic or user-controlled). The third category contains visualisations that rather than showing changes in values of attributes, represent the point in time where a significant change happen. The authors do not provide clear criteria that define unique properties of event-based visualisation in relation to visual encodings.

The systematic overview by Aigner et al. (2011) is also based on the taxonomy proposed by Frank (1998). It classifies visualisation based on three aspects: time, data and representation. Time includes two criteria: temporal primitive (instant or interval) and temporal arrangement (linear, cyclic and branching). The data aspect includes characteristics of the dataset: frame of reference, number of variables and level of abstraction. The first criterion divides visualisations into abstract and spatial. The authors justify it due to the large difference between non-spatial and spatial encodings — a difference that is also considered in this thesis. The second criterion categorises visualisations into univariate and multivariate visualisation, due to the challenge of representing temporal data where *multiple attributes* change over time — either associated with multiple objects or with a single object. The third criterion,

level of abstraction, concerns the need for visual encodings that efficiently support visual analysis when the number of items is too large to be displayed without some level of aggregation being used. The representation aspect repeats the distinction between static and dynamic representations and also includes the dimensionality of the visualisation: 2D or 3D.

In terms of visualisation design, the classification maps out some *choices* for visualisation design — it does not include considerations about how one aspect limits choices in another aspect, for example. The authors note the fact that their classification is useful for single views — that is, designing a single perspective on a dataset — and call for the development of a general framework for time visualisation that considers *all* the enumerated aspects of time *with* multiple perspectives. While their work was succeeded by other frameworks that make advances towards that objective, as evidenced in next section, this call is still relevant as a research motivation.

Taxonomies, frameworks and design spaces

Moving towards *generative* works that provide structured approaches to facilitate designing visualisations, Daassi et al. (2005) introduced *patterns* of temporal data visualisation design (see fig. 2.3) by adapting the steps of the visualisation model by Chi (2000). The original model describes visualisation design as a pipeline of transformation steps from data modelling to rendering. In the proposed adaptation, the visualisation process is separated into non-temporal and temporal pipelines, which are *merged* at different stages of their pipelines.

There are five feasible patterns of combinations and the varying merging stages indicate the *dependency* between the non-temporal and temporal domains. Two cases are equivalent to the distinction between static and dynamic representation as in Muller and Schumann (2003). The other three cases consider the different stages where processes are merged: mutual dependency of the encodings on both processes, partial dependency of the non-temporal encodings on the temporal encodings and full independence of encodings. The mutual dependency process means that changes in any of the processes are reflected simultaneously on the other process. The authors define also two sub-cases in this category: whether a single set of visual marks represent both domains or there are *separate* sets of visual marks for each domain. Partial dependency of the non-temporal encodings means that the temporal domain is processed up to the rendering process, and visual marks for non-temporal domain are then processed *based*

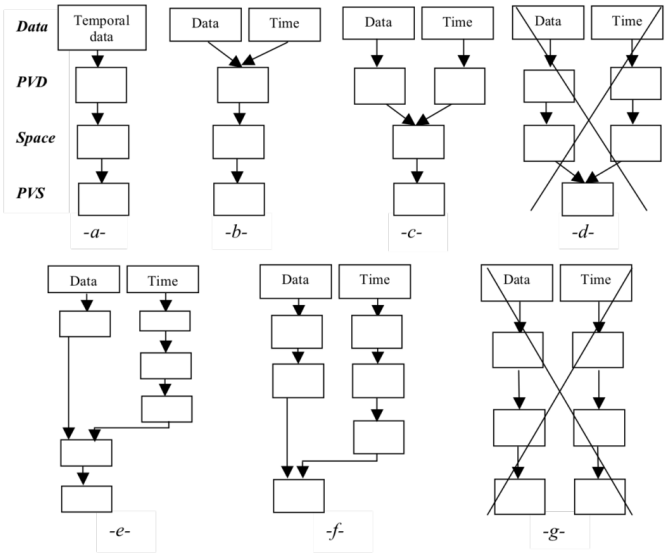


Fig. 2.3 Five patterns of combinations for time-oriented data visualisations by Daassi et al. (2005): (a) dynamic visualisations, (b) static visualisations, (c) mutual dependency of temporal and non-temporal data, (e) partially dependent encodings and (f) fully independent encodings. According to the authors, (d) and (f) are illogical within their taxonomy.

on the temporal domain pipeline. One example is using the time to *position* the visual marks of the non-temporal domain. Modifications of the visual marks of the non-temporal domain, such as changing an attribute that is being visualised, do not alter the temporal domain. The last combination, fully independent encodings, is where temporal and non-temporal domain are simply rendered at the same time, completely independently from each other. The authors define as *impossible* the patterns that would represent multiple views of a dataset, for example, one non-temporal view and one temporal view – although such combination is logical, it is likely that it is out of scope within the context of their taxonomy.

The equivalence of some of these combinations to composite visualisation methods is discussed in the next section. The taxonomy brings to light the differences between preparing temporal data for visualisation compared to non-temporal data, however, it does not describe in the detail these differences, such as what transformations are suitable for temporal data, or the details about the *choices* in each stage of the visualisation pipeline.

Aigner et al. (2007a) later introduced a conceptual framework largely based on their work described in the previous section. While it is oriented towards *visual analytics*

systems, most of the concepts apply to general interactive visualisations for time. The framework has three aspects: internal data structures, visualisation and interaction components and analytical and mining components. Internal data structures contain the same categories defined in the previous work, with the addition of different types of temporal scale (ordinal, discrete and continuous), granularities (none, single and multiple) and viewpoints (ordered, branching and multiple perspectives). It is the first conceptual work to explicitly consider temporal granularities as an important factor in designing visualisations. It also includes concepts related to anchored or unanchored time and distinction between event and states — the first involves data points representing *changes* in a state (e.g. event signalling that a plane has switched state from flying to landed), while, in the second, data points represent *continuous* measurements of the state (e.g. all the time points when a plane was flying and all the points when the plane was landed).

The visualisation component refers to the previously defined categories of static or dynamic representation, the type of non-temporal data (qualitative or quantitative) and dimensionality (2D or 3D). It also includes broadly the concept of *visualisation parameters* as part of *visual analytics systems*, referring to parameters of the *analytical component*, which relates to *data mining* algorithms such as clustering or forecasting. As it is a *conceptual* framework, it is yet another example of work that covers the surface of visualisation design for time, without delving deep into the variety of choices of visual encodings and interactions.

The choices of encodings is the focus of a recent work by Brehmer et al. (2017), who introduced a design space for timeline visualisation (fig. 2.4). Even though their scope is data-driven storytelling instead of exploratory interactive visualisations, the design space maps temporal concepts directly to characteristics of visualisations that are not exclusive to storytelling visualisations.

The design space covers visualisations in which time is explicitly represented through one or two positional channel and contains three dimensions to characterise this. The first dimension is *representation*, and describes the visual arrangement of time. Possible values include *linear*, *radial*, *grid*, *spiral* and *arbitrary* — the last one means that no specific *rules*, such as algorithms, guide the positioning of visual marks. *Scale* refers to the type of transformation used from the abstract temporal domain to the screen. For example, mapping years based on a *chronological* scale results in the distance between each visual mark representing a year being *equal* for all marks. Other possibilities include *relative*, *logarithmic*, *sequential* and a special type of scale called *sequential +*

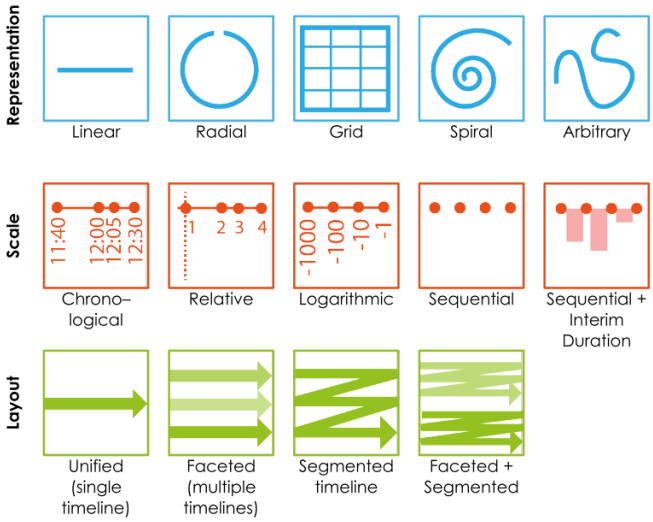


Fig. 2.4 Design space for timeline visualisation with representation, scale and layout dimensions (Brehmer et al., 2017).

interim duration where the temporal distance between marks is emphasised by different symbols. Finally, *layout* refers to the organisation of the data and the temporal domain. They include four possibilities: *unified*, *faceted*, *segmented* and *faceted+segmented*. Unified timelines refer to the visualisation of a single timeline; faceted timelines are *multiple* copies of timelines representing different objects or entities — effectively, it corresponds to the *multivariate* category of visualisation by Aigner et al. (2007a); segmented timelines are those in which a timeline is visually broken down based on time units — reflecting the existence of *multiple granularities*. *Faceted+segmented* represents the case where a timeline is broken down visually and also duplicated for multiple objects.

As mentioned, although the context of the design space is visual storytelling, there is no aspect of the proposed dimensions that make them applicable *only* to these cases. At the same time, while the design space comprises one subset of time visualisations – static visualisations with time mapped to at least one positional channel – the *scale* dimension can still be applied in other visual channels, as it is demonstrated in later chapters of this thesis. The *layout* dimension also reflects *general* possibilities regarding *multivariate* visualisations and the use of multiple granularities. The next section and chapters 4 and 6 also explain how these dimensions are used in the components of this thesis' framework.

Lastly, another similar work is the descriptive framework of temporal data visualisations by Bach et al. (2016a). The authors use the space-time cube (Kraak, 2008) as an abstraction from which visualisations can be derived, describing *operations* on the cube that lead to conceptually different visualisations. It is important to note that the framework is titled *descriptive* as opposed to *prescriptive* rather than opposed to generative. In this thesis, this work is classified towards the *generative* end of theories and models due to the fact that it can be effectively used to design visualisations rather than *only* classifying existing works, even if it does not include *recommendations* of the operations for specific tasks or datasets — there are no general theories in temporal data visualisation with such aim. Four large categories of operations are defined: *extraction*, *flattening*, *geometry transformation* and *content transformation*. Extraction operations are related to filtering and projecting parts of the space-time cube to a visualisation. Flattening operations are related to changing the aggregation level of one dimension of the space-time cube and projecting it. Geometry transformations alter the internal structure of the cube without modifying the contents of it (for example, not removing or adding points). Content transformations, on the other hand, involve operations like adding labels or colours. In total, forty-two operations are defined, including space and temporal variations of some operations — for example, interpolating points in time versus interpolating points in space.

Compared to this thesis and the works described so far, it has a stronger focus on the *conceptual* aspects of relating visualisation techniques through an abstraction rather than relating *visual* aspects. Visually dissimilar techniques are grouped together from the perspective of transformations of the space-time cube, while some visually similar techniques are shown to be conceptually dissimilar. The framework is also useful to *decompose* visualisations in the same perspective and to allow the analysis of different *paths* for designing visualisations. However, as acknowledged by the authors, while the transformations seem to be suited for an interactive approach, there is no discussion about how to use the transformations as part of an interactive visualisation. With the exception of the *time scaling* operation, that defines a *general* aggregation on the temporal axis following a numeric scaling factor, there are no other transformations that make use of temporal granularities and how the other transformations might be affected by the use of multiple time units.

On a lower level, in terms of abstraction level and detail, there is also the TimeBench (Rind et al., 2013), a software library which includes a data model based on the various characteristics of time described in this chapter. It is primarily based on mapping

multiple *temporal elements* to *temporal objects* and realising this mapping through an object-oriented model. The objective is to explicitly separate the *time primitives* as entities on their own, acknowledging that manipulating time is a complex operation that should not require reprogramming. For this research, this conceptualisation is an inspiration for describing the temporal transformations in chapter 6; however, in this thesis, the focus is to expose the properties of time that are required for the transformations rather than to formally define the relationship between the time domain and the non-temporal attributes or objects.

Summary

There is a clear evolution in theories for time visualisation, from early works with stronger *descriptive* power to recent works with stronger *generative* power, as summarised in table 2.1. While both types of work are desirable, there are still gaps in the generative literature that can be addressed. Notably, support for the effective use of temporal granularities is lacking, as well as time-centred interactions. Temporal granularities are acknowledged in descriptive models rather than generative ones, suggesting that it is important to investigate how to include support for granularities in generative models. At the same time, interaction is a secondary aspect in all of the reviewed works. The only reviewed model that contains some temporal transformations (the framework by Bach et al. (2016a)) does not describe them as transformations that could be triggered by users through interactions in visual exploration. As the remainder of this chapter shows, other areas of information visualisation can help to address these gaps.

2.1.3 Hierarchical visualisations & composition methods for visualisation

A major premise of this work is the idea that visual exploration requires an efficient use of the structure of temporal granularities. The hypothesis investigated is that a framework for *hierarchical visualisations* of time allows this efficient use. This section describes the concept of hierarchical visualisations that is used in this thesis and also explain what are the *composition methods* that are used in the framework that allow *multiple perspectives* on temporal data to be visualised.

Power	Authors	Year	Type	Visual aspects	Time Interactions	Granularities
Descriptive	Silva and Catarci	2000	Survey	x	x	
	Muller and Schumann	2003	Overview	x		
	Aigner et al.	2011	Systematic view	x		
	Daassi et al.	2005	Taxonomy	x		
	Aigner et al.	2007a	Conceptual framework	x	x	x
	Bach et al.*	2016a	Descriptive framework	x	x	x
Generative	Brehmer et al.	2017	Design space	x		x

Table 2.1 Summary of the conceptual works for time-oriented data visualisation, ordered by their descriptive and generative powers. While visual aspects are covered by all the works, interaction and temporal granularities are considered by only a small number of these frameworks and the most recent works are focused only on visualisation aspects. The power order also matches the temporal evolution of the references, signalling a progress in the ability of these conceptual works to facilitate *designing* new visualisations for temporal data. It is important to note that *more generative power* does not necessarily mean *less descriptive power* — the table shows the path that led to the current stage of conceptual works.

Hierarchical visualisations are those that represent parent-child types of relationship from hierarchical structures such as trees. These hierarchical structures can be formed by *true* or *explicit* hierarchical information or by *false* or *implicit* hierarchical information, which are formed arbitrarily. The explicit type of hierarchy comprises of information such as organisational structures, spatial hierarchies (eg. countries, states, cities) and temporal granularities (e.g. year, month, day of month). False hierarchies are formed by combining non-hierarchical information through *conditioning*: arranging data points in hierarchies matching selected attributes (e.g. year and region). Both types can be visualised through two categories of hierarchical visualisations (Schulz et al., 2011): explicit or implicit. Explicit *visualisations* encode the relationship between nodes in different levels of the hierarchies using marks such as lines or arcs between them. Implicit visualisations use positional variables to encode this relationship — positioning nodes from the highest level of the hierarchy near the top of the screen, for example, or placing children nodes *inside* representations of parent nodes. Although explicit visualisations are out of the scope of this thesis, they will be referred in this section as a type of composition method.

Schulz et al. described a *design space* for implicit visualisation of hierarchies, with dimensions that define the representation of the nodes themselves (*node representation*), the representation of the relationship between the nodes (*edge representation*) and the method for visually laying out the nodes (either starting from the *root* of the hierarchy, via subdivision, or with the *leaves*, via packing). They also include a dimensionality category, which can be 2D or 3D. The most important dimension related to this work is *edge representation*. Three types of representation are described:

- *inclusion*, in which a representation of a child node is *fully* rendered inside the representation of a parent node (fig. 2.5 (a));
- *overlap*, in which a representation of a child node partially overlaps the representation of a parent node (fig. 2.5 (b));
- *adjacency*, in which a representation of a child node is positioned *immediately* outside the representation of a parent node, without overlapping it (fig. 2.5 (c)).

The design space does not include any mention to layout algorithms, nor considers representations where the *alignment* is used to denote a parent-child relationship, such as in adjacency representations with the inclusion of *space* between nodes. The authors also do not discuss how the design space can be applied in temporal data visualisations. Another interesting characteristic of the design space is that the the dimensions are

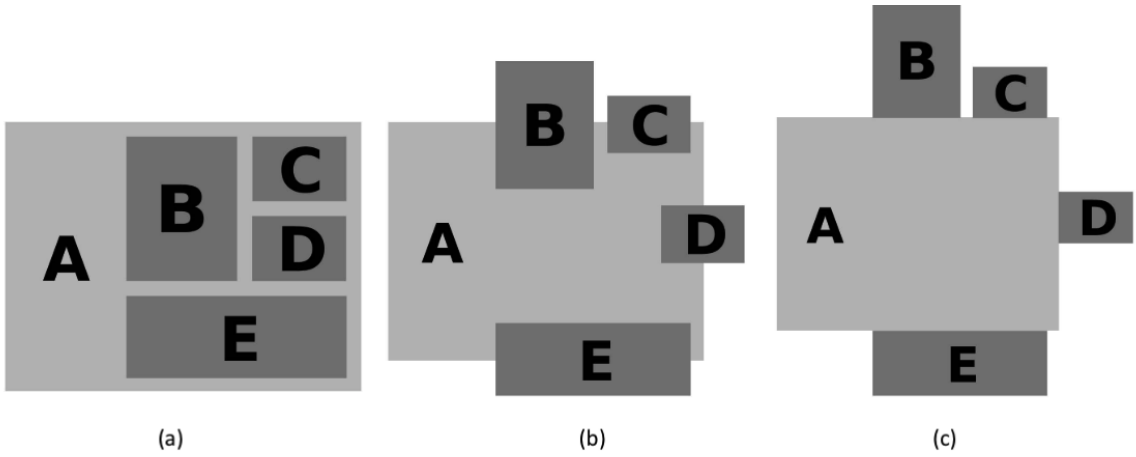
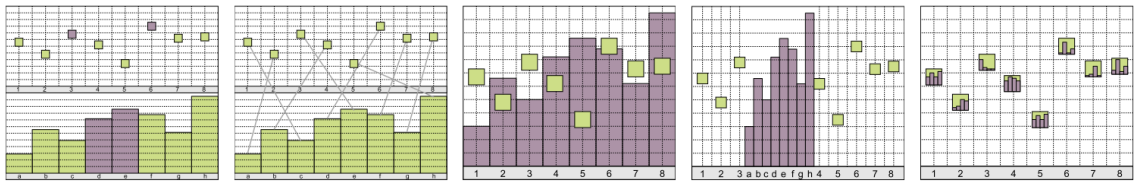


Fig. 2.5 Implicit hierarchical representation methods from Schulz et al. (2011): (a) inclusion, (b) overlap and (c) adjacency.

applicable to the hierarchical structure as a whole — there are no ways to apply the design space at different levels of hierarchies as it is described. The authors acknowledge that as part of an issue in *implementing* the design space. The fact that *instances* of the design space can be combined for different levels, which the authors call as *mixing design choices*, relates heavily to the concept of *composite visualisations*. These are visualisations containing *multiple perspectives* on the data, based on various types of visual arrangements. Javed and Elmqvist (2012) proposed five methods that generalise the previously existing *coordinated multiple views* paradigm (Roberts, 2007) and the *juxtaposing* and *superposition* methods proposed by Gleicher et al. (2011). They are:

- *Juxtaposition* (fig. 2.6 (a)) positions views *side-by-side*, in a similar way to the *adjacency* method suggested by Schulz et al.. This method is also conceptually similar to the idea of *fully independent encodings* by Daassi et al., without the requirement that encodings are different;
- *Integration* (fig. 2.6 (b)) also position views side-by-side, however, these also include *explicit* links between views, such as lines connecting marks representing the same data point in different views. Juxtaposed views, on the other hand, rely on *shared visual variables* as *implicit* links, such as the same colour scale being used in both views to identify data points across views;
- *Superimposition* (fig. 2.6 (c)) defines views rendered *on the same space* and *overlaid* on top of each other, such as maps containing multiple layers of features. The defining feature of this method is the fact that the underlying data in each



(a) Juxtaposed views. (b) Integrated views. (c) Superimposed views. (d) Overloaded views. (e) Nested views.

Fig. 2.6 Visual composition methods from Javed and Elmqvist (2012): (a) juxtaposition, (b) integration, (c) superimposition, (d) overloading and (e) nesting.

view is different, even though the *coordinate system* and boundaries of it are the same. One example is multiple line charts drawn on top of each other, where each line chart is considered a separate view.

- *Overloading* (fig. 2.6 (d)) is a variation of superimposed views, but, in this case, all the views share the same underlying data structure. This composition method describes visualisations where a different encoding, called *client* is overlaid on top of another one, called *host*, using the *visual structure* of the host. The examples given concern the case when a host visualisation can be enhanced by the addition of a client visualisation. This is conceptually similar to the *mutually dependent encodings* case with *separate* visual marks by Daassi et al..

- *Nesting* (fig. 2.6 (e)) defines a client-host relationship where the client visualisation is rendered *nested* inside visual marks of the host visualisation, replacing the original visual mark. While the authors do not require that the encodings of the client and the host are different for nested views, the examples they give are all based on this idea. Conceptually, the nesting defined in this design space can be seen as a case of mixing design choices in the implicit hierarchy design space. The notion of an implicit hierarchy is embedded in the fact that each client visualisation is replacing a visual mark of the host, therefore it contains only the data that is associated with the data point it is replacing.

Although the idea of *client-host* suggest a hierarchical relationship, it is only defined for overloading and nesting. Furthermore, the authors do not discuss the fact that visualisations can depict *multiple* encodings for the *same* object or the *same encoding* for *multiple objects*, even though some of the given examples fit into one of these categories. Gleicher et al. (2011) suggested *juxtaposing* and *superposition* for implicit compositions and *explicit encodings* for explicit compositions, in the context of *visual comparison* of multiple objects or events. These design choices regarding the organisation of datasets for the different compositions are described by Munzner (2014)

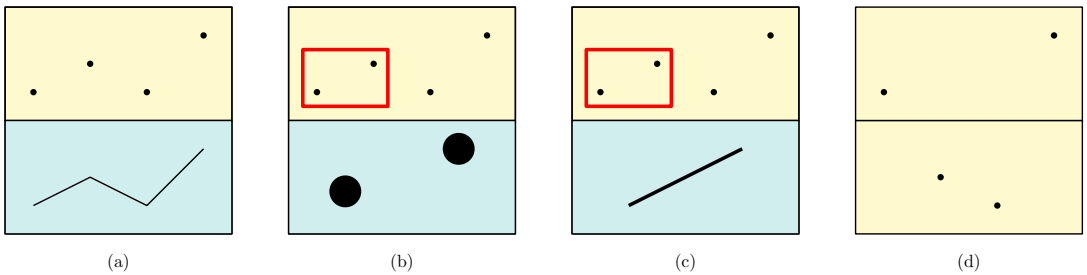


Fig. 2.7 Visual composition methods proposed by Munzner (2014): (a) different encoding, all data, (b) different encoding, one subset, (c) same encoding, one subset, (d) same encoding, multiple subsets.

as the decision about what parts of the dataset are shown in each view: the same data, a subset of the original dataset or disjoint subsets (facets) of the original dataset. Coupled with the option of using the *same* or *different* encodings, four combinations of data and encodings are defined:

- *Same encoding, one subset of data*: this combination describes visualisations where the same encoding that is used for the whole dataset is also used to provide a *detailed* view of *part* of the dataset, which is the *overview+detail* principle for visualisation (fig. 2.7 (c));
- *Same encoding, multiple subsets*: sometimes called *faceting*, this combination describes the the general concept of *small multiples* Tufte (1983), in which several views of disjoint *parts* of a dataset are shown side-by-side (fig. 2.7 (d));
- *Different encodings, all data*: this combination describes the concept of *multiform* or *multiple views*, which is the general term used for *composite visualisations* (fig. 2.7 (a));
- *Different encodings, one subset*: this describes the combination of the overview+detail principle with multiform visualisations (fig. 2.7 (b)).

Some of these combinations are special cases of the methods suggested by Javed and Elmqvist iterated for same or different encodings. However, Munzner only defined them for juxtaposed views rather than for *all* composite view methods. Likewise, it does not include the case where multiple subsets are displayed along with the parent set. This is only described as part of partitioned views and nested visualisations, along with a discussion about how to partition the data into subsets or disjoint subsets, following the idea that either *dimensions* or subsets of *values* can be used to organise

datasets hierarchically. Elmqvist and Fekete (2010) also used this notion of partitions and expanded it into a framework for hierarchical aggregation by introducing lower level concepts for implementing hierarchical visualisations, which are based on a direct representation using *visual aggregates*. The authors discuss how to perform the aggregation, what type of information can be conveyed through aggregated items (such as the average value of an attribute), rendering strategies and visual interaction mechanisms that allow the hierarchical structure to be explored without modifying the underlying structure. Yet again temporal data visualisation is not mentioned in this framework.

The *Hierarchical Visualisation Expression* (HiVE) notation by Slingsby et al. (2009) also has a model for hierarchical visualisations based on *false hierarchies*. The motivation for this model is to support answering research questions about a dataset using different configurations of false hierarchies and different perspectives on these configurations. The use of implicit hierarchies does not mean, however, that the visualisation must also be of the *implicit* type. While examples given by the authors use the *treemap* category of visualisations and are thus based on the *nesting* method, the method of composition itself is absent from the notation. This thesis uses the model reconfiguration of hierarchies used by HiVE as a motivation for applying hierarchical visualisations to temporal data. HiVE primary concern, however, is providing the notation to describe the arrangement of hierarchies. Visualisation aspects such as layouts and shapes are not specified or investigated by the authors. Additionally, the model only includes interactions in the form of operators that *rearrange* the hierarchies or modify visual mappings.

MacNeil and Elmqvist (2013) introduced *visualisation mosaics*, a model and notation for designing juxtaposed visualisations with a grid-like arrangement. The model is based on the notion of dividing a visual space into *groups* of *tiles* of visualisations. The space division and arrangement of the mosaics follows a recursive definition of the groups and tiles in the notation, but there is no notion of *data hierarchies*. The idea is that attributes of the data, not subsets, can be seen through different visualisations. A similar, non-hierarchical approach, was proposed by Wickham and Hofmann (2011) as *product plots*. These plots use false hierarchies to partition the view and display counts and proportions through area or length-based primitives. The false hierarchies are derived from statistical concepts such as *conditional probability*, from which the name derives. The main concept is generating charts by alternating vertical and horizontal primitives and display conditional proportions, resulting in implicit visualisations

which mix juxtaposition with nesting. Because of the focus on displaying counts and probabilities, none of the primitives are point-based or connected with lines and temporal variations are not considered.

Summary

This section examined the various *composition methods*, summarised in table 2.2, that can be used to display hierarchies, including methods not originally described for such purpose. The table highlights the similarity between the methods described in the literature that were not proposed with the same aim or even derived from each other, suggesting that the concepts of hierarchical and composite visualisations are highly related. Besides the notion of explicit and implicit hierarchy *visualisation*, the concept of explicit or implicit *hierarchies* was also described, as well as the various alternatives for combining encodings and data in multiple views. The surveyed methods, however, do not generalise the idea of partitioned or conditioned datasets for all composition methods; this is explored in chapter 4 as part of the contribution of this thesis.

Two other aspects that are missing in these works are *time* and interaction. The fact that, in some cases, temporal granularities are a natural match for hierarchical and composite views raises the question about *how* these methods can be used with temporal data, which motivate this thesis. With the exception of the HiVE notation, interaction is at most considered as part of the *navigational* aspects of exploring hierarchies that have already been defined, such as focussing on some parts of the hierarchy, but not as a means to enable users to interactively reconfigure hierarchies.

2.2 Further related work

The next three subsections contain brief overviews of areas that inform parts of the research presented in this thesis and help address the aforementioned gaps. The first, taxonomies of interaction methods for visualisation, relates to the *transformation* component of the framework presented in this thesis. Categories of interactions are described and analysed in the bigger picture of interactive visualisation. The second section contains notations and description languages for visualisations, which are used to describe how the visual encodings map to elements of the data — this is related to the *view* component of the framework. The third section contains an overview of tools and environments for visual exploration. Examples of these tools and the aspects of

Source	**Implicit**				**Explicit**
Javed and Elmqvist (2012)	Juxtaposition	Superimposition	Overloading	Nesting	Integration
Gleicher et al. (2011)	Juxtaposition	Superposition	-	-	Explicit encoding
Schulz et al. (2011)	Adjacency; Overlap	-	-	Inclusion	-
Daassi et al. (2005)	Fully independent encodings	-	-	Mutually dependent, separate marks	-
MacNeil and Elmqvist (2013)	Juxtaposition	-	-	-	-
Wickham and Hofmann (2011)	Juxtaposition	-	-	Nesting	-

Table 2.2 Summary of composite visualisation methods for hierarchical and non-hierarchical visualisations. The use of juxtaposition for both types of visualisation is clear across the literature, while the other methods vary between hierarchical and non-hierarchical visualisation. Empty cells are justified due to the different objectives of each work: the first two do not have an emphasis on hierarchical visualisations, while the third work does not cover explicit visualisation of hierarchies, which are also out of the scope of this thesis. The last column is included to show that at least one model for time visualisation contains similar concepts to composite visualisations.

visual encodings and interactions that they support are given in relation to temporal data and hierarchical visualisations.

2.2.1 Interaction taxonomies

Understanding how users interact with visualisations is essential to choose *which* method of interaction that is appropriate for the variety of tasks and visual encodings. While the field of *human-computer interaction* is concerned with most of the research on the methods that allow users to *input* their actions in computers, research in information visualisation also investigates the *science of interaction* (Pike et al., 2009) that helps with understanding the role of interaction in visualisation. Yi et al. (2007) and Roth (2012) argued that there is a large variation in the nature of taxonomies proposed within the scope of this science of interaction: some taxonomies focus on low-level interactions (Buja et al., 1996; Chuah and Roth, 1996; Dix and Ellis, 1998; Shneiderman, 1996), while others focus on high-level tasks (Heer and Shneiderman, 2012; Zhou and Feiner, 1998). Both also suggested that the variation can be explained by the *stages of action* model by Norman (1988). Roth, examining it in the context of cartographic visualisation, named it as as stages of *inter*action; between this author and Norman, the only difference is the name of the object being interacted with: the former refers to visualisations, the latter refers to any object in the real world.

The model, displayed in fig. 2.8 contains seven cyclical stages that describe how a user interacts with a system by: (1) forming a goal, (2) forming an intention, (3) specifying an action, (4) executing an action, (5) perceiving the state of the system, (6) interpreting the state of the system and (7) evaluating the outcome. The steps in the direction of the visualisation are named by Norman as the *gulf of execution*, whereas the steps in the direction of the user are named as the *gulf of evaluation*; in the *science of interaction*, one goal is to reduce these gulfs to facilitate the use of visualisations – this is not explored in this thesis. Roth (2013) explained the aforementioned variation in taxonomies by suggesting that interaction taxonomies can be classified into *objective* taxonomies, related to the first two steps of the cycle, *operator* taxonomies, related to step 3, and *operand* taxonomies, which are related to step 5 because they describe interactions based on the *target* of the interaction. In summary, a *visualisation* is conceptually represented by *operands* that are affected by *operators* triggered by the user, in order to fulfil his *goals/objectives*.

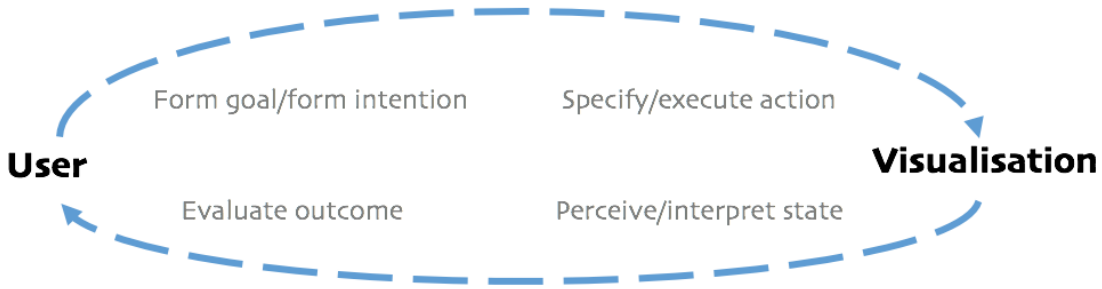


Fig. 2.8 Stages of action model by Norman (1988), with the inclusion of *Visualisation* by Roth (2012).

Among these steps and types of taxonomies, *operator* and *objective* taxonomies are those of most interest to this research. As stated in section 2.1.1, theories and models for time visualisation are rich in *operand* taxonomies but poor in *specifying actions*. Therefore, in order to develop the *transformation* component of this thesis' framework, the interactions included in interactive visualisations for time were classified and analysed. For this purpose, the categories of *interaction intents* proposed by Yi et al. (2007) were used. These categories represent high-level intents that do not depend on the visual encodings and can be mapped to low-level interaction techniques. The taxonomy does not contain objective measures that can be used to classify existing interactions, which means that some interactions can fit into more than one category. However, the descriptions are general enough that the interactions in the systems can be distinguished from each other at a suitable level. In comparison with existing taxonomies, the advantage of this taxonomy is that it does not imply the use of specifics visual encodings or low-level interaction mechanisms, such as those that describe interactions such as *brushing*. The categories are:

- *Select*: mark data items, separating them visually from the other items;
- *Explore*: navigate the *visual space* to view different parts of the dataset;
- *Reconfigure*: show different parts of the dataset by rearranging the visual space with different perspectives;
- *Encode*: change the visual representation of marks (e.g. colour);
- *Abstract/elaborate*: change the aggregation or abstraction level, either in the data space or the visual space;
- *Filter*: show or hide data items following specific conditions;

- *Connect*: change the visual representation of items *relative* to one or more items of interest, or introduce new items based on the same condition.

In chapter 3, it is explained how these categories were used to arrive at the *temporal transformations* of the framework, which are then described in chapter 6.

2.2.2 Notations and description languages for information visualisation

The literature covered so describes high level concepts related to visualisation. To enable applying these concepts in a practical medium, over the years, visualisation grammars, declarative embedded languages and libraries have been developed. To this thesis, the interest is in how these languages allow specifying the mappings from data items to visual marks and defining composite views.

The category of visualisation grammars includes the *Grammar of Graphics* (Wilkinson, 2005) (GoG), the *layered grammar of graphics* (Wickham, 2010) (LGoG), *Vega* [1] and *Vega-Lite* (Satyanarayan et al., 2016), which is an extension of Vega. They provide commands and rules at different abstraction levels to specify visualisations. While the internal models and semantics of the rules vary among these grammars, they all follow the underlying idea from the original GoG, which is describing visualisations with multiple basic blocks that map data items to the visual encodings. These basic blocks are composed of data transformations, definition of scales that transform values from data space to abstractions, different ways of projecting these scales with different coordinate systems and rules for assigning data and values to positional and retinal channels of visual marks. Additionally, these grammars also support other graphical features beyond the rendering of visual marks, such as drawing the *axes* that represent the scales or *tooltips* and *captions* in common charts. These grammars are accompanied by implementations in different languages, which also lead to different ways of building them. Particularly, Vega and Vega-Lite have JavaScript implementations that use *static* specifications of visualisations, in files that have a structured notation for describing *key-value* tuples (JSON). On the other hand, LGoG has been implemented as *ggplot*[2] in the R programming environment, which requires either running *scripts* to build visualisations or *inputting* commands to build the visualisations interactively.

[1] https://vega.github.io/vega/
[2] http://ggplot2.org/

In terms of hierarchical visualisations and composition, the support for them in these languages and grammars vary in the following ways: *default functionality*, *unique commands for composition* and *visual variable mapping*, with the last one also being divided into layout arrangements and retinal property mappings. Default functionality means that one composition method is used as default, unless another composition method is explicitly defined. This is the case for *superimposing multiple encodings for the same data* in GoG, LGoG and Vega — Vega-lite does not have any default functionality regarding compositions. *Unique commands* means that the language has specific properties or operators to define compositions, which is the case for the *facet* operator that is used in LGoG and Vega-lite, as well as the *repeat*, *concat* and *layer* operators in Vega-lite. The *facet* operator juxtaposes multiple subsets with the same encoding in the two grammars that have it. *Concat* and *repeat* are used to juxtapose views for the same dataset; the difference between them is that *concat* allows a completely different specification of additional views, while *repeat* has a systematic parameterised composition. The parameterisation means that multiple charts can be generated with only one visualisation parameter varying; for example, bar chart views for two different quantitative variables changing over time. *Layer* superimposes different encodings on top of each other; it can be combined with *repeat* and *concat* to generate multiple superimposed charts.

Some of these grammars also use the properties that specify the mapping of data items to visual channels as means of composition. In the GoG, this is done either through the configuration of the coordinate system or assigning categorical variables to retinal properties. As it is a grammar that is based on algebraic compositions of variables of a dataset, with operations such as *unions* and *cartesian products*, the resulting number of dimensions used in the composition defines the arrangement of juxtaposed views. The arrangement is also based on the order of variables in the algebraic composition. Furthermore, as *all* values of a dataset are shown (unless aggregation is specified), when assigning a categorical variable to the colour channel, for instance, points are painted according to the values of the assigned variable. In the case of colouring based on a categorical variable with three values, three colours will be used, signalling the existence of *subsets* in the data. This is different compared to the other three grammers, which have a mix of special aggregation abstractions and visual variable assignment for using subsets of data. LGoG provides a *group* property that defines which categorical variable is used to create the subsets, but assigning certain types of variables to certain channels like colour will have the same result as in the GoG. Vega contains a special *group* abstract mark that defines *panels* for multiple views;

	GOG	LGOG	Vega	Vega-Lite	HiVE
Juxtapose multiple subsets	C, L	O, L	G, L	O, L	V
Juxtapose one subset	C, L	*U*	G	O, L	*U*
Superimpose multiple subsets	V	V	G	V	V
Superimpose one subset	D	D	D	O	*U*
Nesting	*U*	*U*	G	L	D

Table 2.3 Table of support for composition methods in the main notations used for describing visualisations. *U* means unsupported, *D* means default functionality, *L* means support via layout definition, *G* means support by a special group visual mark, *O* means specific operator to specify the composition and *V* means support via visual variable assignment. Vega and Vega-Lite provide various levels of support for the different combinations.

these can be parameterised by the data as well to generate a variety of combinations of encodings through juxtaposition, superimposition and nesting.

In table 2.3, a summary of the composition methods in each notation is given, with the addition, for comparison, of the HiVE notation introduced previously. Besides the clear differences between them, it is also notable that nesting is generally under-supported by these grammars, also reflecting the limitations in the support for hierarchical data. Similar to the frameworks discussed previously, these grammars have varying strengths and weaknesses in terms of support for composition. This is also related to the differences between them and the research presented in this thesis, primarily explained by the fact that designing a general language for visualisation requires definitions of *syntax* and *semantics* in addition to the model. The work presented in this thesis focuses on the development of the model rather than the language used to specify it in practical use.

2.2.3 Tools and environments for visual exploration

In addition to programming languages, end user applications for visual data exploration have also been developed. These range from tools such as Mondrian [3], which enables users to analyse datasets with various visualisation methods, to Tableau (Tableau Soft-

[3] http://www.theusrus.de/Mondrian/

ware, 2018), which enables users to *design*, share and explore interactive visualisations. Other similar tools are Spotfire (TIBCO Software Inc., 2018) and Qlik (Qlik, 2018). To this thesis, the interest is in the aspects of these applications that involve hierarchical and temporal data visualisations. While some of these tools enable lower level reconfigurations, one aspect in particular is the possibility of configuring coordinated multiple views in a *dashboard* approach, which is present in most of them. In order to generate shareable dashboards, these tools also allow users to *transform* their data as needed, cleaning and modifying attributes.

Among the state-of-the-art tools, Tableau stands out in particular due to its origins in academic research (Stolte et al., 2002). It includes mappings from data to visual variables in a similar manner to the Grammar of Graphics, while also offering predefined configurations to users. In terms of hierarchical and visual composition aspects, it has a *tabular* view of the the data and visual space, with limited support for a *hierarchical* arrangement of views. Both Tableau and Spotfire allow users to rearrange views freely in the dashboard mode though. However, the rearrangement is not connected to the data, as there is no native support for hierarchically connecting views so that they are *superimposed* according to the composition methods described above.

These tools are included as part of the survey in chapter 6.

2.3 Chapter summary

This chapter has identified the gaps in the two main research areas that motivate this work, which are: conceptual works on time visualisation overlook interactions and temporal granularities, while conceptual works for hierarchical visualisations overlook the temporal dimension. These are discussed in detail as research opportunities emerging from the main works that define these two research areas. Moreover, other related works that inform the research have also been described. The gaps that were identified suggest the need for more research to bring closer the two areas: the next chapter describes the methodology that is used to develop the framework that helps to achieve that.

Chapter 3

Methodology

This chapter introduces the methodology used to arrive at the main contribution of this thesis, a framework for hierarchical time-oriented visualisations. The first chapter introduced the primary research question — *how can interactive hierarchical visualisations help explore temporal data?* — with three connected secondary questions. This chapter outlines how the framework is used to answer the primary question and how the secondary questions are used to design the framework. First, the context of visualisation design is described, explaining how existing visualisation models relate to and inform this thesis. Afterwards, each secondary question is reviewed, along with the justifications, supported by the literature, for the methods used to answer them.

3.1 Addressing the primary research question

The objective of this and the following chapters is addressing the primary research question, ***how*** *can hierarchical visualisations help explore temporal data?*, through three secondary questions. A framework that encompasses the aspects of visualisation is presented as the answer to the primary question, covering various steps of visualisation design as shown in fig. 3.1. Each component of the framework is derived from investigating the secondary questions. The *view* component is concerned with the description of visual encodings used in visualisations, from a hierarchical and temporal perspective. The *composition* component includes *hierarchical* aspects and the *exploration* of composite multiple perspectives, based on the paradigm of exploratory data analysis. The focus on temporal data exploration is enabled by the *transformation* component. The next section places these aspects and components in the context of

visualisation design; this is followed by the description of the methods used to answer each secondary question and arrive at these components.

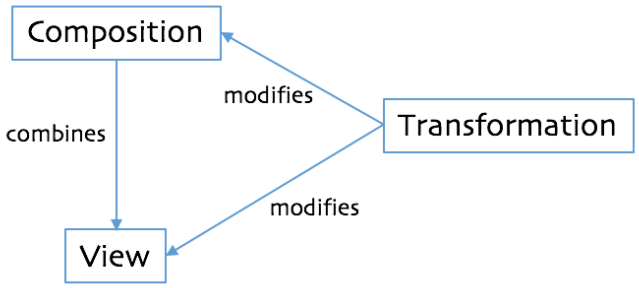


Fig. 3.1 Overview of the framework with the three components. The *View* component contains the descriptions of the visual encodings; the *Composition* component defines how visual encodings are combined in a hierarchical manner. The *Transformation* component modifies the visual encodings as well as the compositions.

3.2 Context

The questions answered in this thesis are not isolated from the process of designing and implementing an interactive visualisation, from data collection to rendering and the continuous interaction feedback from users. As a research topic on its own, the visualisation generation process has been organised and described in various models (Chi, 2000; Jankun-Kelly et al., 2007; Van Wijk, 2005). The first model to describe the computational stages of visualisation design was the *conceptual visualisation process* introduced by Haber and Mcnabb (1990), which contains three major steps: transforming *raw data* into *derived data*, mapping derived data to *abstract visualisation objects* (AVO) and rendering AVO's into a *displayable image*. As this process does not include user interactions, Card et al. (1999) adapted it into the *information visualisation reference model* (fig. 3.2), including user interactions in each step. The adaptation effectively transforms the process into a cycle, acknowledging the fact that interacting with visualisations is a dynamic process that does not end when the visualisation is rendered. However, as a model that can be instanced for various domains and applications, with different types of mappings and abstractions, it does not aim to capture the underlying ideas in the interaction cycle.

As mentioned in the previous chapter, this interaction cycle has been described by Roth (2012) and Norman (1988) in the *stages of interaction* model (fig. 2.8). Because it

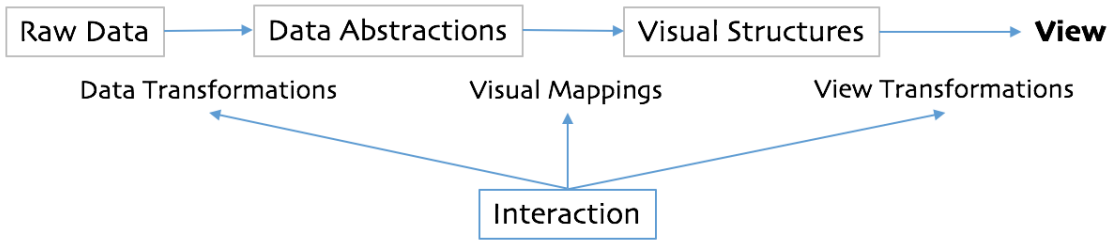


Fig. 3.2 Information visualisation reference model adapted from Card et al. (1999). The model describes the steps that are used to generate a visualisation from any dataset, with interaction allowing these steps to be revisited during execution.

is focussed on understanding the different steps that lead from a user executing an interaction to evaluating its outcome, in virtual or physical objects (named *operands*), the cycle does not account for the process of *generating* the operand. It is evident that this complement Card's pipeline, as operations used to generate *visualisations* as operands can also be part of the interaction cycle; a user can interactively trigger the same data transformations that were used to design the visualisation. Figure 3.3 shows how the steps of *specifying and executing actions* in Norman's model overlap with the three steps of Card's reference model — the merged model is referred to as *Card-Norman model* in the rest of this thesis. In the merged model, the *Visualisation* is the final result shown to the user; it may be part of an interactive system or included on a web page, as well as composed or not by multiple *views*.

The investigated research question is related to multiple steps in these models. The steps of *data transformations* and *visual mapping* relates to generating *data abstractions* from temporal data and hierarchies and the *visual structures* that correspond to these abstractions, respectively. Exploration of temporal data is related to various steps of the pipeline and the whole interaction cycle. The following sections describe how each secondary research question relates to a framework that encompasses all these steps, as shown in fig. 3.1.

3.2.1 RQ1: What are the interactive visualisation methods used for temporal data?

This research question was addressed by surveying the literature of time-oriented visualisation techniques. Three primary steps were used to answer this question: *organisation* of the literature, *description* of the visual encodings found in the organised techniques

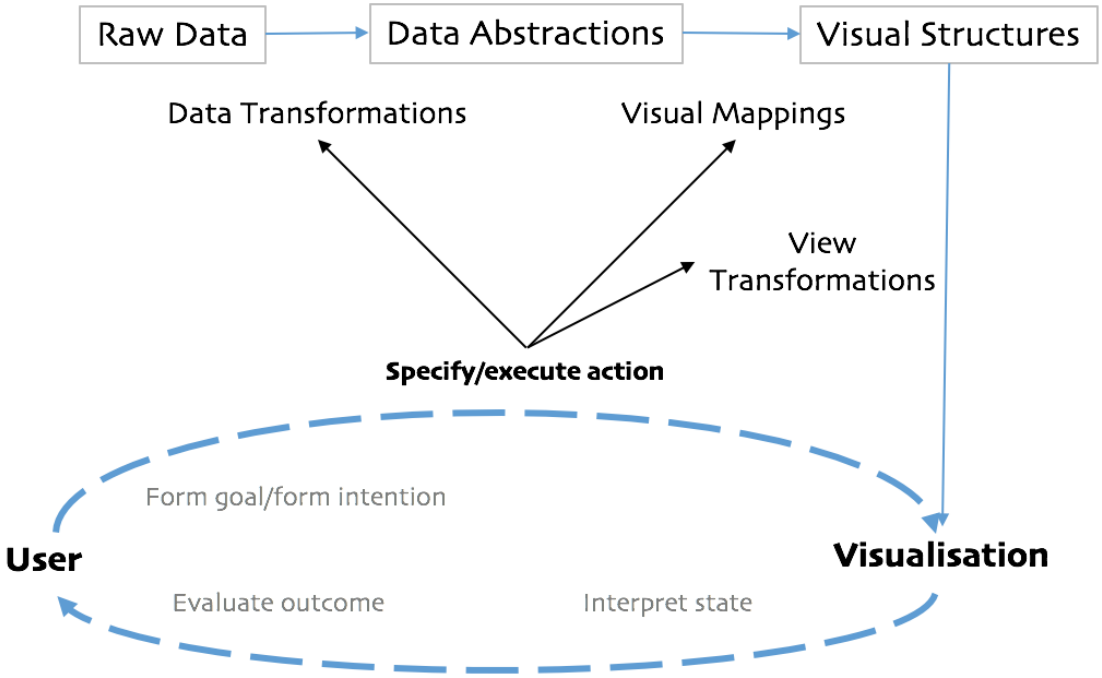


Fig. 3.3 The merged Card-Norman model. The upper part of the model describes the *design* phase of the visualisation, from which the major transformation steps can be *reused* in the *execution* phase, represented by the lower part of the model. The stage of *specify/execute action* from Norman's model is placed outside the loop and connected to the three transformation steps of Card's reference model, showing the conceptual overlap between the models.

and *generation* of visualisations through specifications that allow the *replication* the visual encodings. The generative part is used as the *view* component of the framework.

The initial search in the literature was based on the survey by Aigner et al. (2011), to avoid unnecessarily repeating efforts. The selection of entries at this stage followed the scope of the research, which means that techniques with 3D visualisations and largely based on glyphs, where possible to identify from title and abstract, were excluded. From this first collection of entries, search continued in two paths: following references and searching the primary scientific outlets of information visualisation research. The criteria for exclusion was extended by checking for repetition of *visual encodings* for common visualisation techniques, such as *bar charts*. The outlets examined were: the IEEE VIS conference, including the InfoVis track with papers published in the *Transactions in Computer Graphics and Visualisation* journal; the EuroVis conference, with articles published in the *Computer Graphics Forum* journal; the *Information Visualisation* journal and the *PacificVis* conference. Other outlets were only included as part of citation follow-ups; these include other relevant venues such as the *ACM Conference on Human Factors in Computing Systems*.

The method used to organise and describe the literature was a variation of *thematic analysis* based on a deductive approach (Fereday and Muir-Cochrane, 2006). Thematic analysis is a qualitative research method that is used to derive models or theories from qualitative data; *inductive* or *deductive* approaches can be used. Inductive thematic analysis uses emerging *codes* to categorise data items; the codes are defined by the scientist conducting the survey, and thus is typically used to analyse textual data such as interview transcripts. In contrast, a deductive approach uses predefined codes based on the theories that inform the research. In this thesis, existing concepts of temporal data and visualisation design, described in chapter 2, were used as codes to classify visualisation techniques. This is justified due to the fact that the research presented in this thesis is not intended to redefine basic concepts; additionally, the surveyed data is not unstructured text, but is derived from an analysis of scientific articles and inspection of visualisation techniques. The coding was done only by the author of the thesis.

The *organisation* of the literature was based on two criteria: *type of coordinate system* used in the visualisation and *type of time primitive* that is represented. Both emerged as characteristics that form *pre-conditions* for designing the rest of the visualisation technique: mapping any variable to a *positional channel* has different meaning depending on the type of coordinate system that is used, for example. At the

same time, it can be used to analyse the similarity between visualisations that use two different coordinate systems. The second criterion, type of time primitive, is also an aspect that limits further design choices, leading to encodings that are not compatible with each other or that cannot be explored through multiple perspectives, as seen in the different types of representation of time in chapter 2. Initially, other aspects of time were also used, such as linear or cyclic. However, as the survey went on it became clear that these were not distinguishing aspects for designing the framework.

The second step, *describing* the encodings, was done by identifying how and which visual channels are used in each surveyed technique. The main method used to do this was an adaptation *structure of information visualisation* proposed by Card and Mackinlay (1997). It involves listing the data variables that can be identified in visualisation techniques and mapping them to visual channels: position, colour, size, etc. The original method also included description of interface widgets, such as *sliders*, and basic data transformations, such as *sorting*. Another difference is the use of named variables in the original method; in this thesis, they were replaced by variable types. The details about how this method was applied in this thesis is described in the next chapter.

The final step, related to the *generation* of visualisations, was to encode the descriptions as *specifications* of layers. This step required choosing basic visual properties and valid *values* for these properties that allow the *replication* of visual encodings. The aim is to allow the use of these building blocks in hierarchical visualisations. As such, this step was fully completed by answering RQ2.

3.2.2 RQ2: How can hierarchical composition techniques be combined with the surveyed visualisations?

This question is addressed in chapter 4 by investigating how hierarchical techniques, including both the use of hierarchical data structures and the visual methods that are used to display them, can be combined with the visualisation techniques surveyed in RQ1. There was not a particular methodology that was used to arrive at the component, such as the ones used in RQ1 and RQ2, but nonetheless there is a supporting logic behind its definition. It started with an analysis of the characteristics of conditioning data and generating hierarchical structures, with the introduction of a data and visualisation model. This led to the mathematical description of the relationship between data variables and the hierarchical structures, which underpins

the combination between elements of hierarchical structures with the composition methods identified in section 2.1.3. These combinations result in the visual composition methods supported by the framework, which are described in detail in the rest of the chapter.

As mentioned, later in chapter 5, these visual composition methods are used to complete the specifications of encodings by connecting the layers. Revisiting the survey of the previous question and completing all the specifications finalises the answer to both RQ1 and RQ2.

3.2.3 RQ3: What are the temporal interactions that facilitate exploring temporal data in hierarchical visualisations?

The final secondary question is related to the gap identified in the first chapter concerning the lack of conceptual works on the interactions that are needed to explore temporal data. While the *view* and *composition* components allow designing hierarchical and composite time-oriented data visualisations, interactions are needed to complete the *Card-Norman cycle* in a visual exploration context beyond assigning visual variables. These interactions are defined as part of *transformation* component of the framework, in the form of conceptual operators that can be implemented and triggered by different interfaces and encodings. In order to derive these transformations, interaction methods from time-oriented visualisation techniques and applications were surveyed. The initial set of references used in this answer was the same that was used in RQ1. It was, however, further extended with works that were out of the scope of the first answer as there could be interactions with time visualisations that are independent from the visual encoding that is used. The process of search that was used in RQ1 was also applied in this; the following coding process was done only by the author.

The methodology used to derive the interactions started with a summarisation of these works based on the categories of interaction intents proposed by Yi et al. (2007), which was described in the previous chapter. For each entry, a short description of the interactions described in the original research articles is classified under an interaction intent. From this initial step, the interactions were narrowed down to four categories of interactions that are related to the framework: *reconfigure*, *encode*, *abstract/elaborate* and *filter*. This excluded interactions that are *view transformations* in the Card-Norman cycle, such as *explore* and *select*. At the same time, interactions that relate to non-temporal variables are also left out of the classification, as they

also include conceptual or data transformations that are not within the scope of the research.

Following the filtering step, the surveyed interactions were analysed from a temporal point-of-view: they were categorised based on the properties of time that they modify. Finally, each category was expanded into operators that receive multiple parameters and modify the time domain in various ways. Chapter 6 describes these operators in detail, comparing the possible parameterisations and how they used to define the complete transformation component. In the Card-Norman model, the *Transformation* component is equivalent to the step of *data transformations*, manipulating the *data abstractions* and only indirectly affecting the visual structures. The component is the answer to this secondary research question.

Combining the answers

The answers to these three secondary questions are combined in chapter 7, where the framework as a whole is summarised and presented as the answer to the primary research question. In that same chapter, the *interplays* between the different components of the framework are explained, as well as how they relate to the main research question and goals of this thesis. As outlined throughout this chapter, the components of the framework fit into the various steps of Card-Norman model, as depicted in fig. 3.4. The *View* component characterises the *Visualisations* that are the target of the interactions in the lower part of the diagram; in the upper part of the diagram, this component is related to the *visual mapping* step of transforming *data abstractions* into *visual structures*. The *Composition* component, through the specification of conditioning variables, is related to the *data transformations* step that transform *raw data* into *data abstractions*. The component is also related to the visual mapping step by the use of visual composition methods. Finally, the *Transformation* component contains operators that can be categorised as *data transformations* and *visual mappings* and also as *actions* from the lower part of the model.

3.3 Chapter summary

This chapter presented an overview of the the methodology used to design the framework, introducing additional related work that inform its development. It showed how the

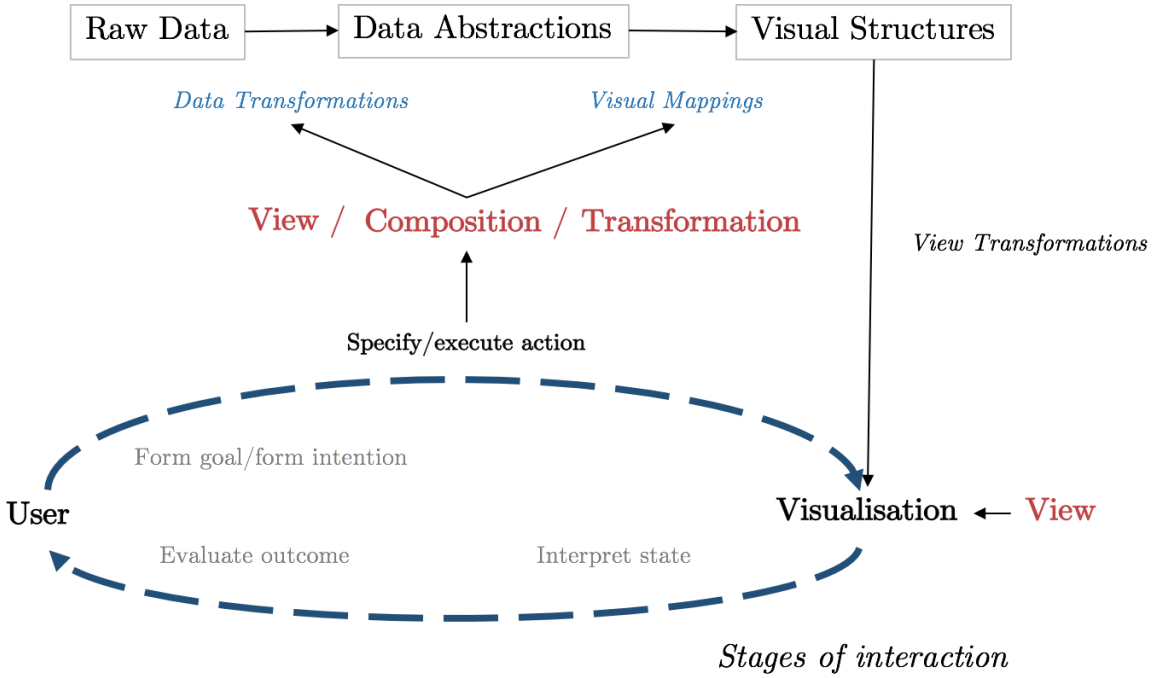


Fig. 3.4 Comparison between the framework's components – View, Composition and Transformation – with the merged Card-Norman model. The arrows from the components point at the steps that they act on. The steps highlighted in light blue are the original steps from Card's pipeline which the components are instances of.

framework is included in a visualisation design and interaction process and how each part of the framework relates to the secondary questions asked in the first chapter. Methodological and design choices were justified for each part of the framework. The next three chapters describe the components of the framework in detail, starting with the *composition component*, where the data and visualisation model that underpins the framework is also defined.

Chapter 4

Composition: conceptual model

The previous chapter introduced the framework that is presented as the answer to the primary research question, describing the methodology used to arrive at each component that form the framework. It also framed the research into the context of visualisation design and generation, relating the components to the stages of Card-Norman's model. This chapter describes the visualisation and data model that underlies the framework, enumerating the different *mappings* that exist when transforming *raw data* into *data abstractions* for use with *hierarchical visualisations*. This is expanded into the *visual composition methods* that allow the transformation from *data abstractions* into hierarchical and composite *visual structures* and serve as a basis for developing the other components of the framework. The chapter and component are introduced as part of the answer to the secondary research question – *how can hierarchical approaches be combined with time-oriented data visualisation techniques?*.

4.1 Addressing the research questions

In light of the hypothesis explored in this thesis and the research questions that drive the research, the main concepts that characterise *hierarchical visualisations* are identified; these concepts are then defined in a practical manner so that they can be *combined* with time-oriented data visualisation techniques. As these other visualisation techniques have not been described yet, this chapter uses abstract encodings for illustration purposes. Before hierarchical visualisations are described, a data and visualisation model is introduced to support them.

4.2 Data and visualisation models

The previous chapter described how the research questions and the components of the framework tie into Card-Norman's model. This section describes the *data transformation* step that generates the *data abstractions* which are then transformed into *visual structures*. In the case of the framework, the visual structures are tightly coupled with the data abstractions. Although it would be possible to define a visual model independently, such as the hierarchical aggregation model by Elmqvist and Fekete (2010), this would mean that the operators in the Transformation component would not cover the whole range of transformations of time as part of the framework. As such, this section describes an *abstraction* of a hierarchical structure; the corresponding visual model uses the structure as reference.

4.2.1 The data model

The data model used in the framework relies on the generation of a hierarchical data structure that underpins the *visual structure*. As previously mentioned, hierarchies can be implicit or explicit. Implicit hierarchies can be formed by *conditioning* data items by any variable in the dataset; explicit hierarchies are based on existing hierarchical information, such as *year* and *month*. A premise of the related *HiVE* language was the idea that certain data arrangements may be perceived as implicit hierarchies through conditioning, even though the data may not be inherently hierarchical. In the data model discussed here, the idea is to combine existing hierarchical information – the temporal granularities – with non-hierarchical information.

The hierarchical structure is generated by successively conditioning and aggregating data items by *matching* attributes of these items and mapping them into nodes corresponding to the values of attributes (see fig. 4.1). The resulting structure (see fig. 4.2) contains an abstract root, values of attributes as inner nodes and the data items as leaves. The values of inner nodes depend on the data type of the attribute; depending on the method that is used to generate the inner nodes, different types of *mappings* occur. This is determined by the use of *value-based* or *item-based* conditioning. In value-based conditioning, all the plausible values in the *domain* of an attribute are mapped to inner nodes. In this case, when no data items match that value, the leaf level will contain no nodes for the path across successive conditioning. In item-based conditioning, inner nodes only exist for the values of attributes that exist in the dataset. An example

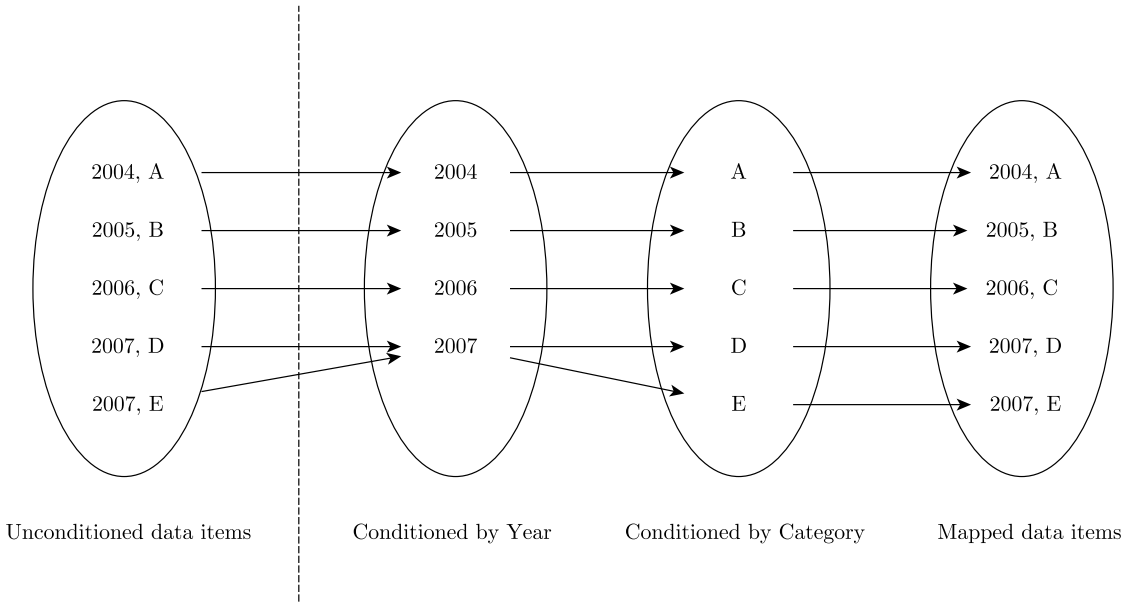


Fig. 4.1 Steps in conditioning data

is conditioning data by *month*, where the domain is the set of values for each month of the year: *January, February, March, April, May, June, July, August, September, October, November, December*. Value-based conditioning would result in 12 inner nodes; item-based conditioning results in nodes corresponding to only the months where a data item exists.

For the purposes of time-oriented data visualisation and coherence with other components of the framework, it is assumed that a value-based conditioning is used for temporal variables, when temporal data is not modified by the transformation component. This prevents the occurrence of temporal gaps in any temporal conditioning; however, it requires tackling the problem of adjusting either the data or the visual representation appropriately by, for example, *imputing* missing quantitative data when an attribute is not recorded for a certain time. Gschwandtner et al. (2012) examined this in a taxonomy of *dirty* time-oriented data, referring to it as *missing tuples*. However, this is not necessarily a *problem*, as it might happen for a variety of reasons, from the data collection method to characteristics of the data, such as an event only occurring at a specific time. Because particular contexts require different solutions to *resolve* missing tuples, the framework does not provide solutions for handling the problem and leaves the best solution to be decided in the implementation.

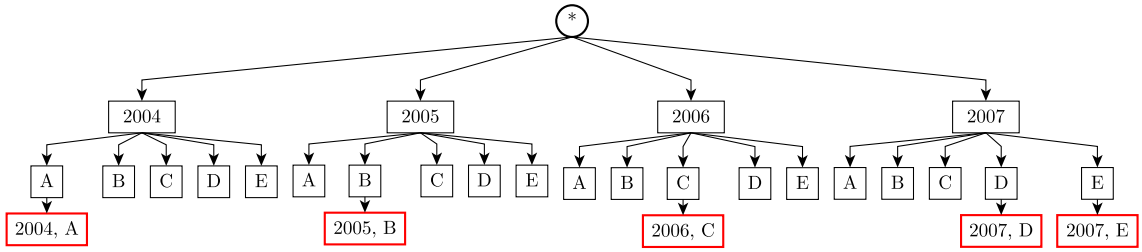


Fig. 4.2 Example of a hierarchical structure. The node at the top represent an abstract root that *aggregates* all data items. Subsequent levels condition data items by *year* and a categorical attribute. The leaf level contains the un-rendered data items, mapped by their conditioning path.

4.2.2 The visual model

The inner nodes in the resulting hierarchical structure correspond to individual *visual marks* that will be rendered. For each level, one or more configuration of encodings sets the rules for the placement (layout) and appearance of these marks. The relationship between the levels and nodes is defined through the component described in the rest of this chapter; the definition of encodings is covered in chapter 5.

The hierarchical structure is processed from the root node, which, as an abstract node, refers to the whole visual space available for rendering. The actual rendering in each subsequent level, until the leaves, is defined by the encodings and the visual composition methods, as well as the specification of a conditioning variable. The conditioning variable for each level defines the number of nodes, as well as the relationship between nodes. Although individual data items are placed at the bottom of the tree, the visual representation of inner nodes can be based on data attributes derived from subsequent levels or the data items that are conditioned by each node.

Figure 4.3 shows an example of items with a temporal reference and a quantitative value, conditioned by Year. The star node represents the visual space; the second level contains the four years that are present in the data and are mapped to inner nodes. The dotted line indicates the conditioning path. A representation of this hierarchical structure contains four visual marks; properties of the visual marks will be derived from the associated data items: in the case of 2004, for instance, the value is derived from any expression relating 10 and 15: minimum or maximum value, sum, average or others. The visual representation of the tree is shown in fig. 4.4.

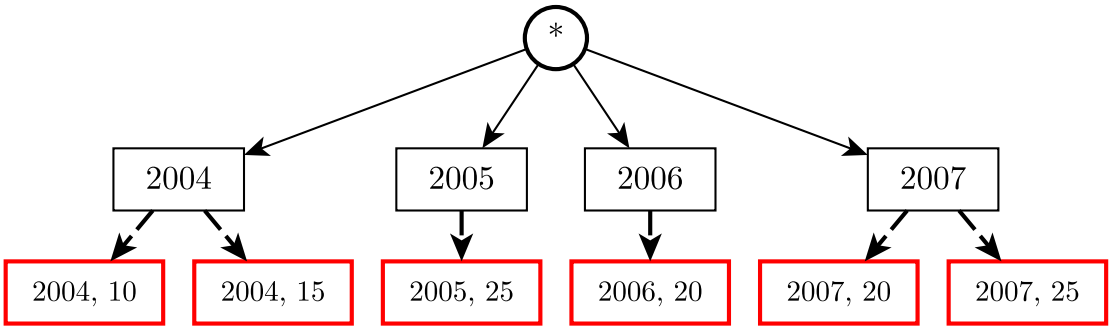


Fig. 4.3 Example of conditioning with multiple items per node.

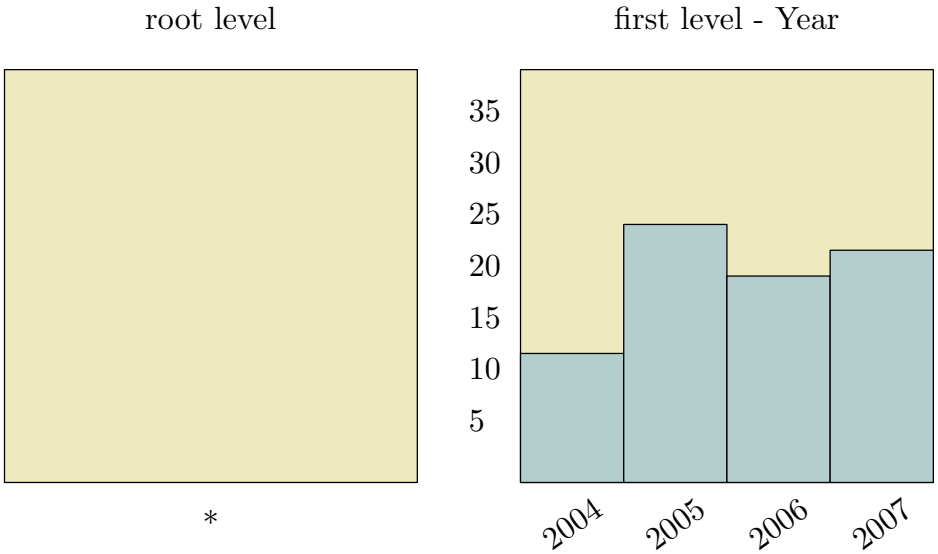


Fig. 4.4 Example of visualisation of tree in fig. 4.3. The left side represents the abstract root node corresponding to the whole visual space (in yellow). On the right side, each bar in blue corresponds to the inner nodes of the tree; the size of the bars from 2005 to 2008 represent the average of the two data items that exist for those two years respectively.

4.3 Types of mapping

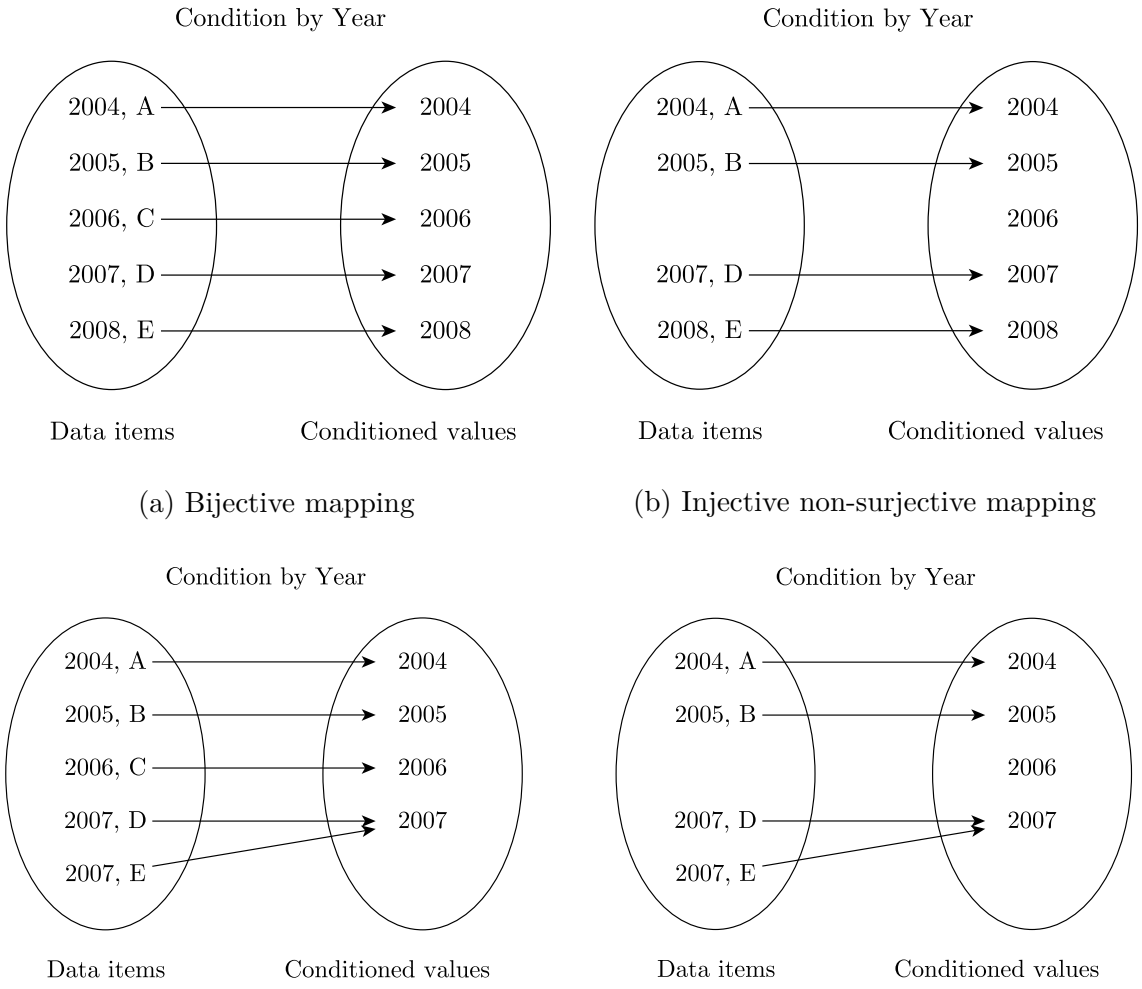
The results of the conditioning process depends on the characteristics of the dataset and the use of item or value-based conditioning. This section describes four cases of *mapping* from one level to another, based on properties of the data items to be conditioned and the resulting conditioned variables; the mappings are described following the ones used in set theory (Lawvere and Rosebrugh, 2003). The mappings influence how the framework can be used, by limiting the use of compositions or encodings and can be used to guide design choices.

Bijective mapping This mapping happens when the number of conditioned values is equal to the original number of items to be conditioned, as shown in fig. 4.5a. With bijective mappings, it is impossible add further levels to the hierarchy, as each internal node matches exactly one data item from the original data.

Injective non-surjective mapping This mapping can happen based on the assumption made about filling temporal gaps. In this case, certain values from the variable used for conditioning have no data items mapped to them. Figure 4.5b shows an example where one of the years has no corresponding data items: that year is still used as a value in order to prevent the temporal gap. In this example, the number of conditioned values is greater than the number of items in the left side of the mapping. As is the case with a bijective mapping, it is impossible to add further levels to the hierarchy.

Non-injective non-surjective mapping A type of mapping (fig. 4.5c) that contains temporal gaps and more than one data item is conditioned by the same value: that value is represented by a node that *aggregates* the data items. This is a common case when a multivariate dataset is used and multiple measurements are taken at the same time, for example. The actual relationship between the number of data items and the number of values is indeterminate.

Non-injective surjective mapping The final possibility (fig. 4.5d) is when there are no temporal gaps, as every temporal reference contains at least one data point, and there are multiple values or objets per conditioned value. In this case, the number of data items is greater than the number of conditioned values.



(a) Bijective mapping

(b) Injective non-surjective mapping

(c) Non-injective surjective mapping

(d) Non-injective non-surjective mapping

Fig. 4.5 Types of mapping

Table 4.1 summarises these mappings in terms of the relationship between the incoming data items and the conditioned values. The types of mapping are not used as parameters or properties in the framework; however, they should be part of any implementation of the framework to guide the manipulation of hierarchies and definition of encodings. Some of the encodings covered in chapter 5, in fact, prevent further conditioning due to the shapes used for visual marks. Data-driven guides that use these types of mapping should be used by designers to help users.

Type of mapping	Surjective	Non-surjective
Injective	$D \rightarrowtail\!\!\!\!\!\rightarrow C, \lvert D\rvert = \lvert C\rvert$	$D \rightarrowtail C, \lvert D\rvert \leqslant \lvert C\rvert$
Non-injective	$D \twoheadrightarrow C, \lvert D\rvert > \lvert C\rvert$	$D \rightarrow C, \lvert D\rvert ? \lvert C\rvert$

Table 4.1 Table of types of mappings. The table shows the different cardinalities resulting from each type of mapping. The bottom right cell contains a question mark representing the dependency of the relation between D and C on the characteristics of D. The arrows used are conventional mathematical notation

4.4 Hierarchical relationships

As described in chapter 2, Javed and Elmqvist (2012) suggested five basic methods for composing visualisations: *juxtaposition*, *integration superimposition*, *overloading* and *nesting*. However, two of these can be considered as subsets of others. *Integration* is a variation of juxtaposition with explicit linking between views and can be expressed through visual mappings, as will be described in chapter 5. *Overloading* is a result of mixed visual channel mappings: more than one set of *visual marks* can be specified using the same *coordinate system* and positional variables — this method, however, is not within the scope of the framework. The three remaining methods – juxtaposition, superimposition and nesting – are included in the framework as the primary composition methods.

Given the different types of mapping and the visual composition methods, the last important consideration for this component is the *scope* where the visual composition is applied. In addition to the composition methods, chapter 2 described the combinations of those with dataset arrangements: using one or more view specifications for one subset, disjoint subsets or the whole data. In the context of hierarchies, these combinations are expressed through the hierarchical levels and nodes covered in the earlier sections of this chapter; this results in the types of hierarchical relationships for composition illustrated in fig. 4.6. The red arrows indicate the scope of the composition, assuming that at least one view is specified for the whole hierarchy. *Within level* corresponds to *disjoint subsets*, *same level* corresponds to *whole data* and *between levels* partially corresponds to *one subset*. This results in the following possibilities regarding the use of the encodings: different encodings for the same level, same or different encodings between levels and same or different encodings within level (the configuration of same encoding for the same level is redundant).

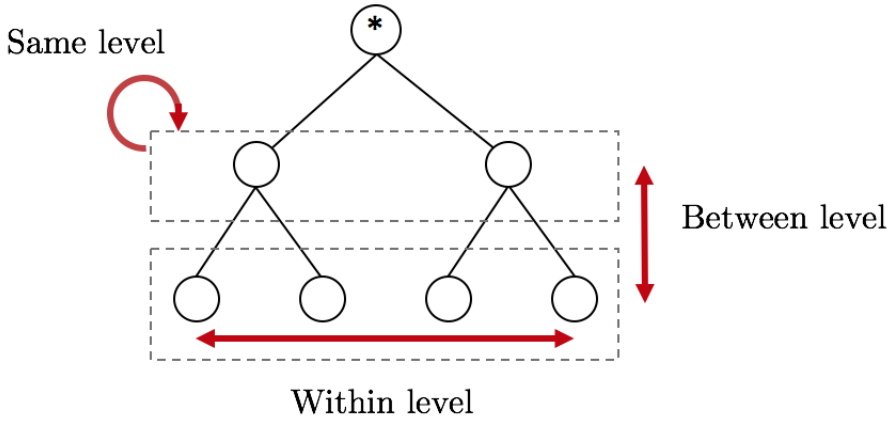


Fig. 4.6 Types of hierarchical relationships for composition

	Juxtaposing	Superimposing	Nesting
Same level	Different encoding	Different encoding	-
Between level	Same encoding Different encoding	Same encoding Different encoding	Same encoding Different encoding
Within level	Same encoding* Different encoding*	Same encoding* Different encoding*	- -

Table 4.2 Table of hierarchical composition methods. Methods marked with a star are defined in the *View* component.

One further subdivision can be identified in the diagram. While *within level* relationship happens between inner nodes, *same level* and *between level* relationships operate independently of individual nodes, in the scope of hierarchical levels. This is explained by the fact that *within level* is defined for the nodes representing the values of a conditioning variable, while the other compositions operate on the hierarchical structure itself. For that reason, *within level* compositions are defined by the specification of visual encodings as part of the *view* component (see chapter 5). The composition component contains explicit composition methods only for *same level* and *between level* compositions.

4.5 Hierarchical composition methods

The composition component allows the specification of 8 hierarchical composition methods, as shown in table 4.2, integrating two types of hierarchical composition, three visual composition methods and two variations of encodings. This section describes how each visual composition works with the visualisation and data models and the design decisions and parameters that are required for practical use of the framework; it only covers each individual composition rather than combinations of compositions. The combinations of compositions which allow the exploration of the resulting design space are described in chapter 7. Additionally, the specifications of the visual encodings are omitted in this chapter as they are not necessary for describing the compositions. Each method is accompanied by a signature (see chapter B for a full description of the grammar) that describe a configuration of composition method, conditioning variable and an unspecified encoding (e.g. E_1 for a random encoding).

4.5.1 Same level composition

View compositions on the same level are arrangements of views based on the same data items. Although they may have different visual characteristics, the visual marks in each correspond exactly to the same data item from the hierarchy. This type of composition is similar to general non-hierarchical composition, as there are no exclusive *characteristics* of hierarchies that affect it. There are two plausible ways of composing such views: juxtaposing and superimposing, both with different encodings, as using the same encoding provides exactly the same information. Some authors (Munzner, 2014) refer to this composition as applying different views to the same *subset* of a dataset.

Juxtaposing same level with different encodings

Signature: $J(E_1,E_2)$

This type of composition (fig. 4.7) is a thoroughly explored topic in information visualisation research as the *multiple views* or *coordinated multiple views* paradigm. Wang Baldonado et al. (2000) conceptualised *multiple views* and proposed guidelines in the form of *rules* to follow when designing visualisations as multiple views, such as considerations for designing consistent views, the number of multiple views and the complementing nature of multiple views. The last point is intrinsically related to

exploratory analysis — the juxtaposition of different encodings provides different perspectives simultaneously, which is not possible with a single view.

Juxtaposing allows the use of *explicit links* to connect visual marks between views, which is one of the options for *coordinating* views. When explicit links are not used, at least one visual channel in each view is generally used as a *shared encoding* to allow users to identify marks representing the same object in both views, for example. In time-oriented views, a common choice is to vary one of the positional variables with different quantitative attributes, while the other variable is fixed to time. This is the case in LiveRAC (McLachlan et al., 2008), where multiple line charts are juxtaposed displaying variation of attributes over time. It is also possible to use a combination of explicit linking with a shared encoding.

Juxtaposed views must be spatially arrange appropriately, including deciding the position of each view and the portion of the available space for view. For example, aligning charts vertically, such as those in LiveRAC, where the horizontal axis is the shared encoding, facilitates comparison between charts.

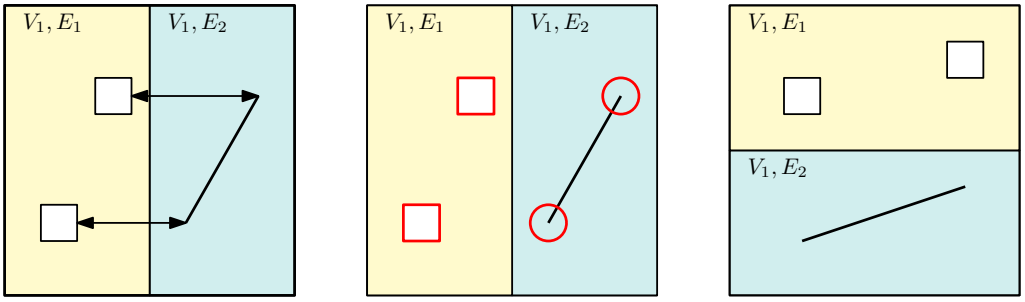
The design decisions for this type of composition are the following (the choices are not mutually exclusive):

- Explicit linking: how to connect visual marks;
- Shared encoding: which encoding is shared between views;
- Spatial arrangement: position and size of views;

Superimposing same level with different encodings

Signature: $S(E_1,E_2)$

This composition method involves using two configurations of encodings to highlight different aspects of the data, *layering* the encodings over each other in a *single* view (fig. 4.8). For temporal data, one example is *dual axis* charts, where two different attributes are used to position marks in the vertical axis, emphasising simultaneous variation of attributes for a single entity (object or event) over time. Similar to juxtaposition, a choice of a *shared encoding* can be used to facilitate comparison between the views. However, this composition method does not impose any conditions on the use of common visual channels — the configuration of encodings can be entirely different, which evidently has consequences on the *legibility* of the graphics.



(a) (b) (c)

Fig. 4.7 Example of same level juxtaposition with different encoding: (a) shows the use of *explicit linking* connecting visual marks in each view; (b) represents the used of a shared encoding, a red outline, to identify marks in both views; (c) displays a variation of spatial arrangement

Javed and Elmqvist emphasised that superimposition might result in *visual clutter* depending on the size of the dataset and the choice of encodings. Additionally, visual marks might be occluded by other layers depending on the the *order* in which encodings are layered. The design decisions for this type of composition are the following:

- Shared encoding
- Layering order

4.5.2 Between level composition

Composition between levels allows various combinations of encodings to be defined through the exploration of the hierarchical relationship between levels. In this category, using the *same* or a *different* encoding is possible, as well as using the three visual composition methods: juxtaposing, superimposing and nesting. As mentioned before, while the *HiVE* language did not prescribe one method for composition, most of the examples were discussed using *nesting*.

The template *signature* for this composition is the following: $C((V_1,\mathrm{E}),(V_2,\mathrm{E}))$, where C is the visual composition method and E represents a *view specification*, which can be the same or different for the conditioning variables V_1 and V_2. For between level composition to be valid, the number of items from conditioning with V_2 must be necessarily greater than V_1.

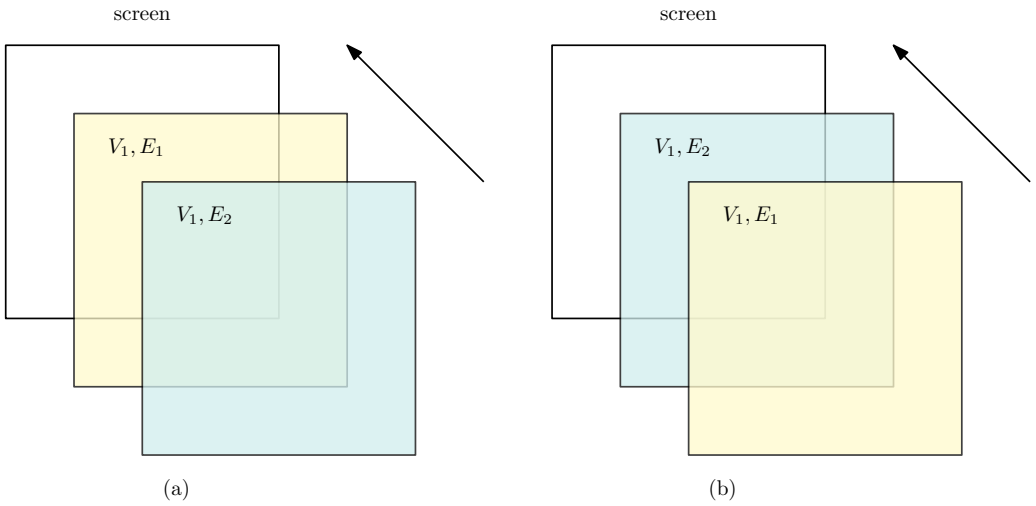


Fig. 4.8 Example of same level superimposition with different encoding

Juxtaposing between level with same or different encoding

Signature: $J((V_1, E), (V_2, E))$

This method arranges spatially isolated views of adjacent hierarchical levels using the same or different view specification (fig. 4.9). This category shares the same basic design choices of juxtaposing on the same level: choices of explicit linking or shared encodings for identifying entities in both views and the spatial arrangement of the juxtaposed views. Juxtaposing *different* encodings is conceptually equal.

The following design decisions are related to this type of composition:

- Explicit linking
- Shared encoding
- Spatial arrangement of juxtaposed views
- Spatial arrangement of second level

Superimposing between levels with same or different encoding

Signature: $S((V_1, E), (V_2, E))$

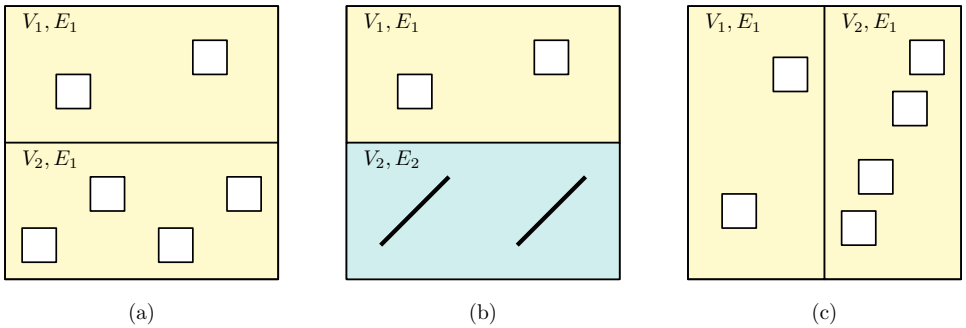


Fig. 4.9 Example of between level juxtaposition with (a) same encoding, (b) different encoding, (c) spatial arrangement variation.

Compared to the previous methods, superimposing between levels (fig. 4.10) allows for more complex arrangements of visualisations. The basic idea of this composition method is to display visual marks for every level in the hierarchy layered over each other, with the same or different encodings, in a single *shared* visual space. As visual marks might overlap each other, the encodings for each level of the hierarchy must be carefully designed. This configuration is particularly useful to contrast attributes of aggregated sets with lower hierarchical levels: individual attributes of items in the lower level can be superimposed over a summarised attribute derived from the aggregated set.

Due to the parent-child relationship between levels, it is possible to use explicit linking with this type of method as well, though it may add a significant amount of clutter. Shared encoding is again an alternative; in this case, the *layout* may additionally be used as part of shared encoding to indicate the relationship between data items between levels.

Nesting between levels with same or different encoding

Signature: $N((V_1, E), (V_2, E))$

As described in chapter 2, nesting (fig. 4.11) is one of the basic composition methods used in *implicit hierarchy visualisation*. The fundamental idea of nesting is to display marks for lower hierarchical levels *within* visual spaces defined by the parent levels. The spatial arrangement in this case is inherited from the parent level, not requiring additional design decisions at composition level. This means, however, that the lowest

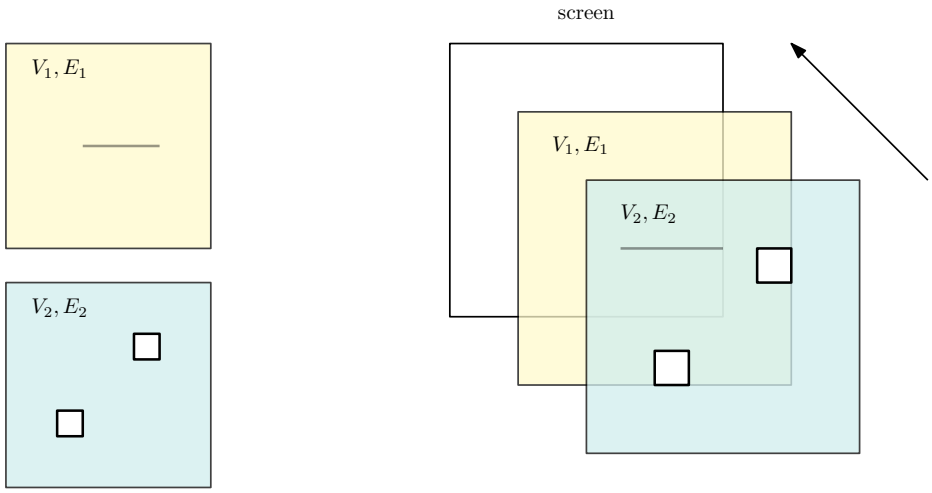


Fig. 4.10 Example of between level superimposition with (a) different encoding, (b) different encoding and explicit linking.

visible level is the only level that can be represented by marks of *any* shape, other inner nodes can only be represented by shapes that allow the nesting to occur, such as area-based shapes.

As the previous composition methods, nesting with same or different encodings are conceptually equal. The implications for the visualisation itself, however, are very different. Nesting visualisations using the same rendering rules results in a consistent visualisation, though not necessarily effective. Mixing encodings through nesting, however, requires more careful design, starting with the coordinate system. Using polar-based visualisations as non-leaf nodes will result in a smaller area for cartesian-based visualisations nested within them; with cartesian-based visualisations, the opposite is true: the area reserved for the polar-based techniques is larger compared to the other configuration.

The other implication of nesting is the size of each visualisation displayed. When partitioning the visual space for each level in the hierarchy, lower levels have a smaller space compared to upper levels. This can be mitigated with appropriate layouts that map data attributes to the size of the shapes, but some visualisation techniques that rely on large visual spaces might be less effective.

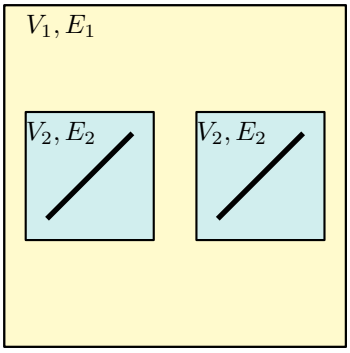


Fig. 4.11 Example of between level nesting. The characteristics of the regions where encoding E_2 is rendered are part of the specifications of encoding E_1.

4.6 Framework context and definitions

As highlighted in the beginning of the chapter, the composition component encompasses two steps in Card-Norman's model. The **conditioning variable** defines the *data transformations* from raw data to *data abstractions*; it also *sets up* the structure that will be used in the *visual mapping* step: the **composition** method then defines the *visual relationship* between the nodes in the hierarchy. The use of conditioning variables and composition methods is part of the answer to a secondary research question: *how can hierarchical approaches be combined with time-oriented visualisation techniques?* In the following chapter, the remaining composition method, *within level*, will be described and the techniques will be introduced as part of the *view* component.

Additionally, the notation for composition can be used to guide implementations of the framework to support the basic interactive transformations of the hierarchy. The following transformations covers the basic cases of adding new levels to the hierarchy, modifying and removing existing levels and also changing a set of encodings.

- **Add conditioning variables** and associated encoding;
- **Swap** conditioning variables and associated encodings;
- **Remove** a conditioning variable and associated encoding;
- **Add different encoding** (or remove) for same level compositions;

- **Change** the composition method.

4.7 Discussion

There are two important points of discussion regarding the visual composition methods. The first is the inclusion of *within level* composition as part of specifications in the *view* component, requiring designers to separate the definition of this type of composition from normal mappings to visual channels. One of the reasons for this is due to the fact that the *view* component handles everything that is defined based on the relationship between *nodes* within the same level, whereas the methods in the composition component are based on the relationships between the *levels* in the hierarchy. One alternative would be to have every composition in the same component; this would require, however, considering how to interpret specifications of encodings when within level composition is used. Currently, the *interpretation* of encodings is consistent for same or between level composition. For example, if *juxtapose* could be applied *within level* through the composition component, the *layout* property of encodings would have to be interpreted differently. This alternative would increase the *conceptual* dependency between components and thus increase the ambiguity of the specifications.

The second point of discussion is the design decisions associated with the composition methods. In the current form, the framework does not expose the properties of spatial arrangements for the compositions, so a description of a visualisation with the framework does not contain information about how to order juxtaposed views on the same level. This decision was made to reduce the scope of the framework in this research, which can be expanded in future work. Among existing visualisation grammars and languages which should be considered in future work, *Vega-Lite* (Satyanarayan et al., 2016) has a mix of *predefined* and *parameterised* composition methods: the predefined methods contain *embedded* spatial arrangements by definition, such as *vconcat* for vertical juxtaposition, whereas parameterised methods such as *repeat* have the *order* of juxtaposed and different encodings also as part of the method definition. Contrasting with this choice of definitions, which were made for practical reasons, it is important to keep a *consistent* definition of properties of compositions in a future extension of the framework presented in this thesis.

4.8 Chapter summary

This chapter introduced the data and visualisation model that are used to describe the visual composition methods supported by the framework as part of the composition component, including the conceptual descriptions and the *design decisions* that affect implementations of the methods. The two main concepts that define this component were introduced as part of the answer to a secondary research question – *how can hierarchical approaches be combined with time-oriented visualisation techniques?*. The next chapter explores the *visual encodings* that describe time-oriented visualisation techniques, completing the answer to this question and addressing the following question: *which interactive visualisation methods are used to explore temporal data?*.

Chapter 5

View: visual encodings

This chapter introduces the *view* component of the framework, which refers to the visual encodings used to display temporal data for visual exploration. This refers to the specification of the views that are used with the composition methods described in the previous chapter, resulting in hierarchical visualisations. This component is designed in three steps following a survey of interactive visualisation techniques for time-oriented data: organising the literature, extracting the necessary information and translating it into layered *view specifications*. The following sections describe the results of these steps.

5.1 Addressing the research questions

Two of the secondary questions are addressed in this chapter. RQ1 – *what are the interactive visualisation methods used for temporal data?* – relates to the *visual encodings* that are used in time-oriented data visualisations. The answer to this question is found through a three-step process as described in chapter 3 and complemented by the composition methods established in chapter 4. This chapter also addresses RQ2 – *how can hierarchical approaches be combined with the surveyed visualisations?* – completing the description of *within level* compositions that were not covered in the previous chapter.

	Type of time primitive			
Coordinate system	Instant	Both	Interval	Total by coordinate
Cartesian	23	2	4	29
Polar	10	-	2	12
Total by type	33	2	6	41

Table 5.1 Distribution of visualisation techniques and systems over the two classification criteria. The table shows a predominance of techniques showing instant time compared to both types and interval time, the majority through cartesian coordinates.

5.2 Organisation of the literature

The two criteria used to organise the literature are *coordinate system* and *type of time primitive*. They were combined to classify the 41 entries in the survey, resulting in the distribution presented in table 5.1. The distribution shows that research is largely focused on developing techniques based on cartesian coordinates to display time as *instants*.

5.3 Description of the literature

5.3.1 Types of variables and visual channels

The second part of the survey involves describing the organised techniques. This was done by identifying the visual channels to which variables are mapped. This requires deciding how to identify the variables; as mentioned previously, Card and Mackinlay (1997) used domain named variables to describe visualisations; in this research, it is important to be able to compare encodings without regarding for domain-specific characteristics. Because of this, instead of named variables, four *types* of variables are used:

- Space: spatial variables includes geographic coordinates and also non-geographic projections resulting from algorithms such as multidimensional scaling (MDS);
- Time: temporal variables include any granularity used to *reference time*;

- Attributes: attributes are any quantitative measurement or categorical variables that depends on the other three types. These generally represent measurements collected over time or any other type of data that changes over time or space;
- Identifiers: these are variables that *identify* objects, events or processes. The underlying *data type* can be numerical or textual, such as name of regions.

This classification is largely inspired by the triad framework by Peuquet (1994), the separation between sets of *references* and *characteristics* by Andrienko and Andrienko (2006) and the generic *sets* identified by Andrienko et al. (2011). Peuquet suggested a conceptual framework for spatiotemporal data based on object-based representation, location-based representation and time-based representation. These concepts match the *identifiers*, *space* and *time* categories, respectively. Andrienko and Andrienko used *references* to describe the unique identifiers for objects, with *characteristics* being the measured or observed attributes. Additionally, Andrienko et al. introduced the set of *thematic attributes*, which are all the characteristics of objects that do not involve space and time. Both match *attributes* used in this thesis.

As the visualisation techniques are categorised as displaying instants, intervals or both, it is also important to distinguish between these occurrences in time variables. In chapter 2, it was discussed that it is possible to model interval data through arrangements of a starting time point, an ending time point and the *unanchored* duration in-between. Time points *inside* the interval can also be *projected* from the interval, such as the middle point of that interval; to visualise interval data this is an essential operation. For this reason, in specifications of visualisations that display interval data, time variables are defined with reference to such properties. For instance, if the duration of the interval is mapped to the size (height, width, radius or angular length) of a shape, it is referred to as *time:duration*. Basic modifiers are *start*, *end*, *duration* and *midpoint*, but encodings can also be based on other time points within the interval.

Specifying how these variables are used requires choosing which visual variables will be used. As the aim is to specify visual encodings for *time-oriented* visualisations, this means that some channels do not uniquely characterise visualisations as such. For example, textual annotations are ubiquitous for any type of visualisation. As a starting point for the survey, *position*, *size* and *colour* were used. Additionally, a *connecting path* was used to indicate when a *line* is used to connect visual marks.

5.3.2 Layouts and shapes

Besides deciding how attributes are mapped to visual channels to characterise visual marks, it is necessary to consider how the visual marks are distributed in the visual space and how they occupy it, or the different *layouts* and *shapes* that are used. While this is a general aspect of visualisations, there are design choices appropriate for time-oriented visualisations. In chapter 2, it was described authors such as Brehmer et al. (2017) define a small set of layouts for visualising time; others, such as Wilkinson (2005), define arrangement rules as parameters for *positioning* visual marks. Yet another approach is the one taken by Wickham and Hofmann (2011), who encoded the use of space along with the specification of shapes: *bars* do not occupy the whole space, while *spines* are space-filling shapes.

To characterise existing visualisations and define how the framework handles the different layouts and shapes used for time visualisation, it is first necessary to describe the different methods used to generate positions for one or two axes, which corresponds to the *visual mapping* step in the Card-Norman's model. This can be done via scaling and translating functions or by algorithms that produce positions based on more complex rules. Based on the input of one or more variables and the output of one or two dimensions, there are four possible cases:

1-to-1 One variable to one dimension: a set of values from a single variable is mapped to one visual dimension (see fig. 5.1);

1-to-2 One variable to two dimensions: a set of values from a single variable is mapped to two visual dimensions:

n-to-1 Two or more variables to one dimension: sets of values from two or more variables are mapped to one visual dimension;

n-to-2 Two or more variables to two dimensions: sets of values from two or more variables are mapped to two visual dimensions;

For time visualisation, this categorisation helps to determine the *dimensionality* of layouts and whether one or more temporal variables are used together or independently from each other. For example, a *spiral* layout depends on two temporal parameters: the number of points in one arm of the spiral and the number of arms, where an arm is a full 360 degrees rotation. At the same time, the formula used to calculate a position along the spiral results in two positions that depend on each other – the angle *theta*

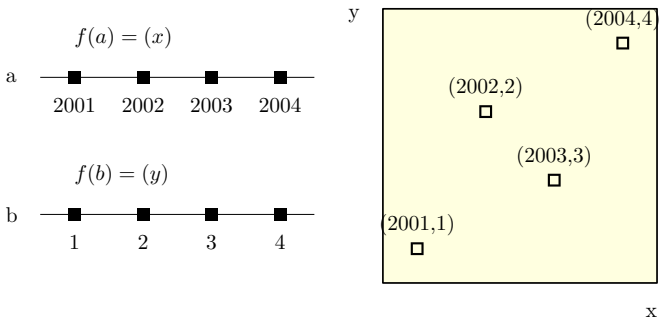


Fig. 5.1 Examples of 1-to-1 layout. In the figure, functions f and g map the two variables a and b independently to the two visual dimensions available. This case covers most cases of scale transformations.

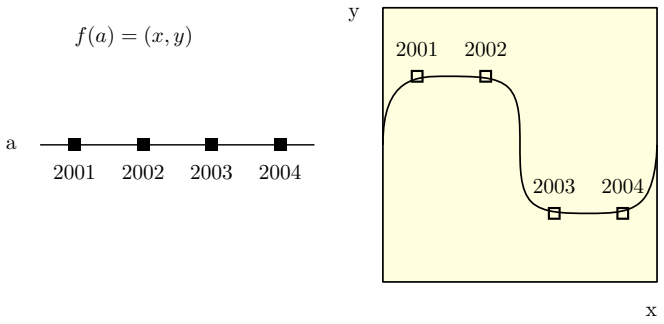


Fig. 5.2 Examples of 1-to-2 layout. In the figure, function f maps one variable to the two visual dimensions x and y. This case covers special cases such as the use of trigonometric functions.

depends on the radius r and vice-versa. This places the *spiral* layout in the fourth case of layouts.

Visualisations that combine time in one dimension with other attributes, such as line charts, are generally of the first type, where time is mapped to a single axis. Although this categorisation is not strict – as a temporal variable can be defined as a combination of multiple temporal scales, such as year and month, or a single *year-month* variable – for the purposes of describing visualisations techniques and translating them into the framework, each granularity is treated as a different variable. Assigning layouts to categories helps to identify how many positional variables need to be modified in order to transition to a different layout, also allowing for grouping visualisations that belong to the same category of layout.

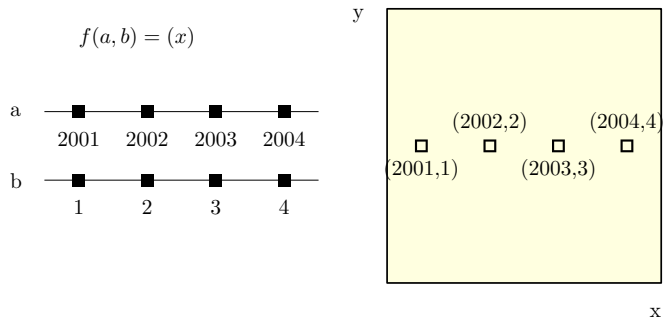


Fig. 5.3 Examples of n-to-1 layout. In the figure, function f maps two variables to a single visual dimension x; in the example, y remains constant. This case covers dimensionality reductions where multiple attributes are used to derive a position.

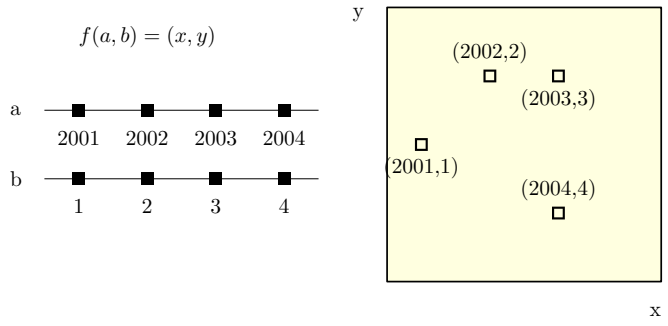


Fig. 5.4 Examples of n-to-2 layout. In the figure, function f maps two variables to the two visual dimensions x and y. This case covers layouts such as spiral formulas.

This leads to identified the different layouts that are used in visualisations. The aim of the view component is to allow unexplored layouts to be defined, however, it is also important to include existing layouts in a distinguishable manner. The choice for the framework is to specify names of formulas or algorithms when they are essential to characterise the visualisation. Examples of this are the aforementioned *spiral* and *multidimensional scaling*. For general cases, the keywords *SCALE* and *ALGORITHM* are used to classify the primary layout mechanisms. Scale-based layouts covers *measurement-to-pixel* transformations – converting from *nominal* or *ordinal* scales to the screen through various type of scales, from linear to exponential transformations. Algorithm-based layouts include *treemap* layouts, such as the one used in *ClockMaps* (Fischer et al., 2012), or the stacking algorithm used in *ThemeRiver* (Havre et al., 2002).

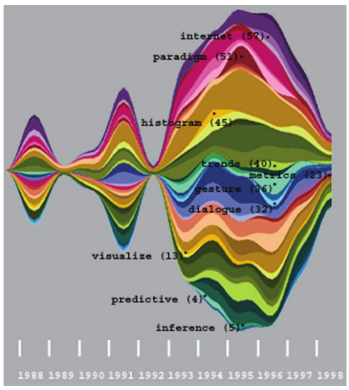


Fig. 5.5 ThemeRiver (Havre et al., 2002)

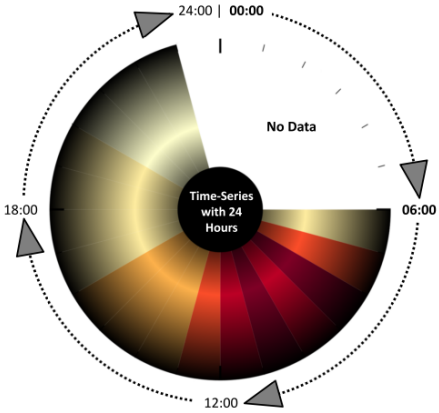


Fig. 5.6 ClockMap (Fischer et al., 2012)

Visual marks are also represented by a number of different shapes. For hierarchical and time visualisations, the shape type impacts the type of time that can be depicted and whether or not some visualisations can be nested, which is based on how some shapes used the available visual space. In this thesis, the following shapes are used: *point*, *bar*, *spine*, *ellipse* and *polygons*, summarised in fig. 5.7. These types of shapes are based on the Grammar of Graphics (Wilkinson, 2005) and the basic primitives defined by Wickham and Hofmann (2011). A point is the basic representation of a visual mark on the screen with no associated width or height. Bars and spines are used according to the definition of by Wickham and Hofmann (2011): bars divide the space along the width, with a data attribute mapped to the height (or with alternate dimensions); spines divide the space horizontally based on a data attribute or with equal partitioning and space-filling on the other direction (or with alternate dimensions). Both shapes can generally be used with nesting composition. Ellipses are the corresponding shapes for points where it is possible to use nesting, as they have an associated area. Lastly,

polygons classify various shapes that result from the use of certain algorithms as layouts and thus do not fit into any other category.

The use of bars and spines acknowledge the difference between the use of space: bars can be partially space-filling, while spines are fully space-filling. Polygon generalises area-based non-rectangular shapes and are also applicable for geographic data, such as maps with varying projection. Points are useful to represent instants and projections of intervals (such as the *midpoint* of an interval); bars can be associated with instants to represent other types of attributes and can also be used to represent anchored intervals. Polygons are usually a result of a combination of variables, including instants.

For visual composition, spines are appropriate for nesting visualisations; point-based visualisations are unsuitable for nesting and bars are suitable depending on their width (or height, depending on orientation). Polygons are generally unsuitable as their shape cannot be guaranteed to be rectangular. The suitability of ellipses also depends on the encoding used to fill them: generally, encodings based on polar coordinates can be nested. For the framework, the underlying data is a primary factor that prevents guidelines to be proposed regarding the suitability of these shapes: however, the aspects discussed here are about the shapes in relation to each other and a implementation of the framework as an exploratory tool. When designing an application for geographical data, for example, it is likely that it will be possible to nest regions within each other using polygons as shapes.

5.4 View specifications

The final step to address RQ1 and complete the component of the framework is translating the descriptions of the visual encodings into specifiable properties and combine them with the visual composition methods. This is done by *decomposing* encodings into *layers* that are described by a conditioning variable, a composition method and a set of visual mappings, including layout and shape. The aim of using the layers is to enable rearrangement of visual encodings based on the use of conditioning variables and the sets of visual mappings. The combination of layers and composition methods describe *complete* visualisation techniques; from this, common layers can be found among techniques and rearranged with the framework, enabling a navigation of the resulting design space and exploration of different perspectives on datasets.

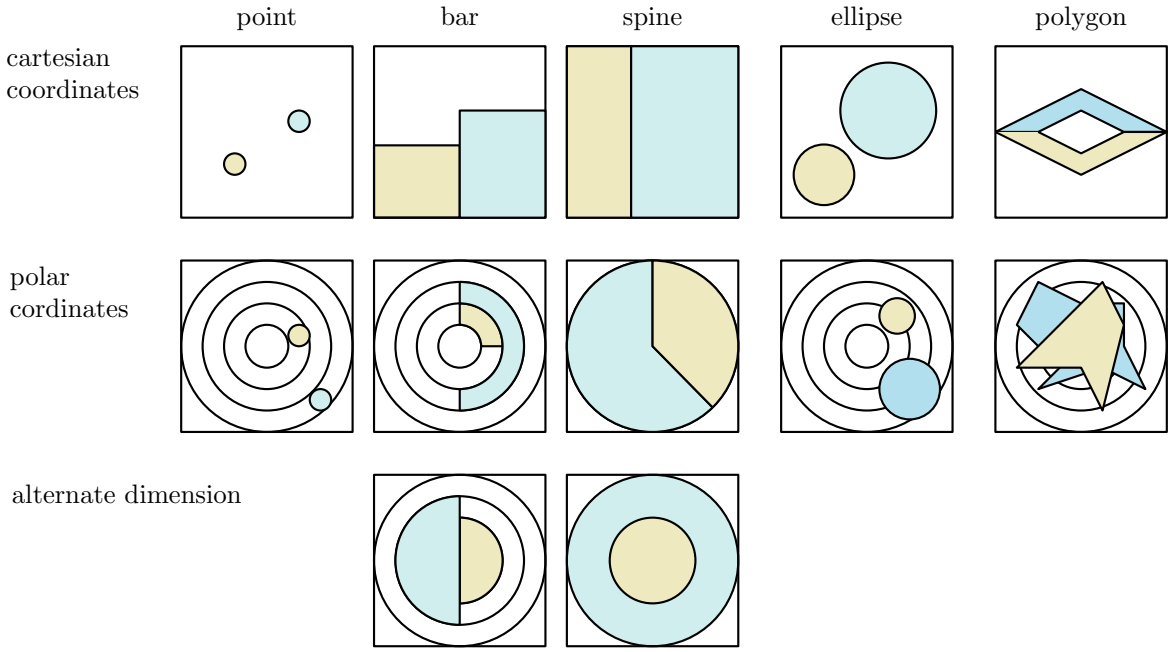


Fig. 5.7 *Shapes* supported in the framework. An attribute is mapped to height and width, respectively, for bars and spines in cartesian coordinates; although the spine shape occupies a larger area, this dimension is not conveying information. The concentric circles in the polar row indicates the radial subdivisions of the available area. The third row represents the use of a variable in a different orientation for polar coordinates. For example, the bar shape is displayed in the second row with a variable assigned to angle, configuring an *annular sector*, whereas in the third row it is assigned to radius. The same applies for spine: in the second row, the angular spines configure *circular sectors*, whereas in the third row the concentric circles are *space-filling* in the radius dimension.

Some decompositions are straightforward to extract based on the publication where the visualisation was described; sometimes, however, the visualisation source does not include details about the *construction* of the encoding or the original description needs to be *adapted*. In such cases, the following method is used to decompose encodings from the mappings descriptions into layered specifications:

1. For each variable that is assigned to at least one positional channel, assign one of the layout cases;
2. For each variable assigned to a shape, mark as a conditioning variable and assign a layer number;
3. For each *attribute* variable listed under another variable type, assign the same layer number. This covers every case of encoding except 2D combinations of two attribute variables, such as some scatterplots;
4. For each variable with layout case *n-to-2*, assign the same layer number. This covers spiral layouts, in which more than one time variable is used for two positional channels;
5. For every layer after the first, assign a composition method between juxtaposition, superimposition and nesting;

This method results in multi-layered visualisations, where each layer corresponds to a 1D partition of space by identifiers, space or time or a 2D combination of any two variables.

Example Table 5.2 contains an example of applying this method to the *Kaleidomaps* (Bale et al., 2007) technique. The technique uses polar coordinates to display instant time. The space is partitioned along the angular dimension for different *identifier* variables corresponding to a measurement; each sector is then partitioned based on a combination of two *time* variables and coloured based on the value of that measurement (*attribute*). Each non-attribute variable is mapped to at least one positional channel (alternating between angle and radius) – they are assigned a *1-to-1* layout case. As the use of space is space-filling for each variable, they are assigned a *spine* shape. Following the method described, the non-attributes variables are marked as a conditioning variable and a layer number is assigned to them. The attribute variable is assigned to the last temporal variable as it is used to *colour* the last layer with a *spine*. Finally, each layer is composed using *nesting*. The visualisation is displayed in fig. 5.8;

the first two layers are displayed in the black-and-white figure, reproduced from the source publication.

Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Identifier	SCALE	–	–	–	✓	-	SPINE	1-to-1	✓
Time 1	SCALE	–	–	–	✓	-	SPINE	1-to-1	✓
Time 2	SCALE	–	–	–	-	✓	SPINE	1-to-1	✓
Attribute	SCALE	✓	–	–	-	-	-	-	-

Layers:

Layer	Conditioning Variable	Comp Method	C	S	P	x/θ	y/r	Layout	Shape
1	**Identifier**	–	–	–	-	**I**	-	SCALE	SPINE
2	**Time 1**	*nesting*	–	–	-	$\mathbf{T_1}$	-	SCALE	SPINE
3	**Time 1**	*nesting*	**A**	–	-	$\mathbf{T_2}$	-	SCALE	SPINE

Table 5.2 Description of visual mappings of Kaleidomaps. Abbreviations: **C**olour, **S**ize, **P**ath. The double column borders in the layers table indicate the separation between the framework's components, as *conditioning variable* and *composition method* are part of the composition component. The composition signature for this visualisation is the following: $N((\mathbf{I},L_1),N((\mathbf{T},L_2),(\mathbf{T},L_3)))$.

5.4.1 Visual mappings

Considering the method described and including the properties defined by the composition component, each layer contains the following set of properties:

Conditioning variable any variable, or a list of variables for special layout cases (see below for spiral example);

Composition method juxtaposition, superimposition or nesting;

Shape *POINT*, *BAR*, *SPINE*, *ELLIPSE* or other polygon shapes;

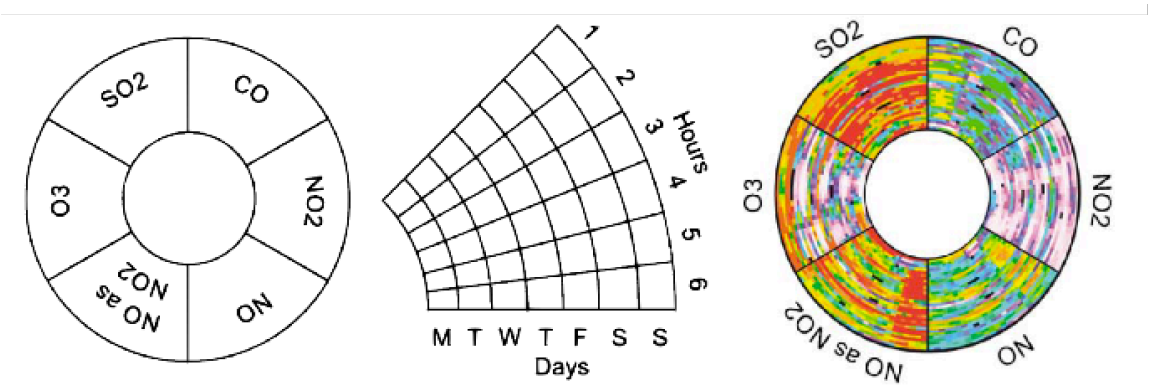


Fig. 5.8 Layers of Kaleidomaps reproduced from Bale et al. (2007). In the original article, the combination of the two temporal variables (middle) is described as a 2D partition; in the framework, separating the two time variables enables different arrangements between layers. The figure on the right is the final visualisation.

Layout *SCALE* for default transformations, any *ALGORITHM* which is supported by the implementation, or other keywords that define a layout, such as *SPIRAL*. Unless specified, the layout choice affects both positional axes;

Colour and/or Opacity any variable;

Size 1 and Size 2 any variable, assigned to the first (width or angle) and/or second (height or radius) axis;

Path any variable;

Position 1 and Position 2 any variable, assigned to the first (horizontal or angle) and/or second (vertical or radius) axis, or a list of variables for special layout cases (see below for spiral example);

5.5 Within level composition

Within level composition relates to arrangements of visual marks that are located in the same level of the hierarchy. In this case, only *juxtaposition* and *superimposition* compositions are plausible; it is illogical to nest visual marks representing data items in the same level within each other. For time visualisation, within level composition covers the common *multiple line chart*. However, the survey did not include this technique, including the *single* line version instead. Supporting *within level* composition in the framework requires defining a convention for values of the properties that enable these

configurations. As such, the following default conventions are used for descriptive purposes:

- **Juxtaposition**: this composition is defined by the use of a *spine* shape with *undefined* size property. The interpretation of this configuration is that spines with no *data-driven* size share the visual space *equally* and are ordered by the variable used for position. *Data-driven* juxtaposition is already defined by the use of a spine with variables assigned to size.
- **Superimposition**: this composition is defined by the use of a *spine* shape with *undefined* layout and size properties. The interpretation of this configuration is that spines are rendered *over* each other, each of them occupying the whole visual space. This is consistent with Wickham and Hofmann's (2011) definition of spines as *space-filling* shapes. In this case, one position channel defines the *z-order* of the the layers.

5.6 Examples

In this section, representative examples of layers and encodings are described using the specifications and the composition methods. The specification of all surveyed techniques is found in appendix A.

1D Identifier or Time

Identifier or time mapped to one dimension are basic building blocks for time-oriented visualisations. Through spine shapes, they are used with nesting composition to partition the visual space hierarchically. When an attribute is mapped to colour, they are generally used as a *heat map* layer. Both configurations were seen in the Kaleidomaps example with both identifier and temporal partitions. A cartesian configuration of heat maps was used by Boyandin et al. (2011) and Guo et al. (2006) and is described using the following specification:

Flowstrates heat map, Boyandin et al. (2011)					CARTESIAN,INSTANT				
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Identifier**	–	–	–	-	-	**I**	SCALE	SPINE
1	**Time**	*nesting*	**A**	–	-	**T**	-	SCALE	SPINE

Table 5.3 Flowstrates specification. Signature: $N((\mathbf{T},L_1),(\mathbf{T},L_1))$

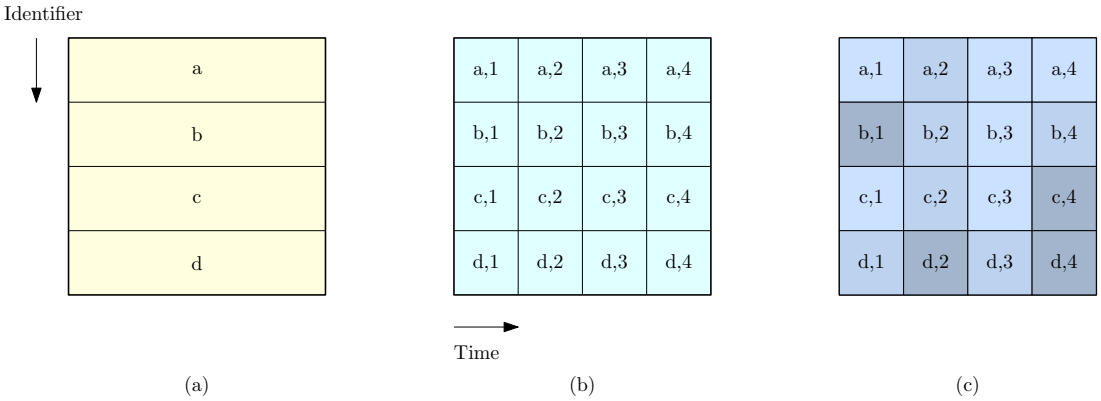


Fig. 5.9 Heatmap example with the two layers: (a) shows the rendering of the first layer; (b) shows the rendering of the second layer without colours; (c) shows the complete rendering with attribute-based colouring.

2D [Time,Attribute]

The configuration that is used for bar and line charts, 2D [Time,Attribute], is described by the following configuration (for bar charts):

Bar chart					CARTESIAN,INSTANT				
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Time**	–	–	$[-,\mathbf{A}]$	-	**T**	**A**	SCALE	BAR

Table 5.4 Bar chart specification. Signature: $(\mathbf{T},L_1)$

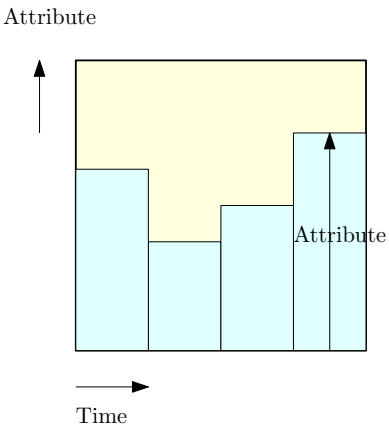


Fig. 5.10 Single layer bar chart example.

Line charts are specified in a similar way, with the use of the *Path* property along with the point shape and no size assigned:

Single line chart					CARTESIAN,INSTANT				
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Time**	–	–	–	**T**	**T**	**A**	SCALE	POINT

Table 5.5 Single line chart specification. Signature: (**T**,L_1)

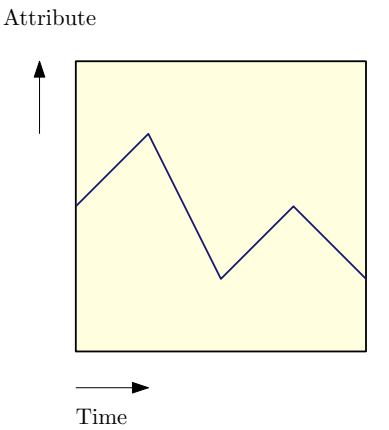


Fig. 5.11 Single layer line chart example.

This specification corresponds to a *single* line chart. The composition component, along with layers defined with *spine* shapes, enables the superimposition of the line chart layer multiple times through *within level composition*, as in the following:

Multiple line chart					CARTESIAN,INSTANT				
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Identifier**	–	–	–	-	**I**	-	SCALE	SPINE
2	**Time**	*nesting*	**I**	–	**T**	**T**	**A**	SCALE	POINT

Table 5.6 Multiple line chart specification. Signature: $N((\mathbf{I},L_1),(\mathbf{T},L_2))$

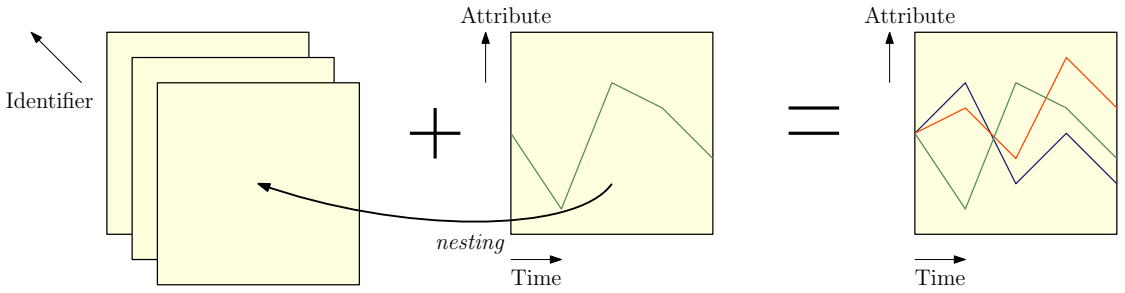


Fig. 5.12 Multiple line chart example. On the left side, *within level* superimposition is displayed – in this layer, spines with undefined size are used. In the middle, the result of specification of the second layer is displayed: a line chart with *time* connecting the points. On the right side, the final visualisation is shown, as a result of the *nesting* indicated by the arrow in the middle of the figure.

2D [Time,Time]

The configuration of time variables in two dimensions can be done with composition for various *1-to-1* layout cases, such as the Kaleidomaps example given, or in a single layer for *1-to-2* or *n-to-2* cases, such the time wave (Li and Kraak, 2013) or spirals (Carlis and Konstan, 1998; Weber et al., 2001). The spiral is a special case of layout, as it is parameterised by two temporal variables: the number of time points in a spiral *arm* and the number of *arms*. In this case, the *conditioning variable* is defined as the combination of two variables; this is the same case for the positional channel. The spiral is described through the following specification:

Spiral, Carlis and Konstan (1998)					POLAR,INSTANT				
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	$[\mathbf{T_1}, \mathbf{T_2}]$	–	–	–	-	$[\mathbf{T_1}, \mathbf{T_2}]$	$[\mathbf{T_1}, \mathbf{T_2}]$	SPIRAL	POINT

Table 5.7 Spiral specification, with two temporal variables being used for conditioning and position. Signature: ($\mathbf{T}$,L_1)

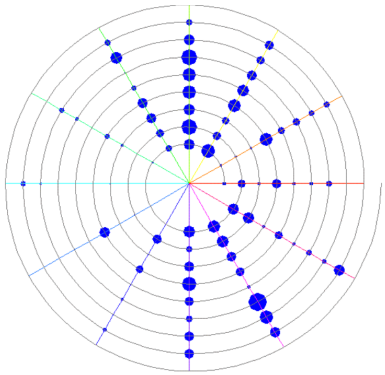


Fig. 5.13 Spiral example

2D [Attribute,Attribute]

This configuration describes *scatterplots*; among the surveyed techniques, it includes the connected scatterplot (Haroz et al., 2016) and scatterplots with time mapped to colour (Lei and Zhang, 2010). The connected scatterplot can be described through the following specification:

Connected scatterplot, Haroz et al. (2016)					CARTESIAN,INSTANT				
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Time**	–	–	–	**T**	**A**	**A**	SCALE	POINT

Table 5.8 Connected scatterplot specification. Signature: ($\mathbf{T}$,L_1)

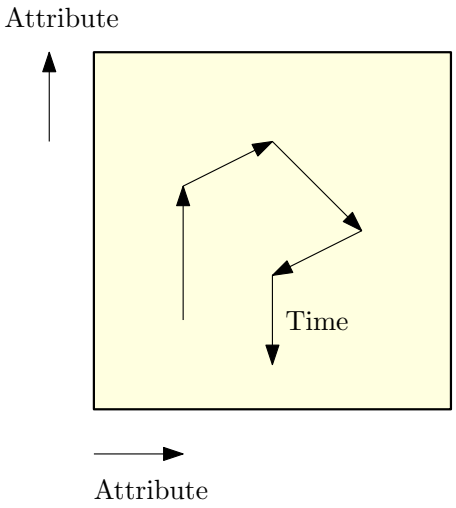


Fig. 5.14 Connected scatterplot example.

5.7 Framework context

The specifications described in this section fit into two parts of Card-Norman's model: the description of the *operands* in the interaction cycle and the *visual mappings* step of moving from *data abstractions* to *visual structures*. This involves the use of six properties related to the resulting visual marks: **colour**, **size (width or angle)** and **size (height or radius)**, **connecting path**, **position (x or theta)** and **position (y or r)**, **layout** and **shape**. In addition to the low level description of encodings, the conditioning variable and composition methods defined in the previous chapter are also used for a complete specification of a visualisation technique.

In the same vein as the operations that were suggested in the previous chapter to manipulate the composition grammar, the following operations should be used to assign and modify the visual mappings:

- **Assign** a variable or value to a property;
- **Modify or remove** a variable assignment;
- **Modify** the coordinate system, when possible;

5.8 Discussion

There are several points of discussion regarding this component. The first important one is the order in which variables are used to describe the hierarchical levels. The *Kaleidomaps* example showed how a different ordering may be used to achieve the same result. For visual exploration, this is a very important factor as it affects the paths that can be taken when navigating between visualisations. The method described in this chapter to *convert* a visualisation into the framework's descriptions assumes a *priority* for identifier variables, followed by time, space and lastly other attributes. This order is justified by the fact that the framework deals with *time-oriented* visualisations, where spatial, quantitative or qualitative attributes are associated with recorded times. In practical use, this may not always be the case; however, it is a characteristic or limitation of the framework that must be considered when using it.

A second point of discussion is the number of channels that were included in this research. As a result of the survey, in addition to positional variables, three other channels were used: size, path and colour. However, other channels could also be used as discussed by Aigner et al. (2011): thickness of lines, texture, glyphs, text annotations, containment or orientation/angle in cartesian coordinates. Some of these were not included as they did not emerge during the survey, such as orientation/angle. Others, such as *thickness of line*, could be applied to various aspects of visualisations, such as thickness of the lines in visual marks or in connecting paths. In this case, several other attributes could also be used: for example, dotted or dashed lines. Although the framework is fully compatible with their inclusion in the future, the decision was, at this stage of the thesis to limit the number of attributes in the view component and focus first on a complete framework.

The relationship between *layouts*, *position* and *size* is also open to discussion. The design choice for the component was to allow low level, domain-independent views to be defined. This means that, at the point of computing certain layouts, the data points for each variable have already been transformed as needed. At the same time, there are layouts that use the results of algorithms or calculations as an important step. One example is the *MDS* algorithm used in *Time Curves*, which generates 2D positions from a *correlation matrix*. In the framework, the MDS positions are the data variables that are mapped to the positional channels; the framework does not make a particular distinction between positions produced by the MDS algorithm or geographic coordinates. On the other hand, visualisations such as *ThemeDelta* are defined by

using an algorithm as a layout; this decision was done with basis on the inputs and outputs of algorithms; in the case of ThemeDelta, the algorithm does not take any special data as input, unlike the correlation matrix used for the MDS.

Another point of discussion is the use of the basic layers with scale and algorithms, in contrast with *predefined* visualisations. Some visual exploration systems such as Tableau use *predefined* visualisations such as *stacked bar* charts and *maps*, still allowing a degree of reconfiguration. On the other hand, grammars such as *Vega-Lite* include properties that allow defining a *stacked* bar chart as an extension of a normal bar chart; the same method works for *area* and *stacked area* charts. In yet another extreme are specific design space for families of visualisations, such as the design space by Baudel and Broeskema (2012) for *treemaps*.

The approach taken in the framework makes the description of views compatible with *transitioning* between configurations in the design space of hierarchical time-oriented visualisations. The argument presented in this thesis is that a flexible approach is more important for researching time-oriented visualisations than a *rigid* one; this is due to the fact that the primary aim of a design space is to enumerate the *universe of design choices* (Schulz et al., 2011), even if some design choices are ineffective for a range of tasks. The relative independence of each component also allows visualisation researchers and designers to choose which *parts* of the framework are needed for their applications or experiments.

5.9 Chapter summary

This chapter introduced a decomposed view of visual encodings that addresses two secondary questions, *what are the interactive visualisation methods used for temporal data?* and *how can hierarchical composition techniques be combined with the surveyed visualisations?* The specifications of encodings as layers is presented as a component of the framework that addresses the primary research question. The combination of this component with the *composition component* presented in the previous section allows the specification and reconfiguration of hierarchical visualisation techniques. Although there are temporal aspects in the definition of the compositions and encodings through the use of temporal variables, the aspects of time that characterise the *exploration* of time-oriented data are yet to be addressed. The next chapter addresses the third

secondary question and completes the framework by extending the reconfiguration of visualisation to time-based transformations.

Chapter 6

Transformation: interactions

This chapter introduces the remaining component of the framework, the *Transformation* component, which contains the *conceptual transformations* of the temporal domain that enable the interactive visual exploration. While previous chapters were concerned with the parts of the framework that connect time-oriented visualisation with hierarchical and composite visualisations, this chapter aims at answering the third secondary question — *what are the temporal interactions that facilitate exploring temporal data in hierarchical visualisations?* As described in chapter 3, the transformations are designed based on a combination of analysis of the interactions in time-oriented data visualisations and the use of conceptual models of temporal data. This chapter details the process that leads to the different categories of transformations and their application as functions with distinct parameters.

6.1 Addressing the research questions

This chapter addresses primarily the third secondary question. The aim of this chapter is to introduce conceptual transformations of the time domain that can be applied to the other components, enabling the use of *multiple perspectives* for temporal data exploration. In order to do that, the chapter relates the existing literature of time-oriented data visualisation with concepts of temporal data. The description of the transformations and their interactions with the other components, besides being presented as a contribution and answer to research question, also addresses the gap identified in chapter 2, where it was identified that current models and frameworks

for time-oriented data visualisation fail to include the time domain in the interactions supported by these models.

6.2 Survey of time-based interactions

The first step of this stage was to analyse the literature to identify the interactions in interactive visualisations for time-oriented data. This included the references surveyed as part of the *view* component and other references that were considered unsuitable for the view component, such as interactive three-dimensional visualisations. This step was used to filter out entries that do not contain any description about interactions, such as new visualisation techniques that were not tested in an interactive environment. For each of the initial 38 entries where interactions were described, the taxonomy of interaction techniques by Yi et al. (2007) guided the extraction of information. In addition to the seven basic categories that the authors describe – *select, explore, reconfigure, encode, abstract/elaborate, filter, connect* – two additional categories were used: visualisation configuration and design transformations. The first is discussed by Yi et al. as a category of interactions that are not exclusive to interactive visualisation, and thus their taxonomy does not include it. In the context of this thesis, however, it is important to consider aspects of visualisations that are part of the various stages of Card-Norman's model, such as the parameters that guide the generation of visualisations. Visualisation configuration includes temporal transformations that are triggered by interface settings, such as resizing the visualisation window. Design transformations include temporal transformations that are applied in the generation of the visualisation, rather than being only applied by user input during *runtime*; this includes aggregations that are used to decrease the number of items on screen, for example. Each entry was analysed and textual descriptions were added to each category where appropriate, as seen in appendix D.

The second step was to filter the interactions that are relevant for the framework, which are those that belong to the following categories: reconfigure, abstract/elaborate, filter, visualisation configuration and design transformations. This excludes the select, explore and connect categories, which contain interactions that, for time visualisation, often involve purely visual transformations, such as highlighting elements, or transformations that involve other types of data, such as querying quantitative information related to certain times. These interactions are out of the scope of the framework (see chapter 9

for more discussion about the limitations of the framework). This step reduced the number of entries to 30.

The third step was to relate the textual descriptions to the *properties of time* that are modified or *transformed* by the interactions. Doing so required determining, from existing theories and models of time, which of these properties are important to support visual exploration. For this, the notion of a *discrete bounded time domain* and the concept of granularities defined by Bettini et al. (1998), described in chapter 2, was used. A time domain contains the temporal references that are part of any dataset being visualised; these are usually *extracted* from the data items being visualised and isolated from other non-temporal data. The *discrete* aspects refers to the fact that the temporal references are indivisible; in such a model, dividing *1 day* requires changing that granularity that used for the time domain. In time visualisation, as the pixel is also an indivisible unit, generally speaking, it is sensible to reflect this characteristic in the abstraction of time as well. The *bounded* characteristic refers to the *lower* and *upper bounds*, or the lower and upper time points, that define the *extent* of the time domain. Although *infinite* time is conceptually possible, the fact that the *visible visual space* is also limited makes it sensible to consider the limits of time where data items cease to exist.

This representation of the time domain includes three other properties of granularity mappings: the unique granularity mappings (i.e. the mathematical properties of granularities), the types of labels and the variation of ordering of granularity labels. The types of labels are related to the different *scales* – changing between absolute time and relative time by re-labelling time points. The order of granularity labels is primarily related to cyclic time units, such as days of the week, in which the first day of the week can be changed; this is related to the variation of calendars and use of time in different application contexts.

Lastly, many visualisations subdivide the time domain into *segments* and display it in a non-contiguous manner. In order for the framework to support this, the concept of *segments* is used, representing contiguous subdivisions of the time domain that can be rearranged with varying length. Each reference was categorised by analysing these six *modifiable* properties in relation to the textual description of interactions; table 6.1 displays the results of this.

The final step in designing the component was identifying categories of transformations based on these properties. This involved identifying groups of conceptually similar

Reference	TE	B	G	GL	GO	NS
André et al. (2007)			✓			
Andrienko et al. (2011)			✓			
Beard et al. (2008)			✓			
Carlis and Konstan (1998)	✓	✓				
Cho et al. (2014)				✓		
van der Corput and van Wijk (2017)			✓			
Gad et al. (2015)			✓			
Gschwandtner et al. (2011)				✓		
Guerra-Gomez et al. (2013)			✓			
Javed and Elmqvist (2013)		✓				✓
Keim et al. (2004)			✓			
Kothur et al. (2013)	✓	✓				
Krstajic et al. (2011)			✓			
Lammarsch et al. (2009)			✓			
McLachlan et al. (2008)		✓				
Shen and Kwan-Liu (2008)			✓			
Sips et al. (2012)		✓				
Tominski and Schumann (2008)	✓					✓
Tominski et al. (2012)			✓	✓		
Wang et al. (2009)			✓	✓		
Van Wijk and Van Selow (1999)		✓				
Wongsuphasawat and Shneiderman (2009)			✓	✓		
Tableau[1]	✓	✓	✓		✓	
Spotfire[2]			✓		✓	

Table 6.1 Survey of the properties of time modified by interactions in the literature. **TE** = Temporal Extent, **B** = Bounds, **G** = Granularity, **GL** = Granularity Label, **GO** = Granularity Order, **NS** = Number of Segments.

transformations and the various parameters that can be used with them. The properties of time were used to define three categories:

- Segmentation: operators in this category allow the number of segments to be modified by dividing the time domain, re-arranging segments or reversing the transformation;
- Granularity: operators in this category change the properties related to time granularities, by changing the level of aggregation, exploring a granularity structure or modifying inner properties of granularities such as the order or labels of time points;
- Extent and bounds: operators in this category change either the temporal extent or the upper or lower bounds of the time domain.

The next section describes the operators within each category, including details of their functionality and the parameters that can be used.

6.3 The transformation component

The component includes 18 individual transformations across the three categories with four types of parameters. Each property of time is modified by operators that can *apply* transformations or *redefine* or *reverse* the effects of previous transformations. This triad of methods allows the transformations to be defined at a more concrete level than a taxonomy of transformations, yet still operate in the *conceptual* realm rather than proposing low-level implementation algorithms. The addition of different types of parameters is also in line with this objective; besides facilitating future implementations of the framework by considering the types of operators and parameters that are appropriate and needed for different encodings and tasks. In addition to the three basic time-related parameters, some transformations require other types of parameters.

6.3.1 Conceptualising the time domain

As mentioned in the previous section, the notion of a discrete bounded time domain is used to define *how* the transformations affect time. In order to describe these transformations in detail, the following formalisation of the time domain is used:

$$\mathcal{T} = \{s_G, e_G, G, E, P\},$$

where s and e are the starting and ending *instants* of the temporal domain, labeled by granularity G; E is the number of *instants* in the domain (or the *temporal extent* of the domain), and $P = \{p_1, \ldots, p_n\}$ is the set of time points such that any $p = \{t_j, \ldots, t_k\}$, with $s_G \leqslant t_j < t_k \leqslant e_G$, where $i \in \mathbb{Z}$, $t_i = G(i)$ is an instant labeled by granularity G. Additionally, the following constraints guarantee that the time domain is *discrete* and every instant within the bounds exist in the abstraction, ensuring that the time domain is complete as discussed in chapter 4: p: $n \leqslant E$, $1 \leqslant |p| \leqslant E$ and $E = \sum_{p \in P} |p|$.

These properties of the abstraction correspond directly to the properties of time used to categorise the interactions, as summarised in table 6.2. The *starting* and *ending* instants correspond to the bounds of the time domain, while the extent correspond to the time extent, or the number of instants in that dataset. *Granularity* (G) reflects the current granularity used to label the time domain, including the order of the labels. The time domain object itself can correspond to the whole time domain or segments of it, with the possibility of multiple time domains $\mathcal{T}, \mathcal{T}', \mathcal{T}'', \ldots$ being defined. Although some of these properties are redundant because they can be derived from each other, they are used in this chapter to describe how some operators can directly modify a property through various types of interactions, with the changes then propagated to other properties.

Example Consider the following example:

$$\mathcal{T} = \{1999_{\text{YEAR}}, 2015_{\text{YEAR}}, \text{YEAR}, 5, \{2011, 2012, 2013, 2014, 2015\}\}$$

This instance of the time domain has a lower bound *1999*, upper bound *2015*, granularity YEAR, extent *5* and the set of time points P corresponding to the instants 2011 and 2015. As it is discussed in the description of the operators, time points can contain aggregations of instants. The following is a valid alternative to P: $\{[2011, 2012, 2013], [2014, 2015]\}$. In this case, the time domain was modified to contain two representable time points.

Parameters of transformations

In order to modify the properties of the abstraction, the proposed operators can receive different parameters, illustrated in fig. 6.1. The basic variation of parameters is based on two types of time, instants $t_i, \ldots, t_n$ and duration d, as well as granularities $G, H, I, \ldots$.

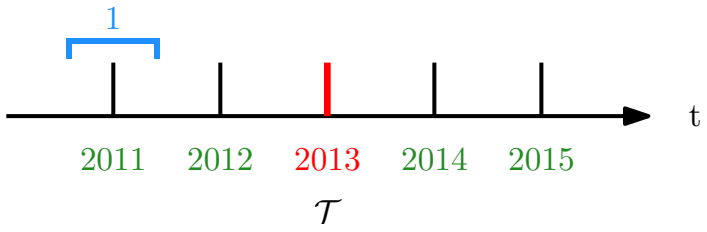


Fig. 6.1 Illustration guide for the description of the operators. The instant highlighted in red was passed as a parameter; the blue square bracket with a number on top indicates that it is a duration of length 1 that was passed as parameter; the green colour used in the labels indicates the granularity that was used – additional granularities have different colours, as seen in fig. 6.5.

The parameters allow the transformations to have the same result through different inputs and behaviours. Passing instants as parameters to a transformation may be more suitable for one kind of interaction mechanism rather than passing a duration, for example. Besides these three parameters, certain transformations require different parameters based on their functionality – these other parameters are introduced along with the operators.

In the following sections, each transformation is described along with a *function signature* that summarises the transformation along with the input parameters. Due to the complexity of the results of the transformations and the fact that they do not illustrate the varying behaviours by the use of different parameters, the formalisation of the output is not included. Instead, illustrations with *before, during and after* figures are used to describe the operators in detail. Additionally, pseudo-code algorithms are included in appendix C in order to facilitate the understanding of the operators.

Preconditions and ambiguity

Applying the operators on the time domain with multiple types of parameters requires a series of decisions on the validity and interpretation of the relationship between them. This includes deciding whether it is logical to use a certain operator in combination with a configuration of time domain or checking if a parameter is temporally valid. In this thesis, the conceptual and pseudo-code descriptions do not include such tests of validity; this is assumed to be decided when the operators or the framework are implemented, when such tests would be embedded in the application. This also includes the use of optional and default parameters; for certain operators, default functionality is *suggested* to complete the space of possibilities.

The second issue regards the ambiguity that results from matching parameters to the time domain. One simple example is scanning for an instant in the time domain in order to divide it in two at that point: after the instant is found, it can be included in the first subdivision of the time domain or the second subdivision. While this has no direct bearing on the *conceptual* definition of the operator, which is to divide the time domain, the functionality and results can be radically different. Two possible interpretations are possible: matching instants through *lower than* or *lower than or equal to* relations, as in figure 6.2. Given the division example, in the first case, the first subdivision would include all instants *lower than* the parameter instant. In the second case, the first subdivision would include all instants *lower or equal* than the instant passed as parameter. In this chapter, a *lower than* interpretation is used for consistency in the descriptions.

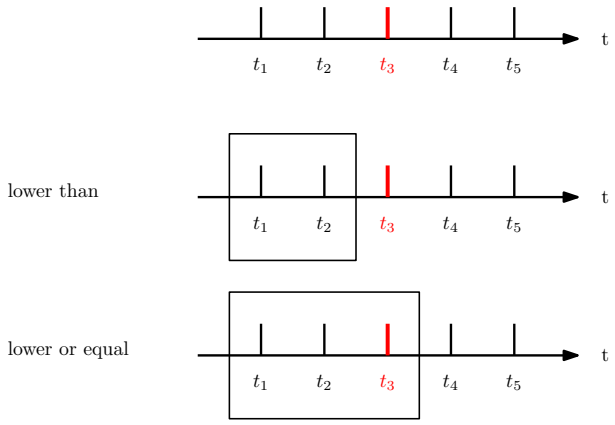


Fig. 6.2 Different relationship types when matching instants in operators. The upper part of the figure shows that t_3 was passed as parameter. In each example, the box outlines the instants that were matched being *lower than* or *lower than or equal to* t_3, respectively.

6.3.2 Segmentation operators

Segmentation of the time domain has a range of applications in various areas. One example is analysis of trajectories (Demšar et al., 2015), which involves space and time along with additional attributes. In this case, segmentation is done by classifying parts of a trajectory that conform to relevant criteria in order to identify, for example,

Category	**Property**
Bounds	s_G, e_G
Extent	E
Granularity	G, P
Segments	$\mathcal{T}$

Table 6.2 Categories of interactions related to the properties of the abstraction of the time domain. As the set of time points is closely related to the granularity, the two properties are included in the same category.

different behaviours along that trajectory. It includes splitting trajectories where there is a significant change in a trajectory's attribute, such as direction or angle of movement, or in spatial characteristics, such as a trajectory staying a certain amount of time in the same location or area. In all these cases, different degrees of automation can be used, from fully automatic segmentation based on algorithms to semi-automated methods, where expert users define parameters or directly choose points to segment (Buchin et al., 2013). Segmentation is not restricted to spatial trajectories, as variations of attributes are also used to segment the time domain in time series analysis (Keogh et al., 1993). The key aspect of the segmentation operators defined in this framework is the independence of the operators from the context – this enables the operators to be used in spatial or non-spatial trajectories, for example.

Regarding segmentation of time in a visualisation context, Brehmer et al. (2017) defined segmented timelines as those that are divided according to meaningful *temporal divisions* of time; such definition restricts segmentation where only calendric units are used to divide the time domain. In this thesis, the concept of segmented time encompasses any subdivision derived from an initial time domain, even if the resulting segment has no *temporal* meaning in terms of calendars.

Segmentation acts by partitioning one time domain $\mathcal{T}$ into two or more domains $\mathcal{T'}_1, \mathcal{T'}_2,$
$\ldots, \mathcal{T'}_n$, each being disjoint in relation to each other such that $E_{\mathcal{T}'} < E_{\mathcal{T}}$ and $E = \sum E_{\mathcal{T}'_i}$. The variation of parameters allows different inputs to produce the same segmentation result, therefore allowing the transformation to be used in different contexts, including variations of the visual encodings, the application domain and the interaction mechanisms that trigger the transformation.

This category contains four operators that are used to *apply* the transformation, two to *redefine* it and one to *reverse* it.

Segment at Instants

Signature: $\mathscr{S}_t(\mathcal{T}, \{t_1, t_2, \ldots, t_n\})$

Segments time domain $\mathcal{T}$ at instants $\{t_1, t_2, \ldots, t_n\}$, generating segments corresponding to the subsets of $\{s_G, t_1\}$, $\{t_1 + 1, t_2\}$, $\ldots$, $\{t_n, e_G\}$ (see fig. 6.3). The type of interval, as discussed previously, will define the composition of the segments. With left-open/right-closed intervals, the first segment would be $\{s_G, t_1 - 1\}$. In addition to the type of interval, the relationship between s_G and t_1 may also result in different segments.

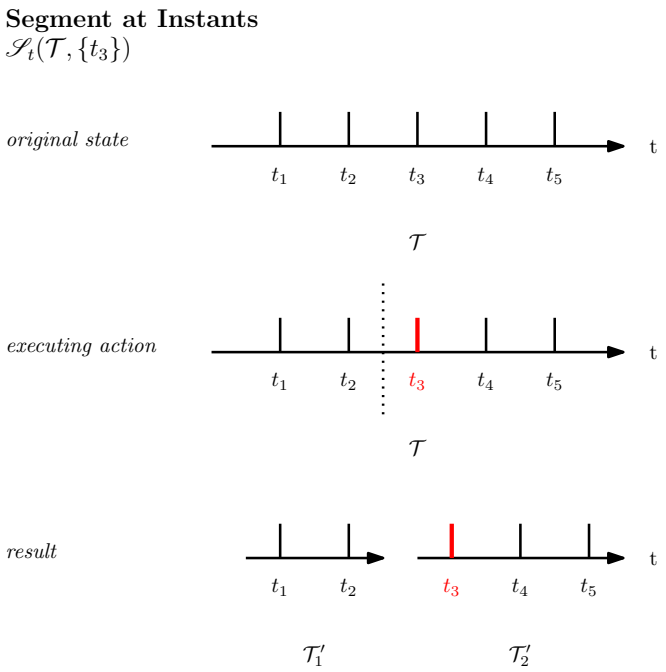


Fig. 6.3 Application of the *Segment at Instants* operator, with instant t_3 passed as parameter. In the second part of the figure, the instant is highlighted in red, with the dotted line indicating where the segmentation will happen. In the third part, the two segments produced are shown, with the parameter time point included in the second segment due to the use of *lower than* relation.

Segment by Duration

Signature: $\mathscr{S}_d(\mathcal{T}, d)$

Segments time domain $\mathcal{T}$ into $\lceil E/d \rceil$ segments, each with extent equal to d (see fig. 6.4). In contrast with the previous operator, where specific time points are used as parameters, duration allows the time domain to be segmented systematically with regular extents. The result of the operator is affected by the extent of the original time domain. If

$E \ mod \ d > 0$, the last segment will have an extent shorter than d. For example, if a time domain has extent $E = 10$ and $\mathscr{S}_d(\mathcal{T}, 3)$ is applied, the result will be three segments with extent 3 and a fourth segment with extent 1. There are two ways to treat this special case: if the objective is to generate uniform segments, the last segment can be *padded* with the missing time points; the alternative is to leave the segment as it is. There are again no consequences for the conceptual idea of segmentation, however, the difference in extents is an important implication for the general use of the operator.

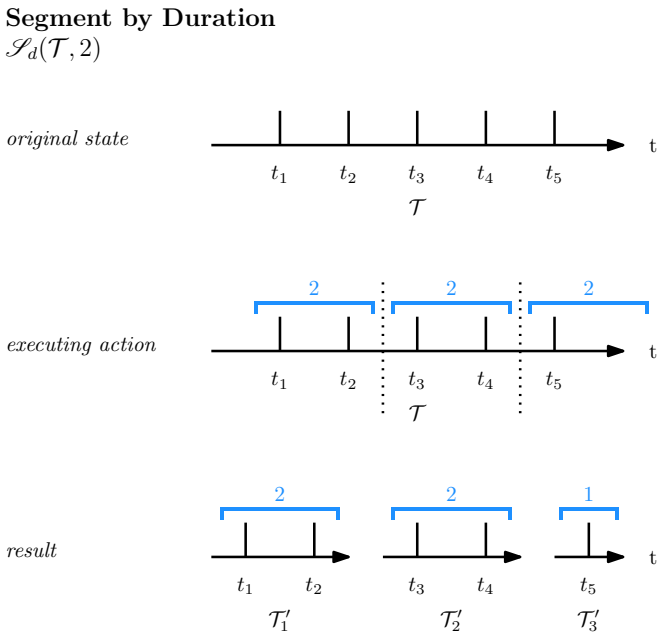


Fig. 6.4 Application of the *Segment by Duration* operator, with duration *2* passed as parameter. In *executing action*, the segments that will be produced with duration 2 are marked. In *result*, the third segment contains only one time point, because, in this example, the parameter resolution for an extent that conflicts with the parameters does not *pad* the resulting segment.

Segment Matching Granularity

Signature: $\mathscr{S}_G(\mathcal{T}, G)$

Segments time domain $\mathcal{T}$ with a *compound* granularity $G + H + \ldots$, where G is *coarser than* $H, \ldots$, into segments matching granularity G (see fig. 6.5). The objective of this operator is to generate segments based on calendar units, which sometimes are not as regular as the segments produced by the previous operator. One example is the variation of number of days in a month; segmenting with a duration of 31 days will not be correct for every month. This operator allows segmenting the time domain with predictable outputs in terms of temporal validity. However, the overall extent of the

original time domain also affects the result of the operator: a segment corresponding to a matched month will have only the number of days that exist within the extent.

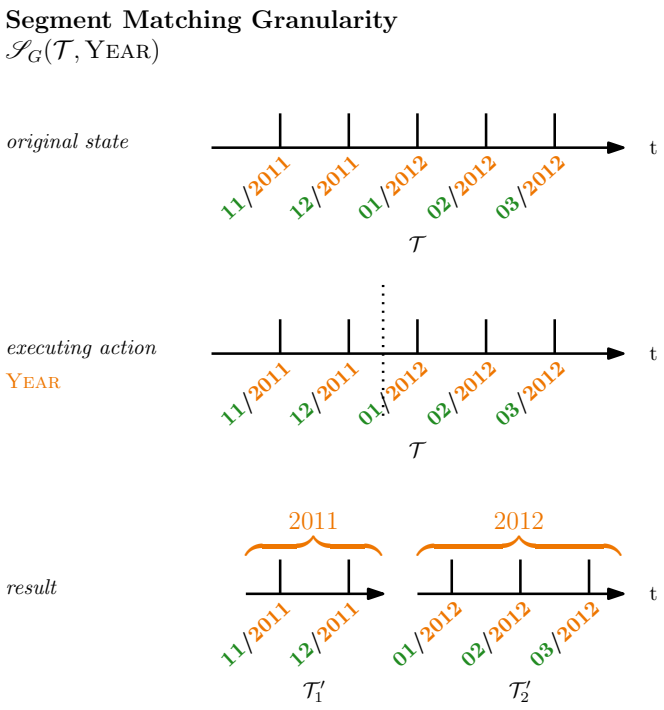


Fig. 6.5 Application of the *Segment Matching Granularity* operator, with granularity *Year* as parameter. In *executing action*, the dotted line indicates where the values of year change. In *result*, the two segments for years *2011* and *2012* are shown.

Segment Relative to

Signature: $\mathscr{S}_{t,f}(\mathcal{T}, t, f)$

Segments time domain $\mathcal{T}$ into segments with length calculated by function f, relative to instant t. Although this operator has no corresponding interaction in the literature, it is the segmentation-equivalent of the *bin relative to* operator (see below) and it is suggested to complete the space of possibilities for the operators. It allows a time point of interest to be chosen and an algorithm or function to be used to generate segments based on the distance to that time point.

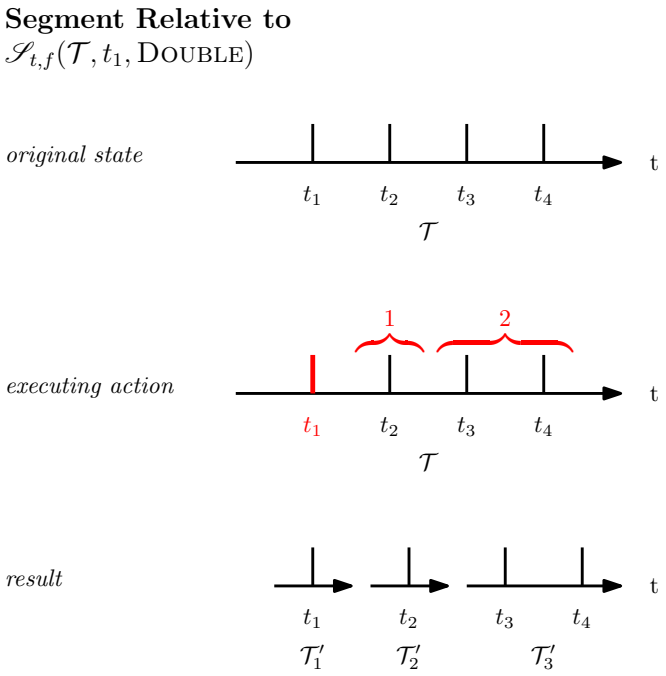


Fig. 6.6 Application of the *Segment Relative To* operator, with instant t_1 passed as parameter and *DOUBLE* as a function that calculates the segmenting distance from the reference point. For each calculated segment, the following segment size will have a twofold increase. This can be seen in the middle of the figure, where the first segment has size *1* and the second segment has size 2, resulting in the three segments the lower part of the figure.

Join

Signature: $\mathscr{J}([\mathcal{T}_1, [\mathcal{T}_2], \ldots])$ Joins two or more segments of time into a unified time domain (see fig. 6.7). This operator allows reversing the segmentation operator, applying it to specific segments or all existing segments.

6.3.3 Granularity operators

This category covers the transformations related to exploring the different time scales in the form of granularities. The proposed operators modify the set of time points associated with a time domain by changing the level of aggregation, the granularity used to label them and the order of the granularity labels. As seen in table 6.1, the transformations in this category are used as basic interactions in the majority of the

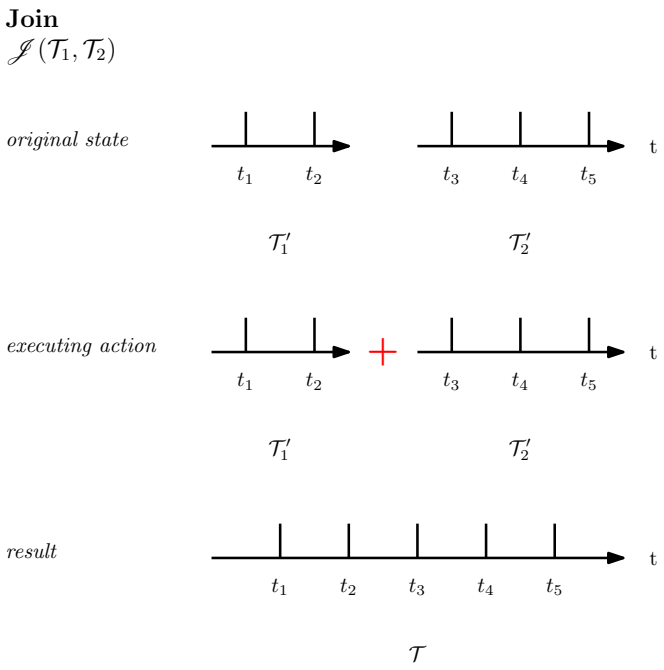


Fig. 6.7 Application of the *Join* operator, with two segments as parameters. The resulting single segment is shown in the lower part of the figure.

surveyed techniques. This section describes the various ways in which this interaction can be conceptually defined.

The operators can either alter the granularity G, indirectly modifying the set P of time points, or act directly on the set of time points by modifying one time point p to include (or exclude) one or more t_i such that a p_i might be equal to t_i, t_{i+1}, t_{i+2} or variations of these. The category also includes operators that modify the order of time points and transform the resulting labels from granularities.

Bin at Instants

Signature: $\mathscr{B}_t(\mathcal{T}, \{t_1, t_2, \ldots, t_n\})$

Aggregates time points between the intervals defined from instants s_G to t_1 and t_n to e_G and intervals in between, resulting in $n + 1$ aggregated time points. Similar to the *segment at instants* operator, the result of the operator will be different depending on the relationship between s_G and t_1 and e_G and t_n, as well as the type of interval used with the operator. This operator has also no corresponding interaction identified in the literature – it is suggested to complete the space of possibilities for the operators.

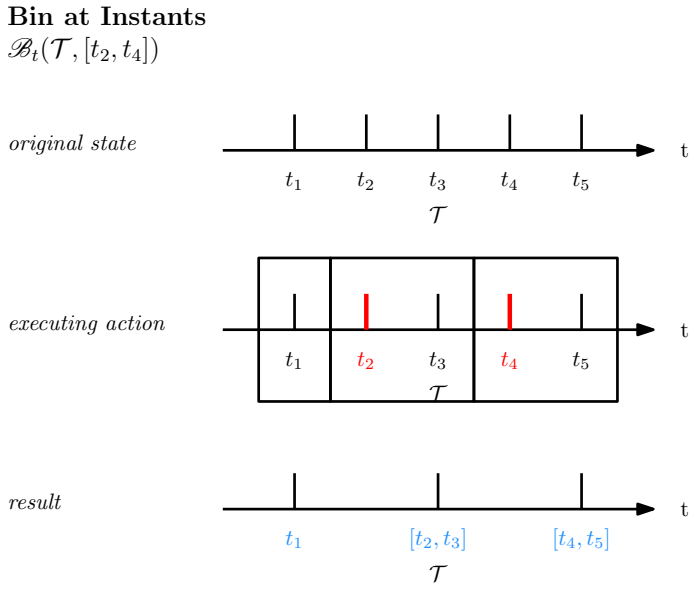


Fig. 6.8 Application of the *Bin at Instants* operator, with time points t_2 and t_4 passed as parameter. In *executing action*, the matched parameters are shown in red and the resulting segments enclosed. The lower part of the figure contains the resulting three time points, where two of them contain two instants.

Bin by Duration

Signature: $\mathscr{B}_d(\mathcal{T}, d)$

Aggregates time points in into $\lceil |P|/d \rceil$ regular bins of size equal to d (see fig. 6.9). Similarly to segment by duration, this operator can be used to systematically aggregate time points. In visualisation, it is commonly used to generate temporal histograms as overviews of large extents of temporal data (André et al., 2007; Tominski et al., 2012; Wongsuphasawat and Shneiderman, 2009), with bins containing an equal number of time points. In these cases, user interaction allows the scaling factor to be modified, although not as part of other exploration interactions.

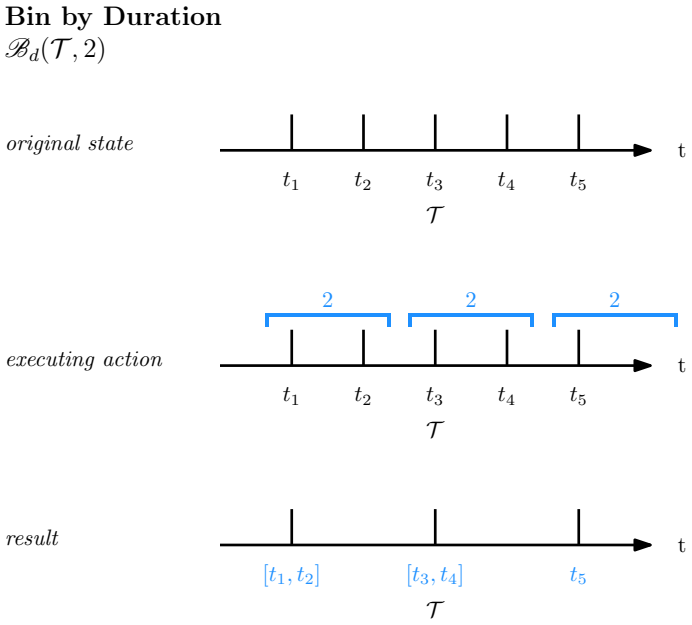


Fig. 6.9 Application of the *Bin by Duration* operator, with duration *2* passed as parameter. In *executing action*, the groups of instants are highlighted, resulting in the three time points in the lower part of the figure.

Bin relative to

Signature: $\mathscr{B}_{t,f}(\mathcal{T}, t, f)$

Aggregates time points into sets defined by function f relative to instant t (see fig. 6.10). This is equivalent to the segmentation operator with the same parameters; however, instead of splitting the time domain, this operator groups time points into bins, reducing the number of time points. This operator allows aggregations based on *fisheye views* (Furnas, 1986) transformations in the data space rather than in the visual space. One example is aggregating time points following a logarithmic scale, which is one of the types of scale proposed by Brehmer et al.: in this case, time points are aggregated in larger bins according to their increasing distance to the reference point. Other authors experimented with lenses modifying the visual space, such as Krstajic et al. (2011) with linear and logarithmic lenses and Kincaid (2010) with a distance-based lens.

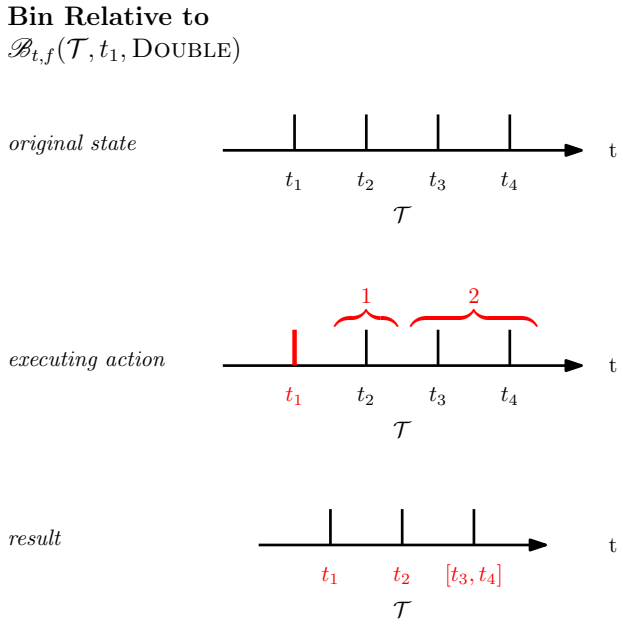


Fig. 6.10 Application of the *Bin Relative To* operator, with instant t_1 passed as parameter and *DOUBLE* as a function that calculates the binning distance from the reference point. For each calculated bin, the following bin size will have a twofold increase. This can be seen in the middle of the figure, where the first bin has size *1* and the second bin has size 2, resulting in the three time points in the lower part of the figure.

Expand

Signature: $\mathscr{X}(\mathcal{T})$

Reverses transformations done with previous operators by re-mapping every instant t_i to a single time point $|p_i| = 1$ (see fig. 6.11). This operator is suggested according to the aim of describing operators that allow reversing previous transformations.

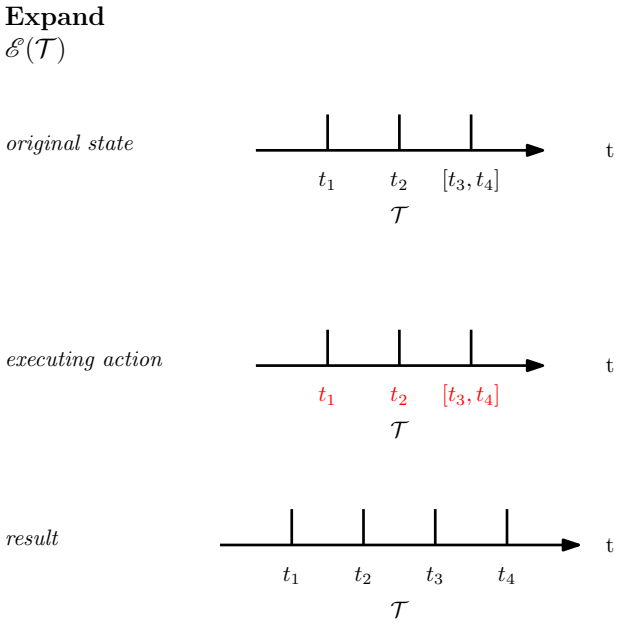


Fig. 6.11 Application of the *Expand* operator.

Change Granularity

Signature: $\mathscr{C}_G(\mathcal{T}, [G])$

This operator modifies the granularity that is used to label the time points in the domain, modifying the whole time domain including s_G and e_G (see fig. 6.12). It allows converting between granularities arbitrarily, as long as the underlying data supports them. Interactively changing granularities has been proposed as a central idea in some visualisations Keim et al. (2004); Lammarsch et al. (2009); Tominski et al. (2012) in order to explore different temporal scales; it also guides the reconfiguration of an event exploration system suggested by Beard et al. (2008). The application of this operator requires conversion between granularities that may not also be easily determined. For example, if a dataset contains references in *Months* and a user wants to change to *Month-Day*, the application implementing the operator must decide if month 08 is converted to $08-01$, $08-31$ or a midpoint such as $08-15$.

In addition to converting between granularities, it is suggested that, as *default functionality*, this operator allows removing granularities from time domains without any parameter, re-labelling existing references into a *sequential scale* based on the set of natural numbers $\mathbb{N}$.

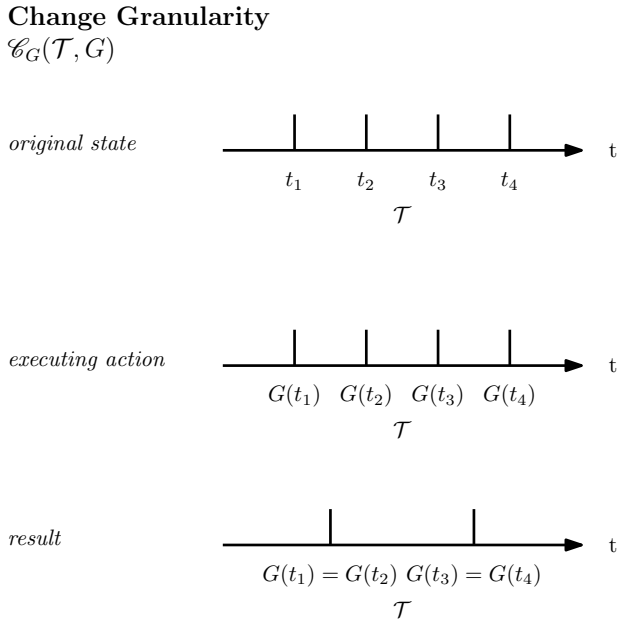


Fig. 6.12 Application of the *Change Granularity* operator.

Roll Up/Drill Down Granularity

Signatures: $\mathscr{C}_{\text{RollUp}}(\mathcal{T})$ and $\mathscr{C}_{\text{DrillDown}}(\mathcal{T})$

Re-labels time points using a predefined granularity G *coarser* or a granularity I *finer* than the current granularity H (see fig. 6.13). This operator allows systematic exploration of granularities when used with a hierarchy of granularities. While calendar granularities are actually defined in a *lattice* with multiple conversion paths, a linear path of conversion is generally used for interaction. This is the case of the application LifeLines2 (Wang et al., 2009), where a mouse wheel scroll changes the granularity from *Days* to *Months* and so on. *Roll Up* and *Drill Down* are basic operations in OLAP cubes that allows exploration of coarser or finer levels in a multidimensional array; in this thesis, they are suggested as time-exclusive operator in order to differentiate it from similar operators, such as exploring a hierarchy of geographical areas in a similar linear fashion.

The operators can have two types of behaviour: exploring single or composite granularities. In the example given in fig. 6.13, *Year* is transformed into *Year-Semester* (composite granularities); the alternative behaviour would transform it into *Semester*, resulting in only two time points: *1* and *2*. As this pair of operators is entirely dependent on the granularities of the dataset, it is suggested that the behaviour is

implemented based on the context. For this reason, the pair of operators is defined as it is rather than multiple operators being defined for each type of granularity.

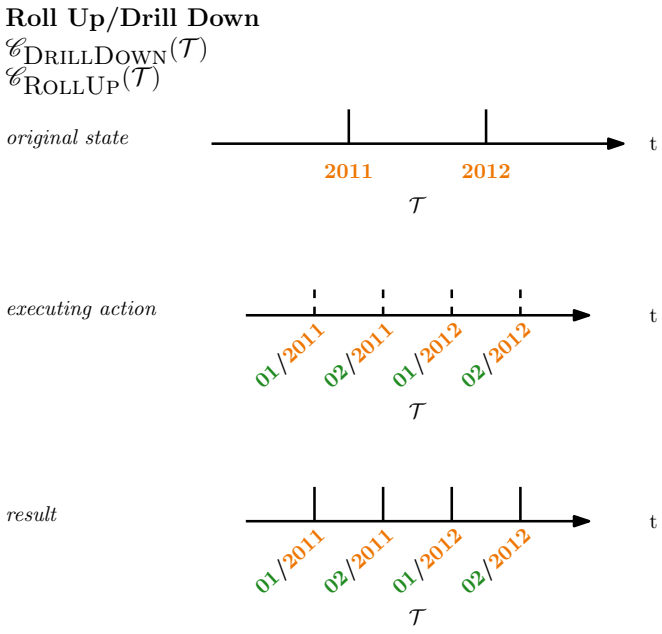


Fig. 6.13 Application of the *Drill Down* operator. In the *original state*, the two time points are labelled by granularity *Year*. When the operator is executed, four time points are added corresponding to a hypothetical immediate finer granularity *semester*. The granularity structure can be different depending on the application: *month* is generally the immediate finer granularity that is used. In this example, a *Roll Up* operator would transform in the opposite direction: the original state would be the last part of the figure, with the resulting state being the first part of the figure.

Rotate

Signature: $\mathscr{R}_D(\mathcal{T}, D)$

For a time domain $\mathcal{T}$ with *cyclic* granularity G, if direction D equals RIGHT, replaces starting point s_G with ending point e_G and re-labels time points with the label of the previous time point. If D = LEFT, replaces ending point e_G with starting point s_G and re-labels time points with the label of the next time point. Although cyclic granularities conceptually represent a repeating cycle with no clear beginning and end, visualisations generally *project* cyclic time onto a linear visualisation, such as a line chart. In this case, rotating the time points allow different starting and ending points to be explored, possibly showing patterns or trends that were otherwise hidden. Systems like Tableau and Spotfire allow users to choose the *starting month* for a fiscal year, for example, due to the variation between countries in terms of calendar systems

and business calendars. This operator generalises this idea to any cyclic granularity.

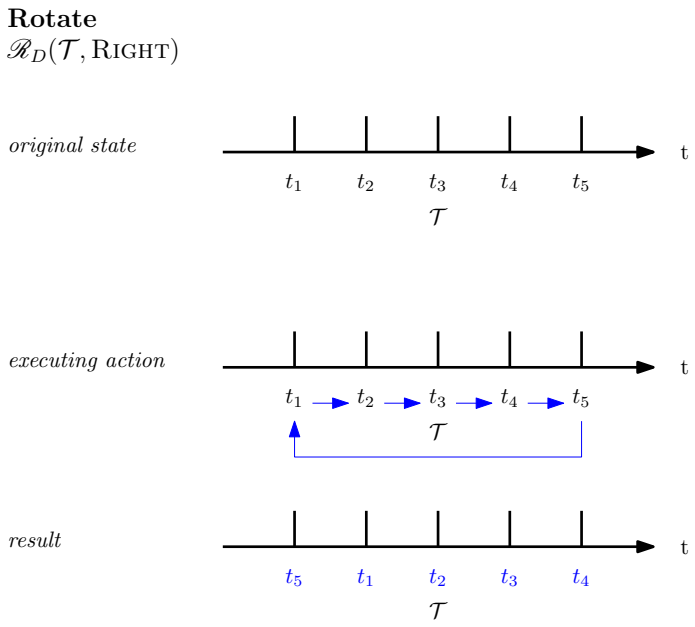


Fig. 6.14 Application of the *Rotate* operator, with direction *RIGHT* passed as parameter. Passing *LEFT* as parameter would shift the labels in the other direction.

Align

Signature: $\mathscr{A}_t(\mathcal{T}, t)$

Changes the *origin* of time domain $\mathcal{T}$ to time point t, re-labelling other time points *relative* to the new origin (see fig. 6.15). This effectively transforms the scale used in the time domain to a *relative* time scale. If the granularity used is a compound granularity (G+H), the finest granularity is used to calculate the distances.

6.3.4 Temporal extent and bounds operators

Operators in this category change the bounds of the time domain with two types of parameters: new *instants* mapped to the lower and upper bounds or using *duration* to modify the existing bounds. In this category, using *granularity* as a parameter is irrelevant, as modifying the bounds only make sense with quantities of time. This category of transformation corresponds to the *filter* category in Yi et al.'s taxonomy; narrowing the temporal extent is implemented in several visualisations (Carlis and

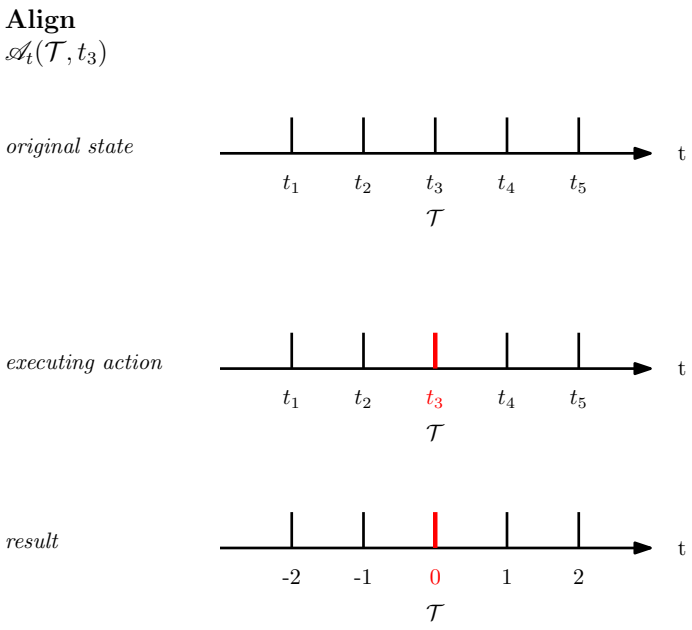


Fig. 6.15 Application of the *Align* operator, with instant t_3 passed as parameter.

Konstan, 1998; Ferreira et al., 2013; McLachlan et al., 2008; Sips et al., 2012), allowing the users to focus on interesting parts of the time domain, also reducing the number of time points displayed — depending on the visual encoding, this may also help to reduce clutter. Extending the time domain allows for reversing the trimming operation; in unaltered time domains, this transformation can be used in real time applications where new data points are expected to be included.

Trim At

Signature: $\mathscr{T}_t(\mathcal{T}, [t_1], [t_2])$

Changes the bounds of time domain $\mathcal{T}$ from $s_G < t_1$ to t_1 and $e_G > t_2$ to t_2 (see fig. 6.16). Either t_1 or t_2 can be optional. This transformation decreases the extent of the time domain with *instants* passed as parameter. Generally, this means that users choose the new bounds directly through interaction, although algorithms that choose the new bounds based on criteria relevant to the application can also be used.

Trim By

Signature: $\mathscr{T}_d(\mathcal{T}, [d_1], [d_2])$

Reduces the extent of the time domain $\mathcal{T}$ by durations $d_1 + d_2$ (see fig. 6.17). The new

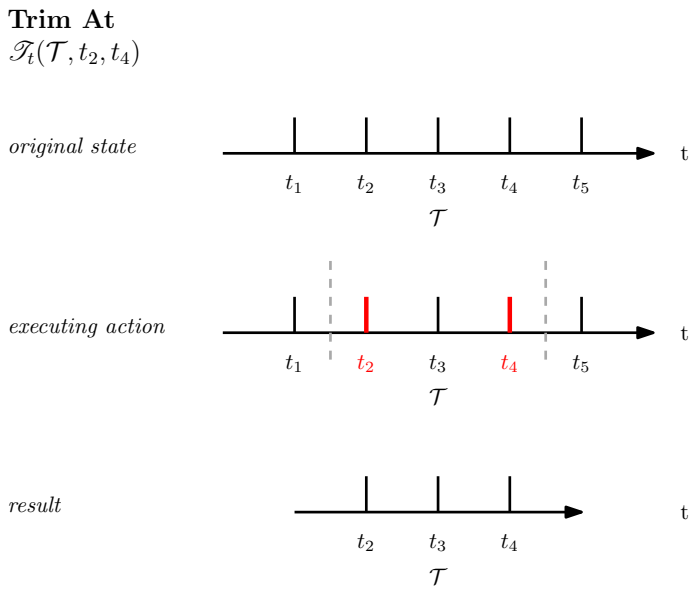


Fig. 6.16 Application of the *Trim At* operator, with instants t_2 and t_4 passed as parameters.

lower bound of the time domain is $s_G = s_G + d_1$ and the new upper bound of the time domain is $e_G = e_G - d_2$. Either d_1 or d_2 can be optional, or zero as default. The new temporal extent corresponds to $E - (d_1 + d_2)$. This operator is equivalent to *Trim At*, but with *duration* as parameter.

Extend To

Signature: $\mathscr{E}_t(\mathcal{T}, [t_1], [t_2])$

Changes the bounds of time domain $\mathcal{T}$ from $s_G > t_1$ to t_1 (see fig. 6.18), adding time points between t_1 and s_G such that $t_1 < t < s_G$, and $e_G > t_2$ to t_2, adding time points between e_G and t_2 such that $e_G < t < t_2$. Either t_1 or t_2 can be optional, or zero as default. This operator can be used to reverse the *trim* transformation by extending the time domain with either a new starting point or a new ending point.

Extend By

Signature: $\mathscr{E}_d(\mathcal{T}, [d_1], [d_2])$

Extends the extent of the time domain $\mathcal{T}$ by durations $d_1 + d_2$ (see fig. 6.19). The new lower bound of the time domain is $s_G = s_G - d_1$ and the new upper bound of the time domain is $e_G = e_G + d_2$. Either d_1 or d_2 can be optional, or zero as default. The new

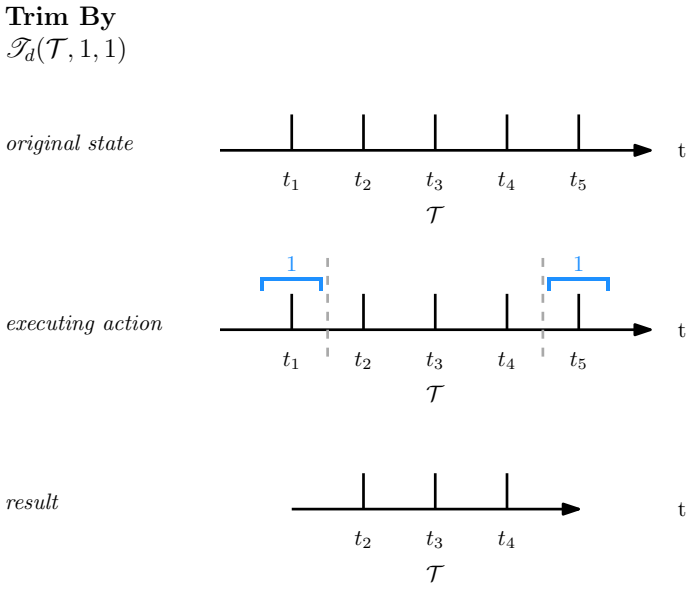


Fig. 6.17 Application of the *Trim By* operator, with durations 1 and 1 passed as parameters.

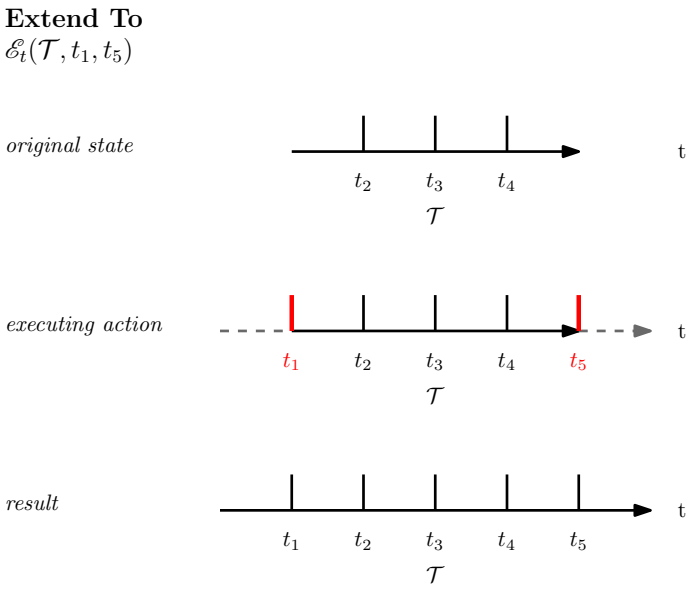


Fig. 6.18 Application of the *Extend To* operator, with instants t_1 and t_5 passed as parameters.

temporal extent corresponds to $E + (d_1 + d_2)$. This operator is equivalent to *Extend To*, but with duration as parameter.

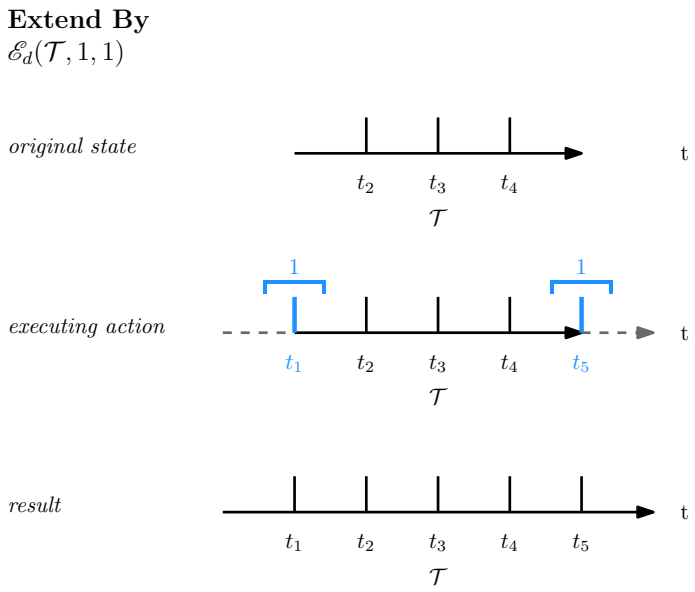


Fig. 6.19 Application of the *Extend By* operator, with durations 1 and 1 passed as parameters.

Compound granularities

In the case of *compound granularities*, such as YEAR-MONTH, the parameters in this category are assumed to be labeled by the finest granularity in the domain, MONTH in the example. However, the operators do not necessarily require such approach. It is possible, for example, to extend a YEAR-MONTH time domain by *12 months* as well as *1 year*. As the resulting time domains with the two types of parameters will be exactly the same, there is no difference between using a single or a multi-granular parameterisation.

6.4 Framework context

The suggested operators, summarised in table 6.3, have different purposes in the framework and can be combined in different ways with the other two components. A single *time domain* can be assigned as a temporal variable in the view and composition components. The concept of multiple domains and conditioning variables in the framework, however, requires the use of *identifiers* to map data items to different

Category	**Operator**	**Parameter variation**
Segmentation	Segment	At Instants
	”	By Duration
	”	Matching Granularity
	”	Relative To
	Join	-
Granularity	Bin	At Instants
	”	By Duration
	”	Relative To
	Expand	-
	Change Granularity	Granularity
	”	Roll Up
	”	Drill Down
	Rotate	-
	Align	-
Extent and bounds	Trim	At Instants
	”	By Duration
	Extend	To Instants
	”	By Duration

Table 6.3 Summary table of operators

segments. Consider the following example:

$$\mathcal{T} = \{2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011\}$$

This domain can be assigned both as a conditioning variable and position, resulting in the visualisation in fig. 6.20.

Fig. 6.20 Example of a visualisation of time domain $\mathcal{T}$.

Segmenting the domain by 4 years will result in two new domains $\mathcal{T}_1 = \{2004, 2005, 2006, 2007\}$ and $\mathcal{T}_2 = \{2008, 2009, 2010, 2011\}$; as two domains exist now, data items must be *mapped* to these domains. This can be done with an intermediate derived *segment* (or any other identifiable name) variable: data items containing the years belonging to segment $\mathcal{T}_1$ are assigned to segment 1, while the data items containing the years belonging to the second segment are assigned the respective value (2). The following specification can be used to define a visualisation (fig. 6.21) based on the new segments.

Segmented heat map					CARTESIAN,INSTANT				
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Segment**	–	–	–	-	-	**S**	SCALE	SPINE
2	**Year**	*nesting*	**A**	–	-	**Y**	-	SCALE	SPINE

Table 6.4 Specification of a visualisation of segmented a time domain.

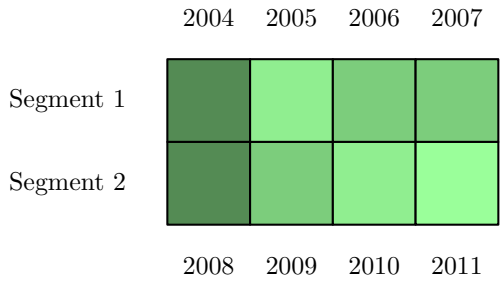


Fig. 6.21 Example of a visualisation showing two time domains as a result of segmentation corresponding to table 6.4.

In contrast with segmentation, the use of granularity and extent operators is straightforward as there is no need for intermediate variables. The following chapter details the use of each category of operators with the visual composition methods, as well as the interplay between other components of the framework.

6.5 Discussion

The principal point of discussion for this component is the number of combinations of categories and parameters. Considering the process of derivation of operators, which started with an analysis of the literature, the question is how *complete* is the component. However, this question requires defining the *criteria* for completeness. In terms of the categories and properties of time, the argument defended in this thesis is that the proposed operators are complete as that they cover the properties of time that relate to the *scope* of the framework – in that sense, the operators do not include any *non-temporal* aspect in the transformations, such as including conditions that depend on attributes.

Another angle to consider is the completeness regarding the *parameters*, which define the unique operators in each category besides the operators with varying functionality. The thesis presents operators considering the different types of time and properties that are used to *transform* time. These include *instants*, *duration* and *granularities*. As the other type of time, interval, is defined by two instants, it is included in the first category. Beyond these properties, there are also parameters that receive a *function* instead of specific time points or granularities. This idea derives from the use of *procedural parameters* (Slonneger and Kurtz, 1995), in which functions are passed as parameters to

other functions; in the component, it enables to *procedurally* define the instants that are used to apply operators following mathematical formulas or algorithms. The inclusion of this type of parameter generalises the use of logarithmic scale transformations, which is one specific type of scale in the design space by Brehmer et al. (2017), to any scale transformation that can be defined as a function with no auxiliary parameters.

As new visualisations are developed and include other transformations not covered by the operators, it is possible to expand the set of operators. However, the current set of operators satisfy the scope of the research, the survey done in chapter 6 and the applied methodology.

6.6 Chapter summary

This chapter presented conceptual transformations of the time domain based on interactions that are used in time-oriented data visualisations. These transformations are defined as operators that act on abstractions of the time domain, receiving different types of parameters that guide their functionality. The categories of operators reflect the properties of time that can be manipulated and interacted with, while the parameters are based on conceptualisation of time itself. The next chapter summarises the framework and relates the components to each other.

Chapter 7

Framework overview

The previous three chapters have introduced the components of the framework individually: the composition, the view and the transformation component. Each of them contains distinct parts that, together, allow hierarchical visualisations to be specified and interacted with. In this chapter, the interplay between each component are described in detail with examples of visualisations that can be reproduced by the framework and complex transformations that can be defined by combining operators with the other parts of the framework, allowing the *navigation* between different visualisation techniques.

7.1 Framework summary

The framework was briefly introduced in Chapter 3, with each component described in detail; the following section relates the components. The View component contains the properties that describe the coordinate system and layout of visual encodings and mappings from the data to visual variables; this includes description of *current* mappings and operators that allow the definition of the *new* mappings. The following properties are used:

- **Coordinate system**: polar or cartesian;
- **Shape**: point, bar, spine, ellipse or polygon;
- **Layout**: SCALE keyword, any supported algorithm or *empty*;
- **Position**: one or two space, time, attribute or identifier variables or *empty*;

- **Size**: one or two space, time or attribute variable or *empty*;
- **Path**: space, time or attribute variable or *empty*;
- **Colour**: space, time, attribute or identifier variable or *empty*;

The Composition component contains the definition of the *conditioning variable* for each level of the hierarchical structure, defining the underlying data model of the framework, as well as the *visual composition* for each level of the hierarchy. They are defined as the following properties:

- **Conditioning variable**: space, time, attribute, identifier or *same*, for *same level* composition;
- **Composition method**: juxtaposing, superimposing or nesting;

Finally, the Transformation component contains the operators that modify the time domain of the dataset being visualised, altering *properties* of time that can be used to modify the data space or trigger transformations of the visual space as well. These operators are:

- **Segmentation operators**: transformations that divide one time domain into multiple time domains or join segments;
- **Granularity operators**: transformations that change the level of abstraction of the time domain by aggregating time points or modifying the granularity used;
- **Extent operators**: transformations that change the extent of the time domain by extending it or narrowing it down;

The following sections describe how the Transformation component affect common visual encodings defined by the View component and can be combined with the visual composition methods defined in the Composition component. For both parts, the effects of the transformations are discussed, including the main perceptual effects in visual encodings and their relation to exploratory tasks. For the combination between the Transformation and Composition component, the *navigation* between visualisation techniques are explored as well.

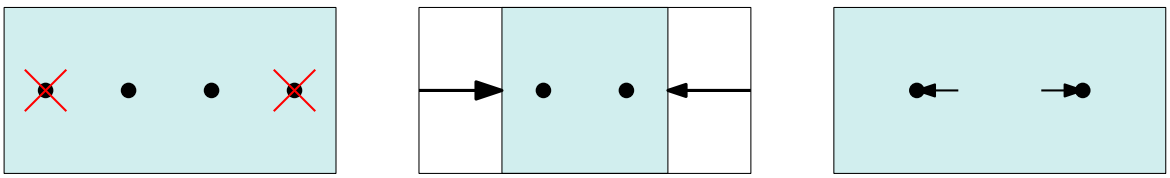
7.2 View and Transformation interplay

One of the purposes of the Transformation component is to provide *general* transformations that are independent from the visual encodings; this means that they do not directly modify these encodings; the changes in the time domain, however, indirectly modify the visual space. Extending the time domain, for example, results in more points being represented: depending on the encoding this might increase clutter. While the interpretation of the functionality of the operators also depends on the application, it is possible to discuss alternatives of these interpretations for the operators. This is covered in the following subsections.

7.2.1 Time mapped to Position

As discussed in chapter 5, mapping time to the *positional* channel is a common approach for interactive time visualisation. The effects of temporal transformations on this channel depend on how the other axis is used: if the other channel is used to display the *distribution* of events or objects over time through *identifier* variables or distribution of values over time through *attribute* or variables. For some trajectory visualisations, spatial variables are also combined with time.

The property of the time domain with the strongest influence on this channel is the number of time points, which can be altered by extent operators or by changing granularities. Reducing the number of points requires deciding what to do with the associated data points; various strategies exist depending on the type of data that is being displayed along with time. Regarding the time axis itself, the visual space can be reduced as needed (fig. 7.1 (a)) or remaining time points can be repositioned (fig. 7.1 (c)).



(a) (b) (c)

Fig. 7.1 Example of the effect of (a) trimming the time domain. In (b), the visual space is reduced, whereas in (c), the points are repositioned to keep the same scale.

For the other axis, the choice depends on various characteristics of the data and the encodings, such as the number of points that are being aggregated or the type of data. For quantitative variables, for example, if two points are aggregated into one after a transformation, a new derived attribute must be calculated and it will change the position of the point on the screen.

7.2.2 Time mapped to Size

Time is usually mapped to size through unanchored durations or anchored intervals. In the first case, duration is treated a normal quantitative attribute, which can be derived from the time domain or not – in the latter case, the operator will have no effect. If the duration is derived from the time domain, then changes in the granularity and bounds of the domain will also change the durations; effects of segmenting the time domain depend on the treatment of durations derived from instants that belong to different segments.

Changing the *granularities* used to derive the durations, within the framework, means that the durations have to be re-calculated, possibly resulting in a new range of durations. The design decisions discussed for position apply in the same way for size. In fig. 7.2 (b), the size of the single bar was calculated as the average of the previous two time points, whereas in fig. 7.2 (c) the size is assigned to the greatest value. The same principle applies to other types of variables; for categorical attributes, if the value of the attribute is different, the visualisation designer must choose the appropriate derivation for the new value.

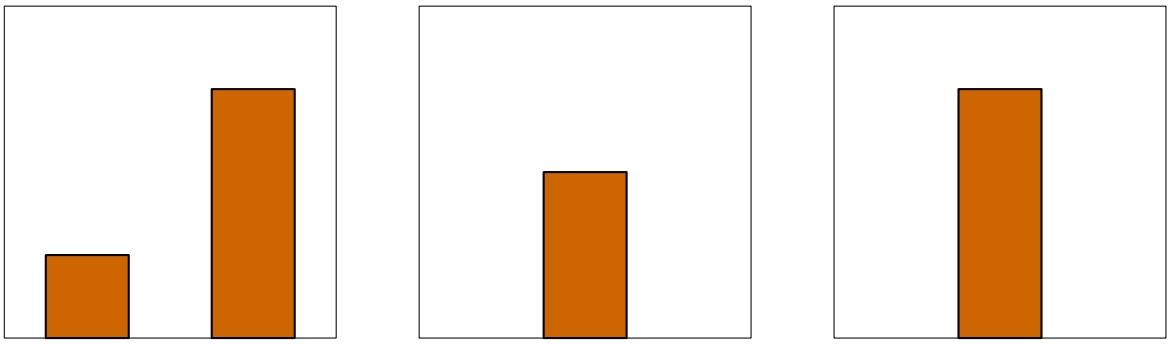


Fig. 7.2 Example of the effect of (a) trimming the time domain. In (b), the visual space is reduced, whereas in (c), the points are repositioned to keep the same scale.

7.2.3 Time mapped to Colour

Displaying time through the colour channel often means using a appropriate single hue colour scales based on the sequential nature of time. This means, in an HSL colour space, varying saturation and lightness. For only a few time points, however, if the objective is to simply *differentiate* the time points, with order not being of any particular importance, it is also possible to use a non-sequential colour scale.

An example of mapping time to colour is the time curve (Bach et al., 2016a) (see appendix A for the specification with the framework), which is a 2D projection of the temporal domain based on the similarity of time points (Figure 7.3a). In this visualisation, the spatial distribution of the time points does not reflect any temporal ordering. As there is not an intrinsic temporal component in the 2D projection, the authors mapped the temporal order to the colour channel and also to the criteria used to connect points.

In the original paper, the authors used a continuous colour scale spanning the whole temporal extent. With this scheme, while it is possible to identify the temporal ordering of events, it is difficult to identify the temporal location when events occur irregularly in time, as in the case of the curves in fig. 7.3a – even though line thickness indicates the length of the interval between two data points. One alternative is to change the granularity of the domain used for colouring, for example, by *rolling up* to a coarser granularity, such as in fig. 7.3b. In this case, one of the factors to consider when choosing the appropriate granularity is that the colour change might suggest an interruption during a burst of activity.

7.3 Composition and Transformation interplay

The previous section covered the cases where time is mapped to a visual channel, describing the results of transformations of the time domain with the operators. Time can also be used, however, in the hierarchical structure as a conditioning variable. This section covers two aspects of the interplay between using time with the composition component and modifying it with the transformation component. The first is the use of the abstraction of the time domain to construct the hierarchy through conditioning; the second is the combination of visual composition methods with the operators to navigate the design space of the visualisation techniques supported by the framework. The latter is related to generating *multiple perspectives* on the data and thus exploratory

(a) (b)

Fig. 7.3 Example of mapping time to colour. On the left side, time is mapped to a continuous colour scale; on the right, the granularity used for colour has been *rolled up* to a coarser unit (year), resulting in three distinct periods. For the second figure, the colour is used to identify the periods rather than convey the temporal order.

data analysis; for that reason, the navigation is discussed in the context of temporal tasks. While the number of paths between visualisations is large, plausible examples will be given based on the techniques found in the literature.

7.3.1 Conditioning with temporal variables

In chapter 6, the operators were described using an abstraction of the time domain that contains properties of time that are derived from a set of temporal references and transformed. With tabular data as source, for example, this is usually a *timestamp* or similar column. On the other hand, the hierarchical structure, as described in chapter 4, is based on assigning *variables* to generate conditioned aggregations of data items, matching the values of these variables. The combination of the transformation and composition components allow the abstraction of the time domain to be used in the conditioning process and transformed accordingly. A time domain can be *configured* to match a variable used for conditioning and the operators enable the transformation of the time domain to be reflected visually. One example is initialising a time domain with *Year* as the granularity, use a corresponding variable for conditioning and use operators

to manipulate it, such as changing the granularity to *Month*. The following section describes how the visual composition methods can be used with these transformations.

7.3.2 Composition with segmentation operators

Segmentation operators have great synergy with *between level* composition: the transformation enables interactive reconfiguration of hierarchies in order to emphasise or compare sub-divisions of the time domain. In terms of temporal tasks, it facilitates looking for *temporal patterns*, comparing *rates of change* and *sequence* between different segments. Segmentation results in several abstractions of the time domain; as briefly described in the previous chapter, data items must be *assigned* to these different abstractions, which can be done by including a new *identifier* variable. This new identifier variable can be used to *condition* the data items via an *intermediate* level, thus allowing the new segments to be used with other variables and to define compositions, as in fig. 7.4.

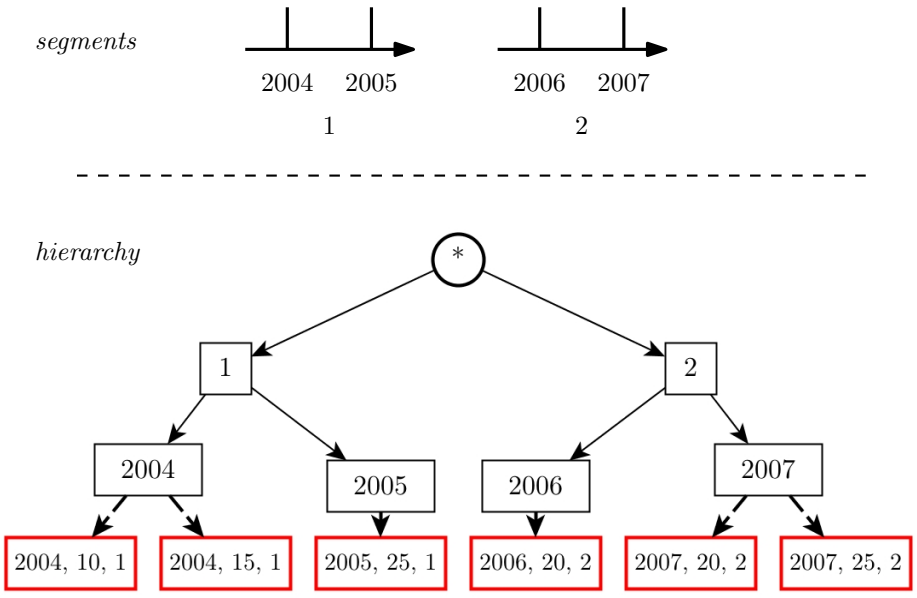


Fig. 7.4 Example of conditioning after segmenting the time domain. A new identifier variable was added to each record to mark the segments.

Segment + Juxtapose Between Levels
Signature: (**time**,E_1) → J((**segment**,E_1),(**time**,E_2))

Segmentation followed by juxatposition (fig. 7.5) allows configuring an *overview* of the segments separated from a *detailed* view of the segments. Among the design choices when using this transformation is the encoding used for both views: the original encoding can be repeated by default for the two views, preserved for the second level or preserved for the first level. In the context of tasks related to temporal data exploration, the combination enables answering questions through summarisation of attributes in the first level and individually for each segment in the second level.

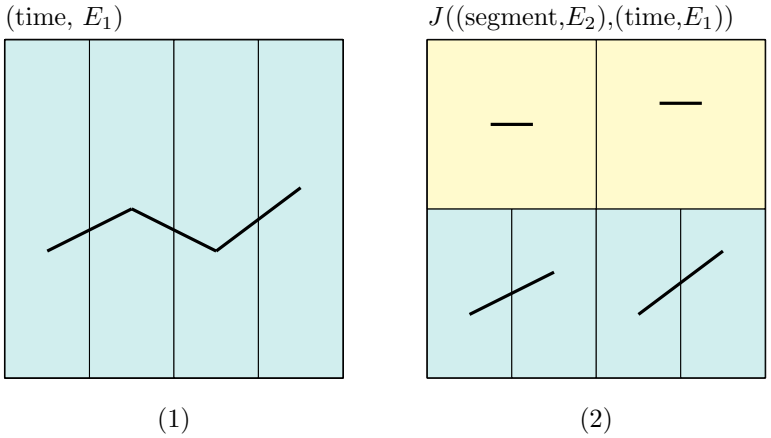


Fig. 7.5 Example of segmenting and juxtaposing between levels.

Segment + Superimposition Between Levels
Signature: (**time**,E_1) → S((**segment**,E_1),(**time**,E_2))

Segmentation with superimposition combines summary and individual attributes in the same *area* of the visualisation. Its effectiveness depends on the choice of encodings due to the overlap; this composition is equivalent to superimposing two levels conditioned by adjacent time granularities with a divided time domain. Rather than displaying a global overview of the whole temporal domain of the dataset, overviews of each segment are displayed separately. In fig. 7.6, (2) contains averages of each segment as horizontal lines inside spines, with the lines of the second level *superimposed* over them.

Segment + Nesting Between Levels
Signature: (**time**,E_1) → N((**segment**,E_1),(**time**,E_2))

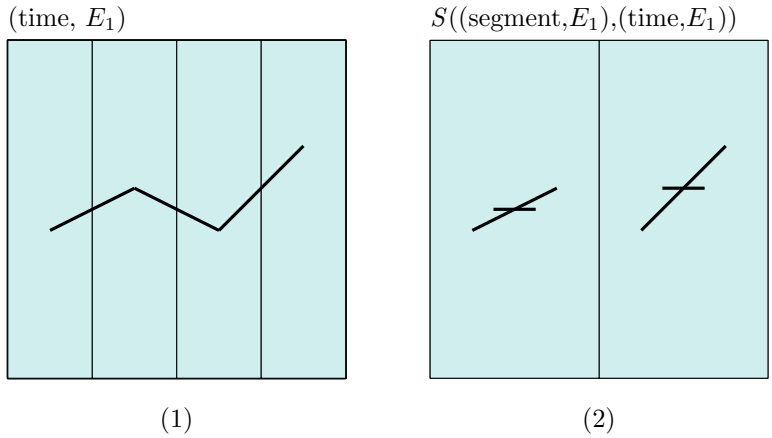


Fig. 7.6 Example of segmenting and superimposing between levels.

In contrast with juxtaposing and superimposing, nesting does not preserve the context of the overview. The composition isolates the segments and allows the application of derived attributes from each segment into the parent level shapes, such as resizing or recolouring shapes and segments. The combination with other transformations, such as reducing the extent, allows closer inspection of segments. Further compositions allow the *replication* of more complex arrangements such as the stack zooming interface by Javed and Elmqvist (2013).

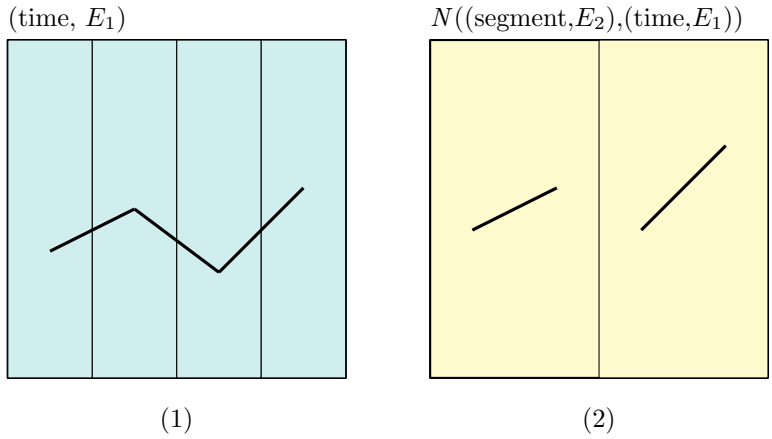


Fig. 7.7 Example of segmenting and nesting between levels.

7.3.3 Composition with granularity operators

This combination realises one of the aims of providing *different perspectives* on the data: the hierarchical structure can be modified to reflect multiple temporal scales enabling encodings to match the characteristic of these scales and tasks related to these scales. Within the category of granularity operators, binning and granularity changing operators are suitable for this approach. The use of *rotate* and *align* operators may be necessary to *adjust* encodings after transformations.

Bin/Change Granularity + Juxtapose Between Levels
Signature: $(G,E_1) \rightarrow J((G,E_1),(H,E_2))$

Changing the level of aggregation while juxtaposing (fig. 7.8) allows different levels to be compared in separate views. Juxtaposing while drilling down or rolling up time allows the visual ordering of levels to reflect the granularity hierarchy, whereas changing to specific granularities may not have the same effect.

Bin/Change Granularity + Superimpose Between Levels
Signature: $(G,E_1) \rightarrow S((G,E_1),(H,E_2))$

This combination allows the use of visual marks to be layered over each other, enabling multiple scales to be compared on the same view. In comparison with juxtaposing, superimposition allows a larger rendering area to be used, although with the risk of increasing clutter and occlusion. Figure 7.9 shows an example where lines are drawn over a bar after the granularity has been changed; the bar encodes the *average* value for the time point, which is expanded into two time points in the line chart.

Bin/Change Granularity + Nesting Between Levels
Signature: $(G,E_1) \rightarrow N((G,E_2),(H,E_1))$

This last configuration is the basis for calendar and grid-based visualisations. The granularity of an unmodified linear time domain can be changed, with each level being *nested* within each other. Figure 7.10 shows how successive applications of the drill down operator along with adding a new nested level to the hierarchy results in a grid arrangement.

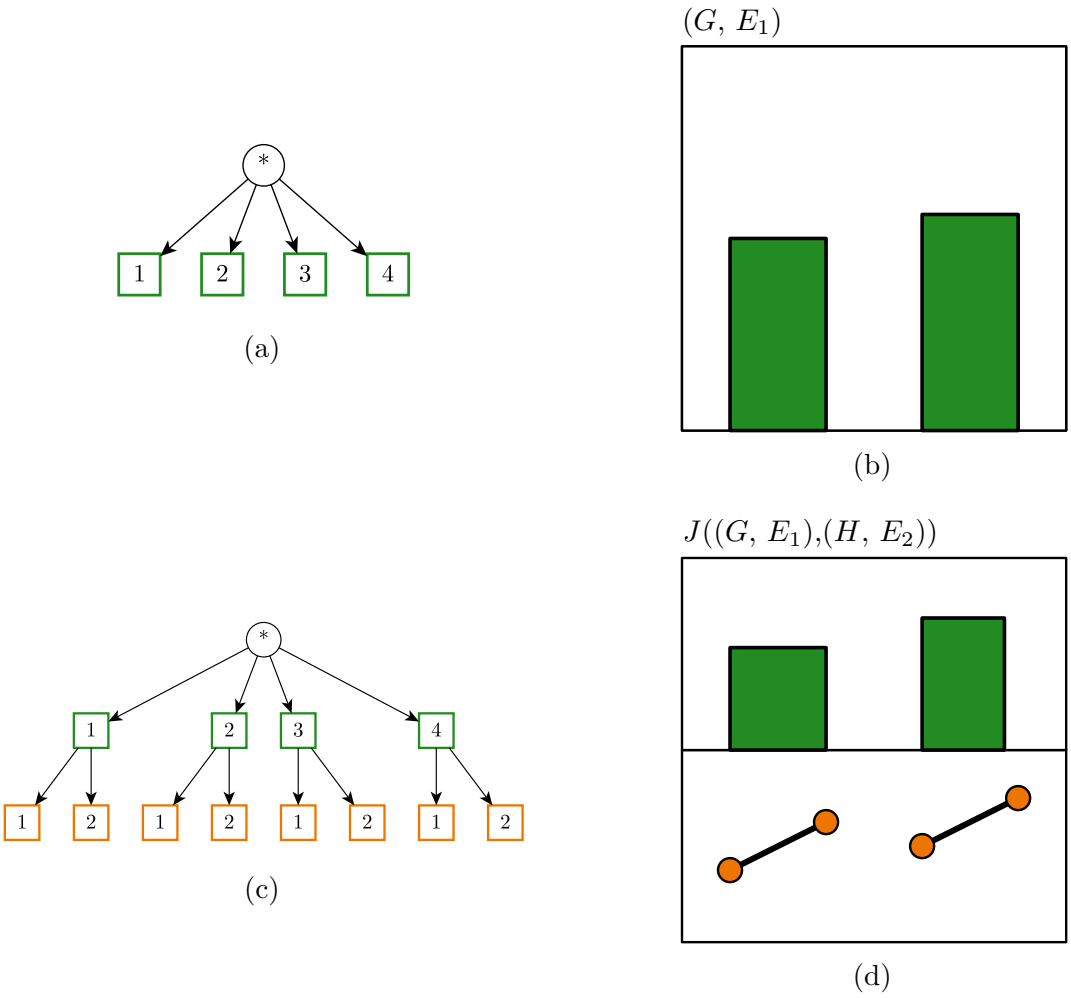


Fig. 7.8 Example of changing granularity and juxtaposing, with the colours of the shapes matching the nodes in the trees. In the second row, the tree has been *drilled down* to a finer granularity H, which is encoded as circles connected by a path and *juxtaposed* below the first level.

7.4 Chapter summary

This chapter described how the components are connected with each other under the unified framework. The effects of the temporal transformations on the view component were analysed; for the combination of the transformations and composition methods, it was explained how the interplay between the two components empowers the visual exploration driven by the various transformations of time. Various examples were given demonstrating the visual result of the changes in the hierarchical structure as a result

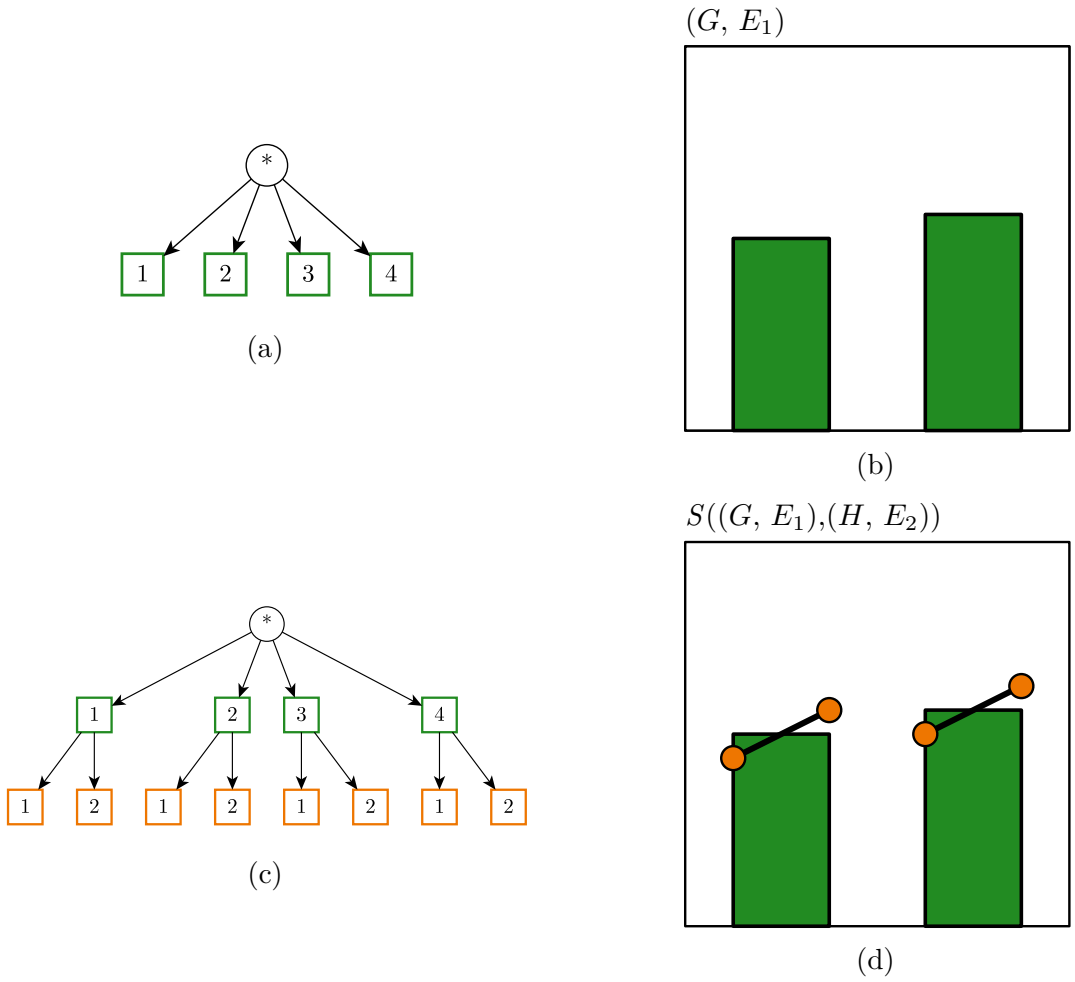


Fig. 7.9 Example of changing granularity and superimposing, with the colours of the shapes matching the nodes in the trees. In the second row, the tree has been *drilled down* to a finer granularity H, which is encoded as circles connected by a path and superimposed over the first level.

of the transformations. The next chapter describes in further detail the *paths* that can be explored in the design space that emerges from the framework.

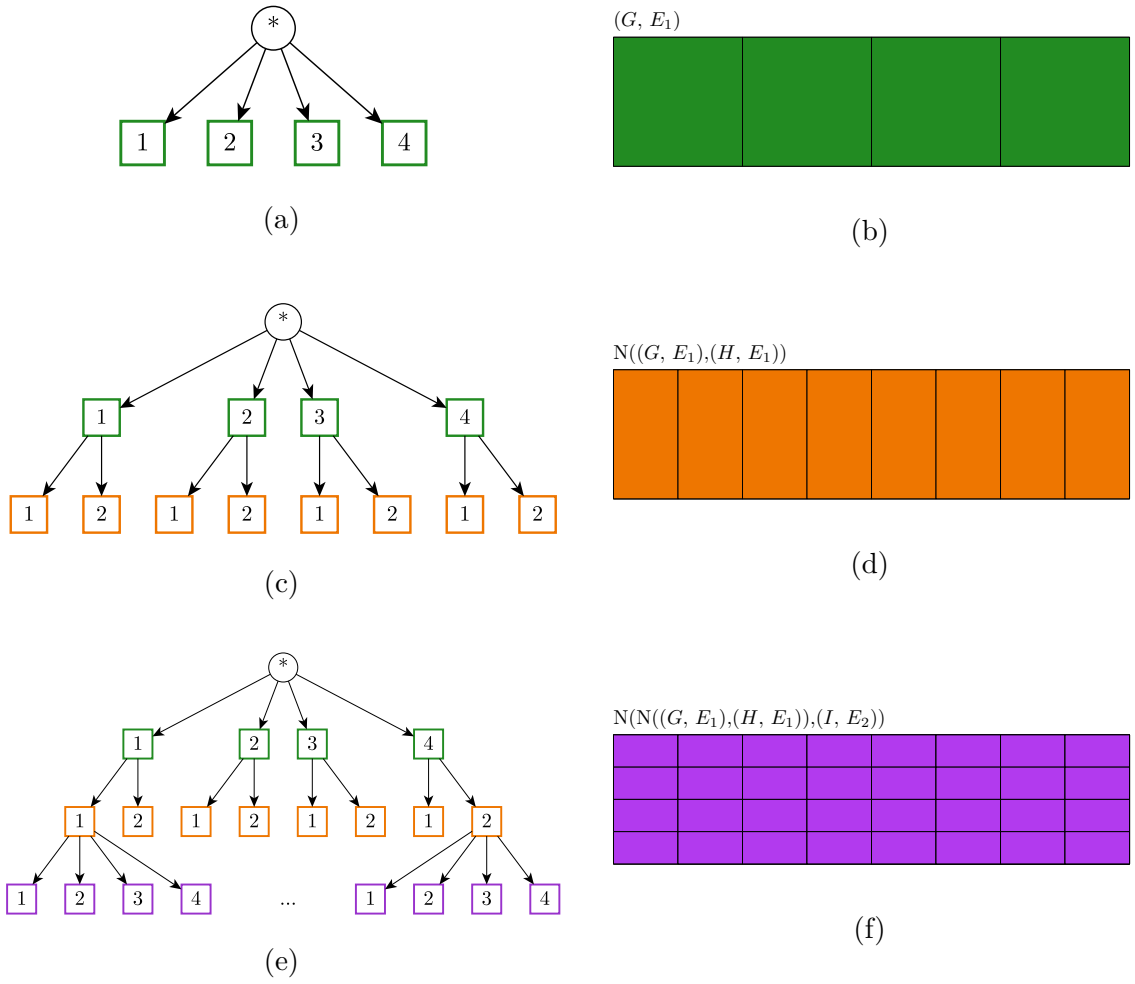


Fig. 7.10 Example of changing granularity and nesting, with the colours of the shapes matching the nodes in the trees. In the second row, the tree has been *drilled down* to a finer granularity H, with the corresponding view repeating the same encoding. In the third row, the tree has been *drilled down* to a finer granularity I; this time, the corresponding view uses encoding E_2 with the orientation of the spines changed to vertical, resulting in the grid visualisation.

Chapter 8

Case study

This chapter demonstrates how the framework supports exploring temporal data through hierarchical visualisations. Questions and data from a visualisation challenge are used to explain how the different aspects of the framework support the design of *multiple perspectives*, as part of an EDA approach to solve the challenge. The aim of this chapter is not to discuss the best configurations for the tasks, but demonstrating the *design process* and the possibilities that the framework enables regarding its three components.

8.1 The dataset

The VAST Challenge[1] is a traditional annual data and visualisation challenge, in which participants design interactive visualisation tools for use in a fictional scenario inspired by real-world datasets and tasks, including fictional locations, characters and data. As part of the 2017 challenge, the setting involved the analysis of temporal patterns for vehicles that roam through a nature reserve with electronic gates that record their passage. Vehicles are assigned ID's and a vehicle type; the gates include exit and entrance gates from the reserve and other types of gates, including camping areas, locations where park rangers scout and other vehicles are not allowed to pass through and the park rangers headquarters. The dataset contains *timestamped* records of the vehicles – the challenge is to design visualisations that help solve challenge. The list of attributes for each record is as follows:

[1] http://vacommunity.org/VAST+Challenge+2017+MC1

- **Timestamp**: including year, month, day, hour, minutes and seconds granularities;
- **Car-id**: a unique car *identifier* that is assigned to every vehicle that enters the reserve and is reused for subsequent visits;
- **Car-type**: classification of vehicles into 7 numeric identifiers: 1 for car or motorcycle, 2 for two-axle trucks, 3 for three-axle trucks, 4 for four-axle and above trucks, 5 for for two-axle buses and 6 for three-axle buses. Additionally, park rangers vehicles have a *P* suffix; only two two-axle trucks are used, as such, park ranger vehicles are identified as *2P*. A general identifier *car-category* can be extracted from this variable, with a category encompassing every type of truck or bus, for example.
- **Gate-name**: identifiers for the following gate types: *entrance*, *gate*, *general-gate*, *ranger-stop*, *camping* and *ranger-base*. Each unique gate is given a numeric suffix, resulting in *entrance0*, *general-gate1*, etc. A general identifier *gate-type* can be derived from this variable by removing the numeric suffix.

The following three questions that are asked in the challenge are suitable for an exploratory approach and will be investigated in this chapter: (1) describing *daily* patterns of vehicles, including where they happen in the park, when they happen during the day and a hypothesis of what the pattern means (for example, vehicles simply passing through the reserve); (2) describing *multi-scale* patterns of vehicles, such as periodic visits by groups of vehicles at the same location and (3) describing unusual patterns, such as vehicles passing through gates that they are not allowed or patterns that are difficult to *hypothesise* given expected behaviour of vehicles, such as long stop-overs in the reserve without visiting a campsite. Due to the size of the dataset, throughout the chapter subsets of the data will be selected for the visualisations. This leaves out important visualisation and interaction decisions about reducing the number of items to be visualised; the aim of the chapter is demonstrating the use of the framework for the tasks rather than designing an interactive application that would help solve the challenge. The author has not participated in the challenge.

8.2 Applying the framework

8.2.1 Relating conditioning variables to tasks

Andrienko and Andrienko (2006) described the relationship between tasks and data as two levels of tasks: elementary and synoptic. *Elementary* tasks refer to the analysis of references individually: in the challenge dataset, this means looking at what happens at a specific day or the *individual* places that a vehicle visited. This also refers to a *group* of vehicles being analysed from an *individual* point-of-view, such as *one* category of vehicles that has never passed by a certain gate. This type of task, on its own, does not *require* visual exploration: they are usually questions that can be extracted using database queries, for instance.

Synoptic tasks refer to the analysis of *groups* of references and characteristics; an example would be looking at the *sequence* of gates that a *car-id* passed through in every day of a specific month. Tasks in this category are suitable for visual exploration as they generally consider multiple references and attributes and require *analysis* of the data rather than simple extraction of values. They can, however, be broken down into questions at *elementary* level. For instance, the following sequence of questions can be examined to define a pattern:

- **Elementary level** On which days of the week were *camping* gates visited?
- **Elementary level** For which vehicles was *camping* the last type of gate visited on the last hour of the day?
- **Synoptic level** What is the sequence of gates that vehicles of type 1 visited from any *entrance* to *camping*, when *camping* was the last visited type of gate on any day?
- **Synoptic level** In what months did the previous sequence occur consecutively?

This sequence of questions can be used, for example, to characterise a pattern of vehicles of type 1 that visit the reserve for camping during summer months, including the gates that they passed through before arriving at the campsites. The first two elementary questions can *guide* the exploration in order to answer the questions at a synoptic level, with the former allowing *filtering* data items that are not required for the latter. The hierarchical structure can be configured to reflect the sets of references that must be examined to answer the questions, based on the *conditions* that are being sought for and the *target* of each question. In the given example, the first question

considers: *gate-name* as condition, *day of week* and *car-id* as targets. Conditioning variables in the following order enables the hierarchy to reflect this: *car-id*, *gate-name* and *day of week*. A corresponding visualisation would show the days of the week that *each* gate was visited for *each* vehicle identifier in the dataset.

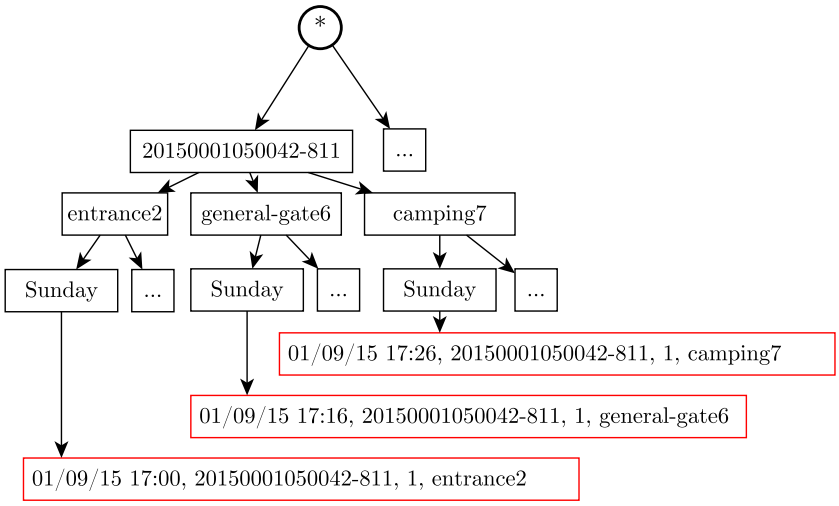


Fig. 8.1 Conditioned tree for *car-id*, *gate-name* and *day of week*.

In the second question, the aim is to identify the vehicles for which the condition *camping as last gate of the day* is valid. To answer it, the hierarchy can be reconfigured with: *car-id*, *hour*, *minute* and *gate-name*. A corresponding visualisation would show, for *each* unique *vehicle-id*, the *gate-names* that were visited for *each hour* and *minute*, which allows identifying the satisfying condition for the task.

Moving from elementary to synoptic levels, an important consideration is about the *aggregation* that results from certain configurations of the hierarchy. As mentioned, elementary tasks considers groups as *individual* elements; for this reason, the configuration for the second hierarchy, which aggregates every timestamped record by the hour of day, is suitable for this task. For the first synoptic task, however, records for the gate must be inspected in temporal order: aggregating them by *hour of day* does not allow the sequence of gates to be described. For this task, the temporal variable must be configured *linearly* with a compound granularity (*day-hour-minutes*) or with multiple temporal levels (*day of week*, *hour* and *minute*). The question also includes part of the condition of the second elementary question: *when camping was the last visited gate of a day*.

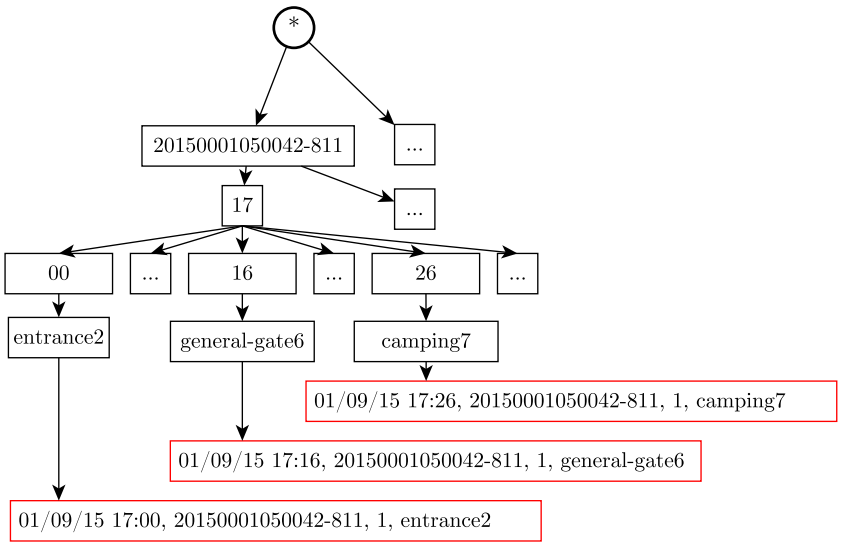


Fig. 8.2 Conditioned tree for *car-id*, *hour*, minute and *gate-name*.

The synoptic question can be broken down into the following parts: sequence of *individual* gate names, *individual* vehicles of type 1, *entrance* gate type, *camping gate type*, *individual last visited gate of the day*. The hierarchy structure can be organised to facilitate the comparison between the elements with the following order: *car-id*, *day of week*, *hours-minutes* and *gate-name*. Figure 8.3 shows the resulting hierarchy for a selected car-id, with the sequence of gates in the last level inner nodes and the individual records in the leaf level.

Configuring the hierarchical structure allows defining the appropriate levels of aggregation of the references required in the different tasks. It is the first step in the generation of visualisations; with the hierarchy defined, the next step is to specify the visual encodings and composition methods that allow examining the *characteristics* associated with these references and answering the questions.

8.2.2 Specifying the visual encodings

Appropriate encodings can be specified by analysing the characteristics of data items that are required for the tasks. In the challenge's example, the dataset consists of records of visited locations. With no measured attributes, characteristics of such data include summary statistics and the distribution of these events in relation to the references, as described in the previous section. This means counting event occurrences

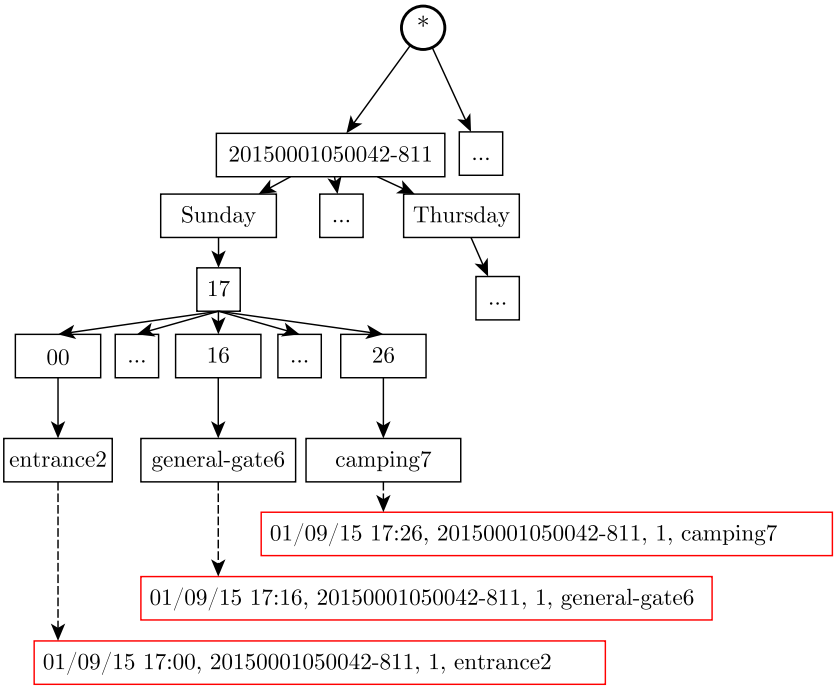


Fig. 8.3 Conditioned tree for *car-id*, *day of week*, *hour*, minute and *gate-name*.

and the attributes that can be derived from it: averages, minimum and maximum counts, sums across categories, etc.

Compared to *elementary tasks*, which can be answered using automated methods, specifying a visualisation for synoptic tasks is more challenging. Considering the hierarchy in fig. 8.3 and the targets of the task, the first design decision for an encoding is to enable the *identification* and *comparison* of *car-ids*. As the nodes in the first level aggregate records per car, a spine shape enables partitioning the space equally for each car. At this point, the order of the spines is not important.

This specification covers the first three levels of the hierarchy, partitioning the space by vehicles and then *nesting* and partitioning the space by the temporal variables, resulting in a heat map arrangement. The last level contains the gates that were visited by each car. With the lowest granularity being *minutes*, it is unlikely that a vehicle passed through more than one gate on the same minute. In this case, the encoding for the the last level can be defined as a *within level* superimposition of spine shapes, coloured by *gate-type*. The use of an opacity would even allow identifying cases in which a vehicle indeed passed through more than a gate at the same minute, resulting in unexpected colours. The final specification for this is the following:

			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**car-id**	–	–	–	-	-	**car-id**	SCALE	SPINE
2	**day-of-week**	*nesting*	–	–	-	**dow**	-	SCALE	SPINE
3	**hour**	*nesting*	–	–	-	-	**h**	SCALE	SPINE
4	**minute**	*nesting*	–	–	-	**m**	-	SCALE	SPINE
5	**gate-name**	*nesting*	**gate-type**	–	-	-	-	-	SPINE

8.2.3 Transforming time

The mappings from data variables to visual channels assume that initial data transformations have already been done. Temporal variables can be modified using the transformations component, while other variables can be transformed as required, such as filtering values of *car-id*. The transformation operators can either be used when designing a visualisation or included as interactions. In the example discussed in this chapter, the use of operators as interaction is discussed simply as a method for exploratory analysis; it will not include discussion about metaphors and interfaces.

Determining which transformations are needed requires analysing the dataset and encoding specification in section 8.2.2. For this, it is also important to consider that, although the framework does not take into account rendering aspects such as screen size, it is nonetheless possible to apply some data transformations based on certain assumptions. As the specified visualisation is based on partitioning the visual space with spines, each car is to be assigned an equal partition. Assuming a normal number of vertical pixels such as 1080, the 18710 unique car-ids in the dataset require the use of filtering and aggregation as a visual requirement that is not related to the the tasks.

In the temporal aspect, the records have an extent of *1 year and 1 month*, starting from *01/01/2015* and ending on *31/05/2016*. The visualisation is configured to use three granularities: day of week, hours and minutes. All granularities are *cyclic*, but, for the task at hand, this does not affect the visualisation. Each granularity can be

assigned a time domain abstraction:

$$\mathcal{T}_{\text{DayOfWeek}} = \{\text{Sunday}_{\text{DayOfWeek}}, \text{Saturday}_{\text{DayOfWeek}}, \text{DayOfWeek}, 7, \{Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday\}\}$$

$$\mathcal{T}_{\text{Hour}} = \{00_{\text{Hour}}, 23_{\text{Hour}}, \text{Hour}, 24, \{00, \ldots, 23\}\}$$

$$\mathcal{T}_{\text{Minute}} = \{00_{\text{Minute}}, 59_{\text{Minute}}, \text{Minute}, 60, \{00, \ldots, 59\}\}$$

Based on the characteristics of the dataset and the encodings used, at this stage no further transformations of time are required for an initial overview. For the actual exploration when implementing the visualisation, segmentation and granularity operators are not generally needed – the time domains are already set up appropriately for the visualisation configuration. Extent operators, however, might help with identifying patterns. For instance, trimming the hour domain to *00-12* enables the analysis of morning daily patterns with possibly more clarity, as each visual mark can occupy more space on the screen.

8.3 Visual exploration

The previous section described how each part of the framework is used when visually exploring data. The rest of this chapter details the steps to answer the outlined tasks following the same idea: analysing how the hierarchical structure relates to the task, the appropriate encodings to visualise such data and the temporal transformations that facilitate the exploration process. In order to display visualisations without distortions, the dataset was *trimmed* by a month; the original data has records for both May 2015 and May 2016; the last month of the dataset was removed.

The two challenge tasks are closely related and differ by the temporal scale where activity patterns are investigated. The first task asked participants to describe *daily* patterns of vehicles, including where they happen in the park, when they happen during the day and a hypothesis of what the pattern means (for example, vehicles simply passing through the reserve). The second task asks for longer-scale patterns. As mentioned in the previous section, the dataset contains 18710 unique vehicles across 7 distinct vehicle categories; the reserve contains 40 unique gates across 6 gate types.

Considering this, several hypotheses can be examined for plausible patterns to begin the exploratory process.

1. Vehicle type patterns: distinct vehicle categories might have varying patterns, such as trucks *mostly* passing through the reserve and cars and motorbikes staying for longer periods;
2. Period of day patterns: early morning and late afternoon commuters might have simple reserve pass-through patterns, whereas visitors might arrive early and leave late in the day. These patterns might be applicable to different vehicle types as well;
3. Gate type patterns: *camping* gates are likely to be visited by cars and motorbikes more than by other types of vehicles, *ranger stops* are likely to be visited *only* by rangers;

Starting with different vehicles types, the first step in the exploration will be defining an overview of the dataset in relation to the types of vehicle and records of these vehicles over the hours of day. The first hierarchical level is conditioned by *Group*, the variable derived from the different types of vehicle; the specification assigned to the level partitions the space vertically using equal sized spines. The second level is conditioned by a time domain configured to *Hour*, with the number of vehicles shown as the *count* attribute assigned to a *bar* height. The visualisation is specified in table 8.1 and displayed in fig. 8.4. The following conclusions can be drawn:

- Cars and motorbikes are the largest category of vehicles passing through the reserve, with frequent activity from 6:00 to 17:00;
- Rangers largely follow working hours with near absence of patrolling in the middle of the night;
- Truck activity also happens more frequently from 6:00 to 17:00;
- Bus activity is generally low throughout the day and night;

One transformation that can be applied to this visualisation is rotating the set of hours to see working hours clearer. For this case study, approximate business hours seem to be a good indicator of activity. Rotating by *-5* sets hour 5 as the starting hour; the resulting visualisation in fig. 8.5 shows the peak of ranger activity and the longer tail that goes until the night. The sudden drops in activity of cars/motorbikes and trucks is also more visible.

Bar chart			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Group**	–	–	–	-	-	**Group**	SCALE	SPINE
2	**Hour**	*N*	–	[–, **COUNT**]	-	**Hour**	**COUNT**	SCALE	BAR

Table 8.1 Specification for fig. 8.4. Signature: $N((\textbf{Group},L_1),(\textbf{Hour},L_2))$

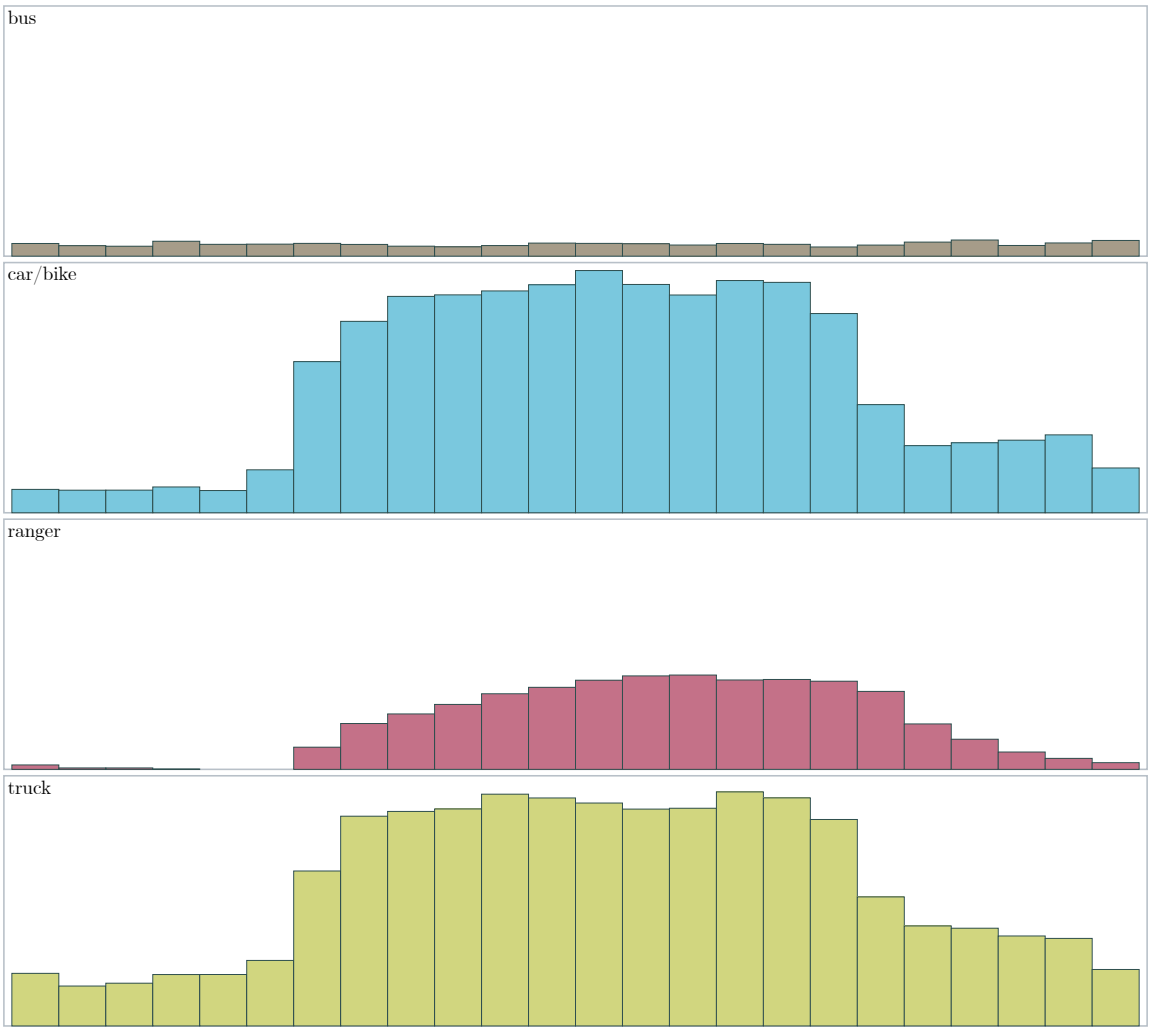


Fig. 8.4 Overview of $N(\textbf{Group},L_1),(\textbf{Hour},L_2))$. buses, cars/motorbikes, rangers, trucks

Several exploration paths that can be taken from this configuration; an interesting path is to relate global level activity to gate level, that is, analysing how each vehicle category *uses* the reserve. To visualise this, two steps are taken: (a) modifying the

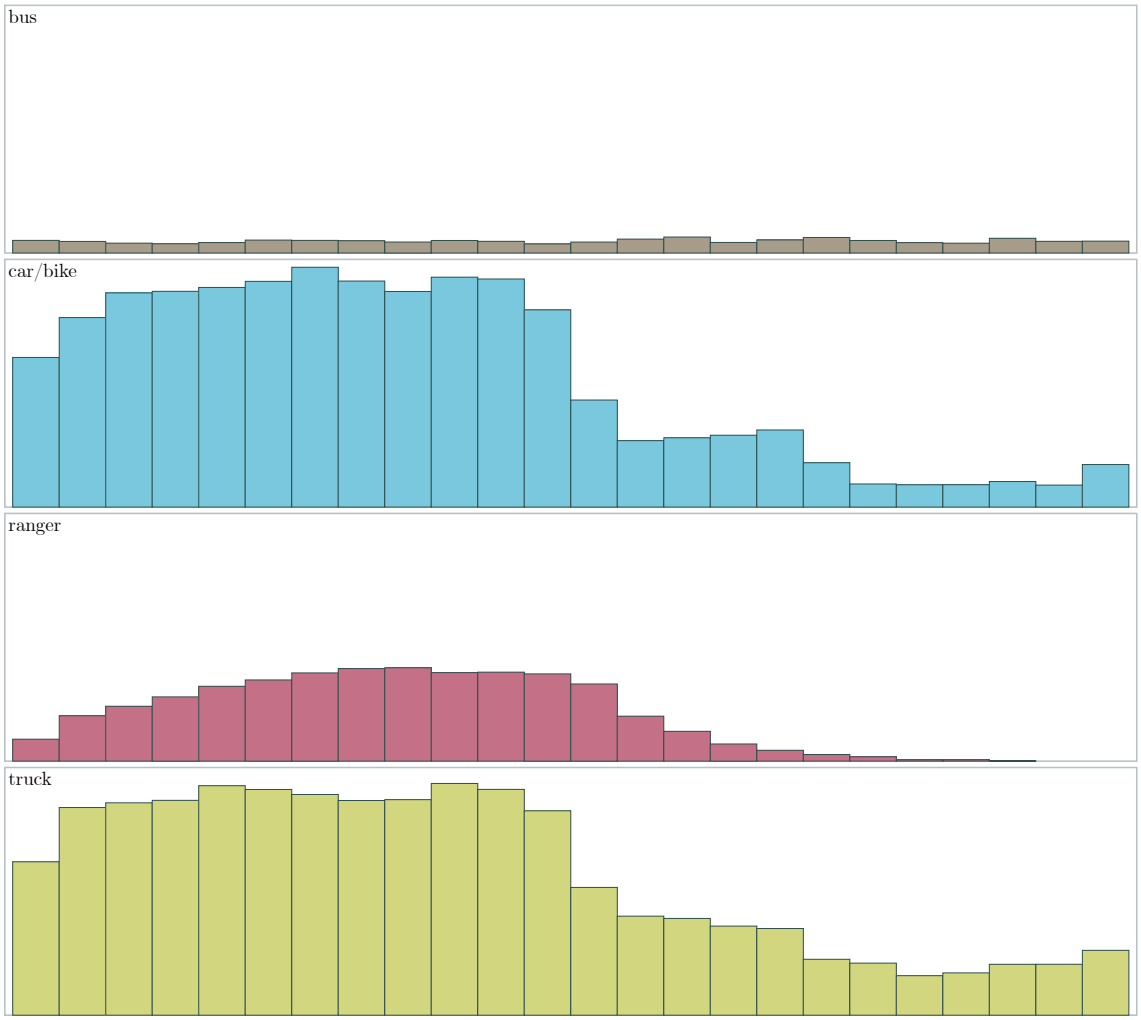


Fig. 8.5 Overview of N(**Group**,L_1),(**Hour**,L_2)) with rotated hours. buses, cars/motorbikes, rangers, trucks

specification used for the *Hour* level, from a bar shape to a spine shape with equally divided areas and (b) adding a new conditioning level for *GateType*. As the aim here is not to have a detailed comparison of each individual activity, a spine shape is used again to partition the space. As the number of records greatly vary per day, the visualisation (fig. 8.6) simply shows the types of gate that were visited in each hour.

An interesting fact that can be drawn from the visualisation is that some trucks appear to be passing through gates that they should not: this is identified in the purple spines in the lower left area of the visualisation. The other type of restricted gate, the ranger base, does not show unusual activity. The chart also shows the daily activity for the

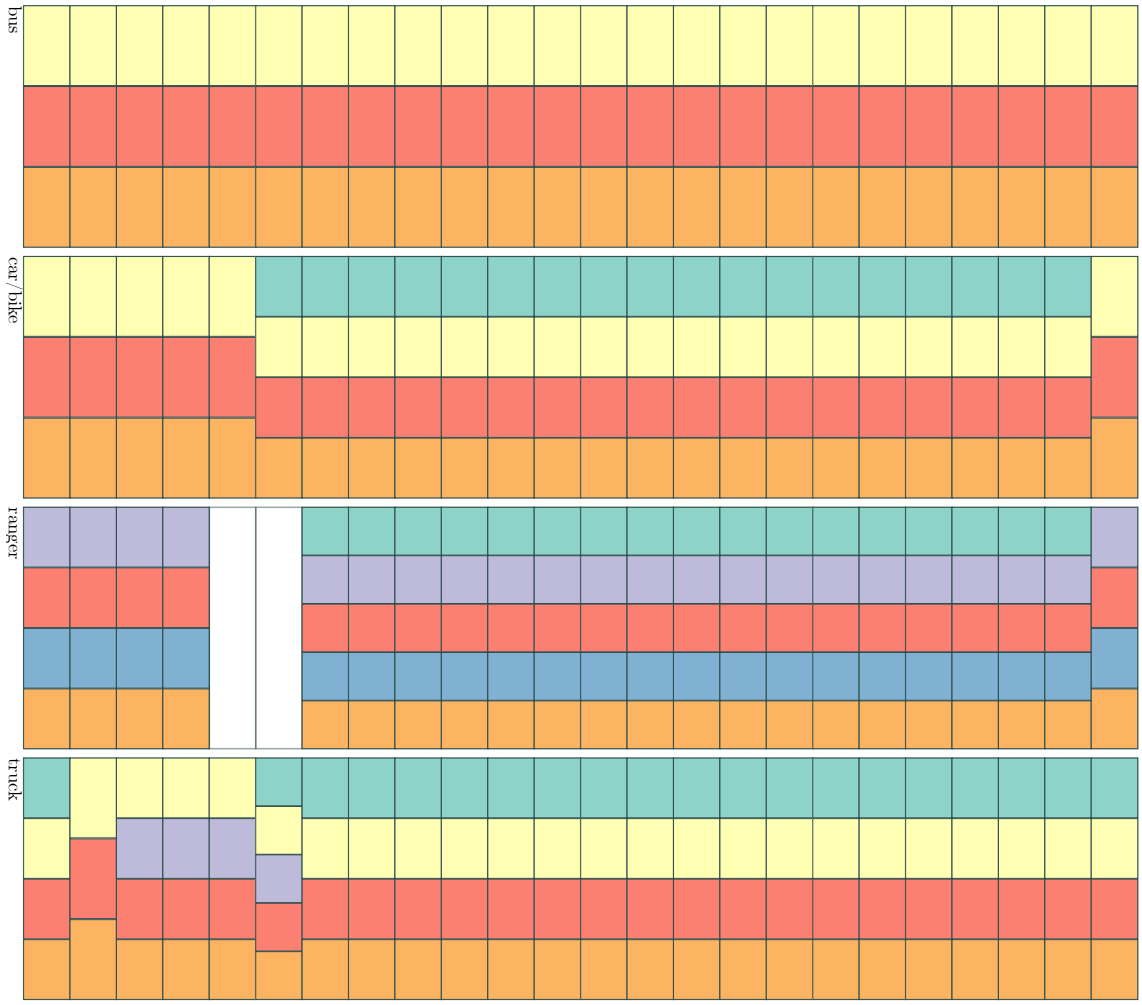


Fig. 8.6 Distribution of individual gate usage per hour per vehicle group. Signature: $N(\mathbf{Group},L_1),N((\mathbf{Hour},L_2),(\mathbf{GateType},L_3)))$. camping area, entrance, gate, general-gate, ranger-base, ranger-stop.

campsites (green spines): vehicles mostly stay within the campsite from 23:00 to 5:00, including cars and motorbikes and trucks; no buses visit the campsites.

Looking into more detailed patterns requires analysing the characteristics of the data before deciding the exploration strategy. First, there is a limited number of park rangers, while there is no theoretical limit on the number for other types of vehicles. Additionally, for non-rangers, it is impossible for any vehicle to pass through any gate without having passed by an entrance at any point in time; as such, it is likely that the number of records for various gates closed to entrances are highly correlated. For displaying absolute numbers, this is also as vehicles passing through the park generally

Stacked bar chart			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Group**	–	–	–	-	-	**Group**	SCALE	SPINE
2	**Hour**	*N*	–	–	-	**Hour**	-	SCALE	SPINE
3	**GateType**	*N*	**GT**	–	-	-	**GT**	SCALE	SPINE

Table 8.2 Specification of stacked bar chart of hourly gate usage for fig. 8.7. Signature: N(**Group**,L_1),N((**Hour,**L_2),(**GateType**,L_3)))

have a fixed sequence of gates that they can or must pass through. As such, the numbers are good for a *rough* estimation that is being done in this analysis.

This can be seen in fig. 8.7, where the bars for each hour are divided by the number of records for *GateType*, effectively resulting in a *stacked bar chart*. *General-gate* records usually outnumber the others, which is justified by the fact that are more general-gates than the other types of gate. However, even the average number of records per gate would be misleading given the sequence of gates. From this point on, the exploration must consider looking at the *individual* elements of the categories, rather than considering them as a whole. The main challenge thus is to manage the large number of individual vehicles.

Heatmap			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**GateType**	–	–	–	-	-	**GateType**	SCALE	SPINE
2	**GateName**	*N*	–	–	-	-	**GateName**	SCALE	SPINE
3	**Hour**	*N*	**Count**	–	-	**Hour**	-	SCALE	SPINE

Table 8.3 Specification of *heatmap* (fig. 8.8) displaying the number of records per GateName per hour. Individual gates are aggregated in gate categories. Signature: N(**GateType**,L_1),N((**GateName**,L_2),(**Hour**,L_3)))

Figure 8.8, with specification in table 8.3, displays the hourly activity per individual gate. The visualisation was initially partitioned by the GateType to emphasise the different gate groups, as indicated by the *gaps* between the groups. The colour of each spine is once again based on the total number of records. This configuration enables quickly finding the peaks of activity for each gate and the periods of inactivity.

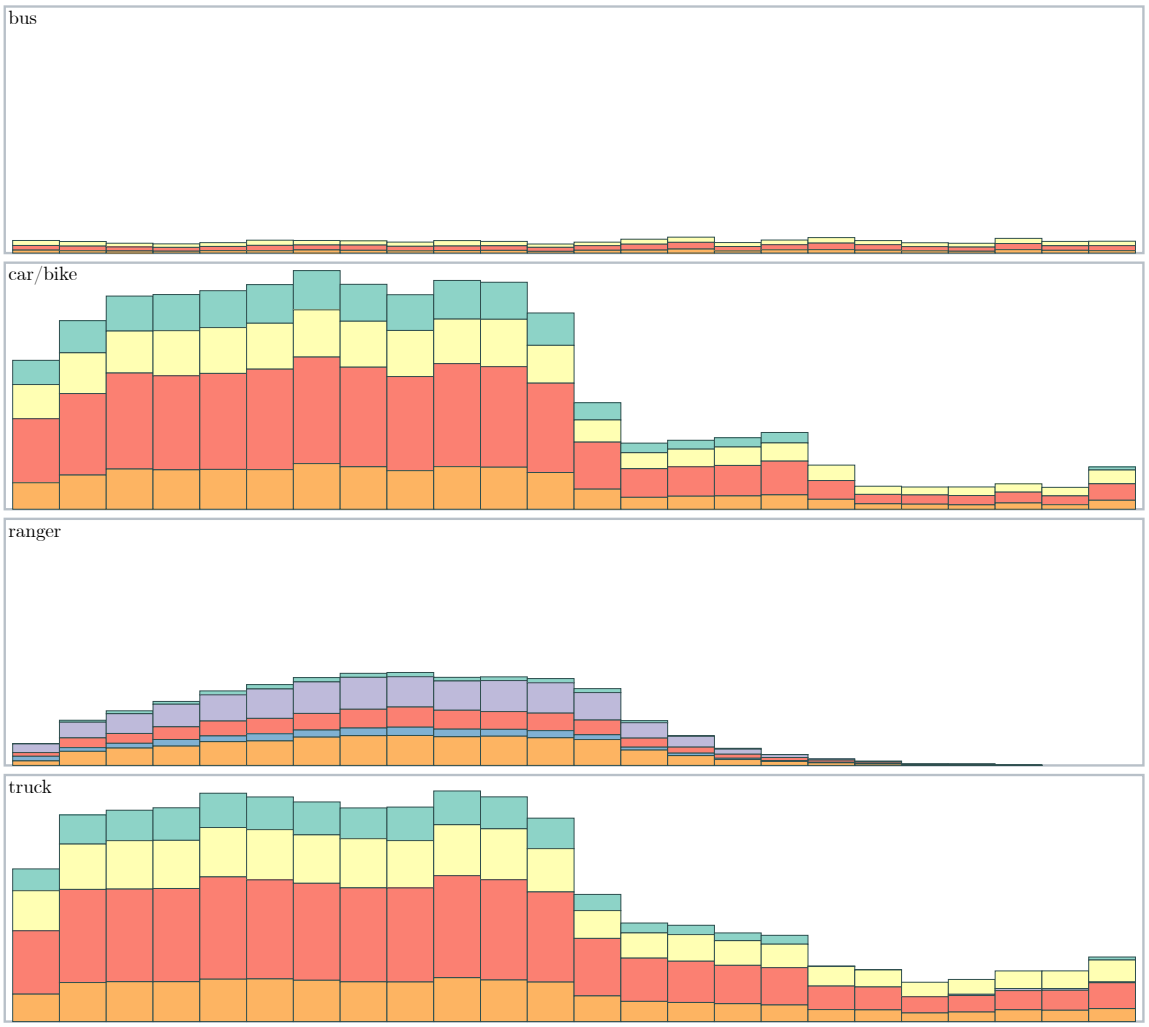


Fig. 8.7 Stacked bar chart of hourly gate usage. Signature: $N(\mathbf{Group},L_1),N((\mathbf{Hour},L_2),(\mathbf{GateType},L_3)))$. camping area, entrance, gate, general-gate, ranger-base, ranger-stop.

Considering the number of gates and hours, the space-filling arrangement makes good use of the available space conveying the required data.

Comparing to previous visualisations, the intense activity in the entrances – a necessary gate to enter the reserve – and the general gates is still clear. Detailed information can be extracted from individual gates – such as *entrance3* being clearly a busier gate than the others. Some of the information in the visualisations may be evident with knowledge about the placement of gates, such as the fact that certain ranger stops are much busier than other stops, probably due to them being on the path between two entrances.

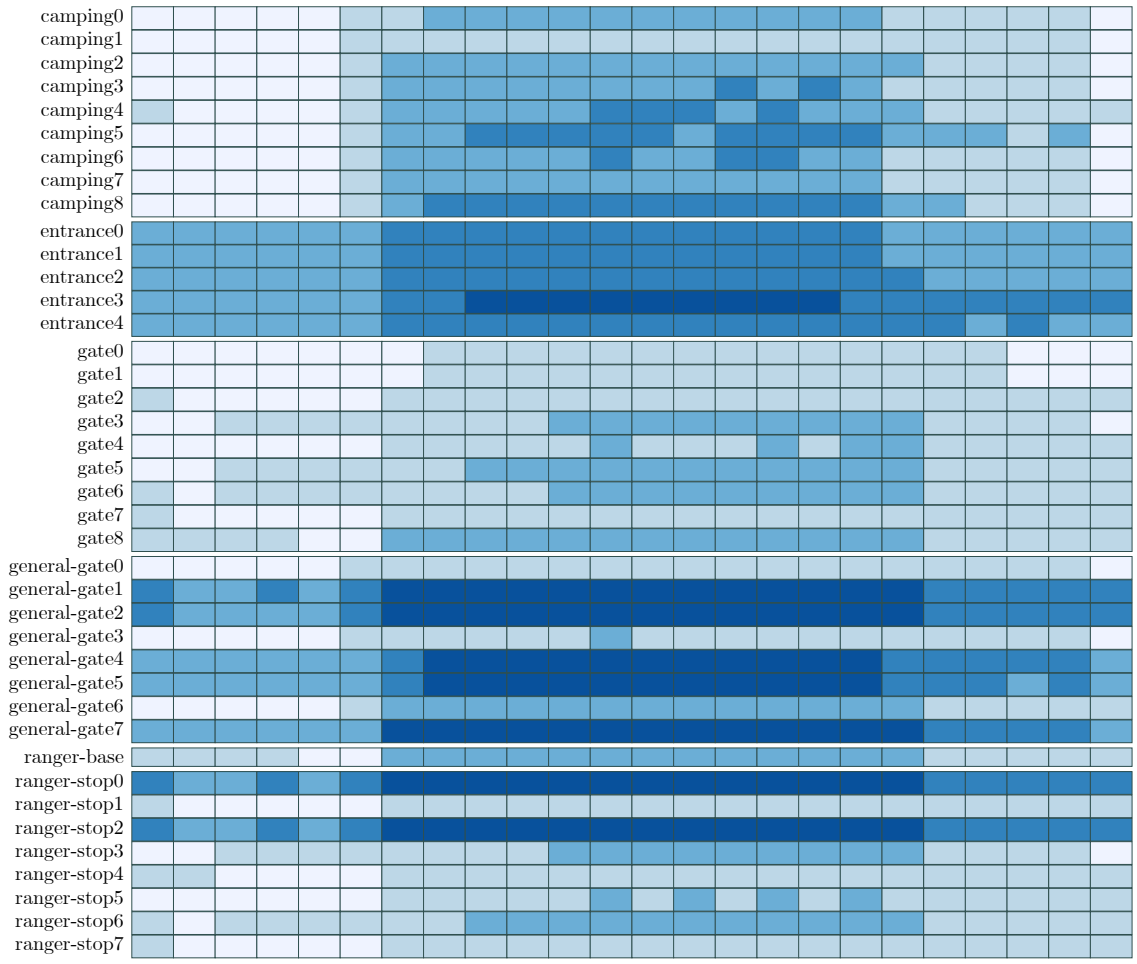


Fig. 8.8 Heatmap for GateType, GateName and Hour for specification in table 8.3. Signature: $N(\mathbf{GateType},L_1),N((\mathbf{GateName},L_2),(\mathbf{Hour},L_3)))$

In addition to hourly activity, multi-scale patterns can be found by combining data aggregated by *Month*, *Days of Month* and *Days of the Week*. Figure 8.9 (specified in table 8.4) displays the distribution of number of records for gate category for each day of the week, for each month. Each line inside a partition corresponds to a month, and each point of the same colour corresponds to the same day of the week, starting on Sunday. The visualisation shows that, in addition to general increased activity during the summer months, there is also an increase in weekend visitors. Alternatively, it is possible to *swap* the variables *Month* and *DayOfWeek*; in this case, the variables mapped to the encodings must also change to preserve the arrangement (fig. 8.10). In the second view, for each day, the monthly variation is inside the same spine. Seasonal summer increase is noticeable for camping, entrance, general-gate and ranger-stops.

The second visualisations may also lead to wrong interpretations due to the inverted configuration of the hierarchy; it is easy to assume that, in each row, the lines and dots in each line form a full sequence. This can be potentially mitigated by modifying further aspects of rendering, such as adding more labels and gaps between the spines. As mentioned in earlier chapters, these adjustments are not in the scope of the framework and should be part of a complete visualisation design process.

			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**GT**	–	–	–	-	-	**GT**	SCALE	SPINE
2	**Month**	*N*	–	–	-	**GN**	-	SCALE	SPINE
3	**DoW**	*N*	**DoW**	**DoW**	**DoW**	**Count**	-	SCALE	SPINE

Table 8.4 Specification of a multi-line chart (fig. 8.8) displaying the monthly variation of gate usage per day of the week. Individual gates are aggregated in gate categories. GT = GateType, GN = GateName. Signature: $N(\mathbf{GateType},L_1),N((\mathbf{Month},L_2),(\mathbf{DoW},L_3)))$

At this point, the limits of an overview of the full dataset are reached. Visualising more detailed daily patterns can be done in two ways: filtering and navigating the hierarchy, or using analytical methods such as *clustering*. As the framework focuses on visual exploration as part of the process of generating hypotheses that can be then investigated with analytical methods, this aspect will not be discussed. In terms of filtering and navigation, as there are well known techniques and guidelines for this, the transformation component only addresses the gap that was identified in the beginning of this thesis. On this subject, Elmqvist and Fekete (2010) proposed interaction techniques to filter and navigate hierarchical visualisations and guidelines to consider in hierarchical visualisation design. These include interactions in the visual space and the hierarchy; hierarchical interactions are appropriate for the framework.

The authors suggest rendering strategies for displaying parts of the hierarchy, such as a *range* of levels, as well as transformations that define the *nodes* or *sets of nodes* being visualised. Both are appropriate to help solving the tasks in the case study. In particular, visualising unclustered records of individual vehicles requires a considerable amount of filtering for any visualisation to be helpful. For daily patterns, this can mean filtering specific *months* and also arbitrarily grouping vehicles to display them. The last operation is consistent with one of the suggested guidelines, which is to keep a

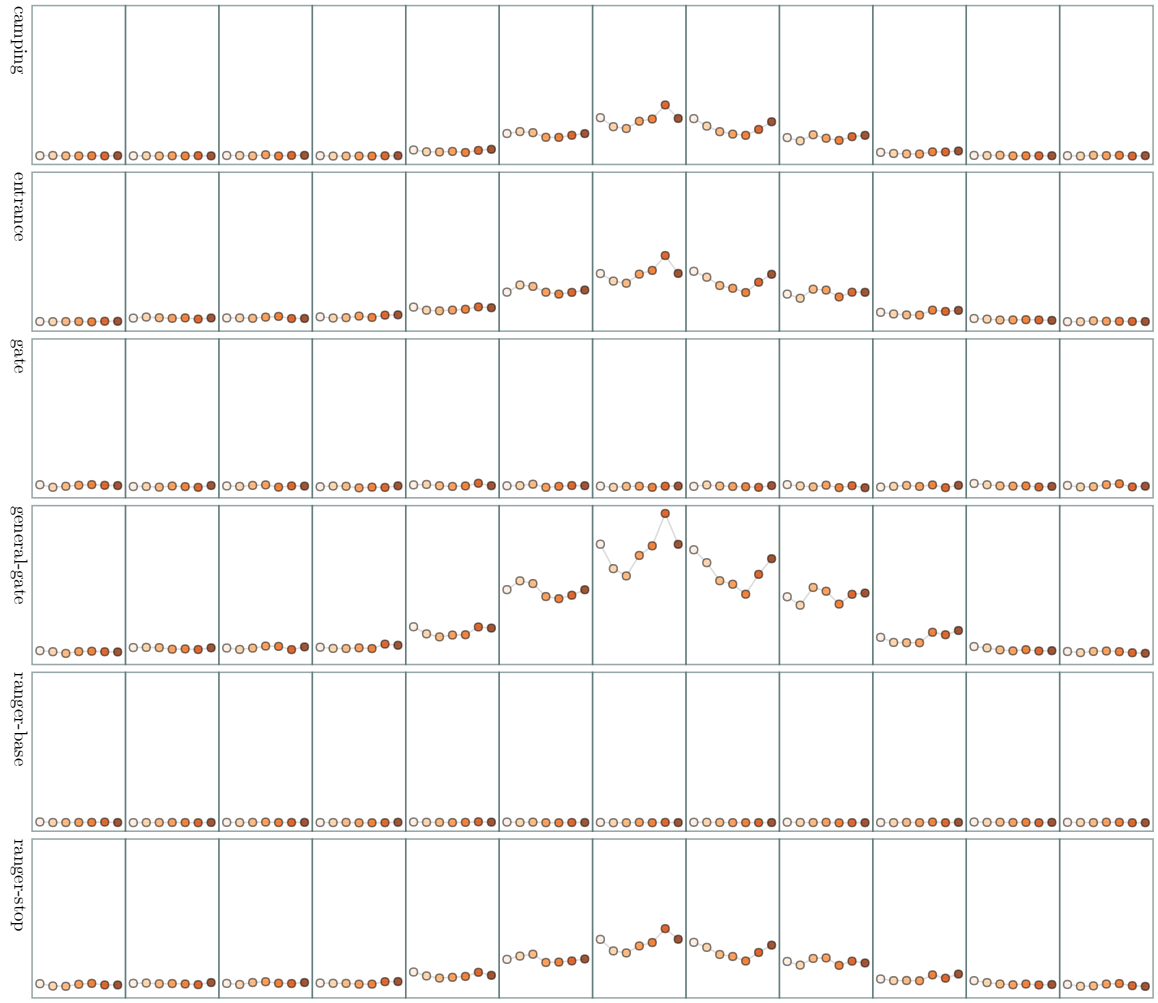


Fig. 8.9 View of $N(\mathbf{GateType},L_1),N((\mathbf{Month},L_2),(\mathbf{DayOfWeek},L_3)))$

budget of visual entities that will be rendered at any time. This enables showing only views that the users can effectively extract information from. This is also compatible with clustering methods where the size of each cluster can be defined as a parameter, whereas hierarchical clustering techniques can be use to match the results of the method with the hierarchical structure of the framework (Tan et al., 2005).

In the context of this chapter, the primary focus of filtering would be on individual vehicles, with the focus on visual exploration, no clustering is applied at this time. In addition to deciding the appropriate techniques for filtering, it is also necessary to choose the dimensions and values of the data that will be filtered. For daily patterns, choosing one category of vehicle at a time is a sensible approach. However, for cars and motorbikes (category 1) and trucks (categories 2, 3 and 4), the number of vehicles

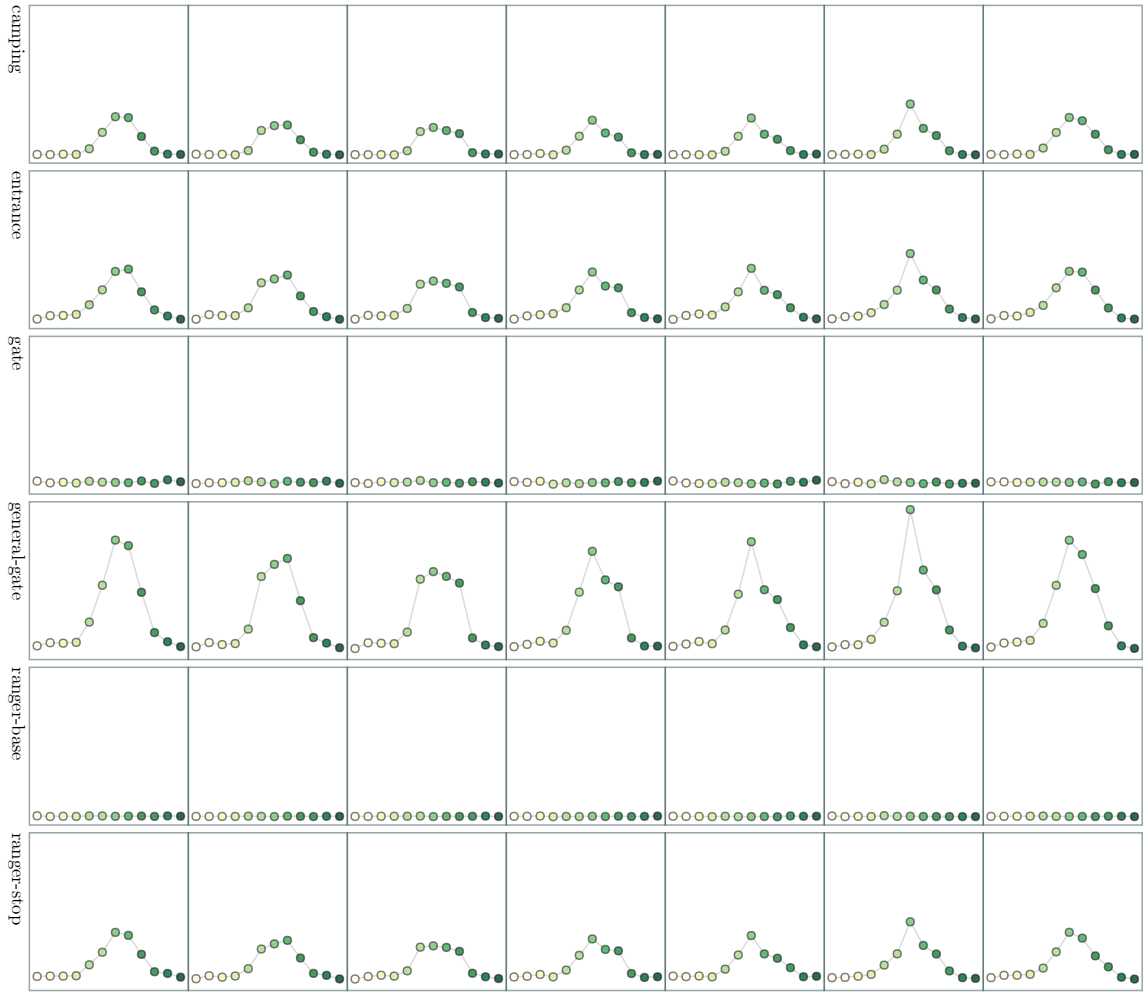


Fig. 8.10 View of N(**GateType**,L_1),N((**DayOfWeek,**L_2),(**Month**,L_3)))

is still a barrier for visualisation. The next dimension to be filtered can be *Month*; as it can be seen in fig. 8.11, activity in the reserve during summer months is particularly intense.

However, filtering by month is yet again not enough for certain categories. While filtering category *6* (three-axle buses) for month *5* results in 63 unique vehicles, keeping the same month filter for category 4 (four-axle trucks) increases the number of records to 253. Because the objective is to analyse in detail days and hours of each day, the resulting visualisation is not be effective (see fig. 8.12). As the framework has limited support for filtering and analytical methods, the chapter will continue by exploring the aspects of the framework that support the visual exploration.

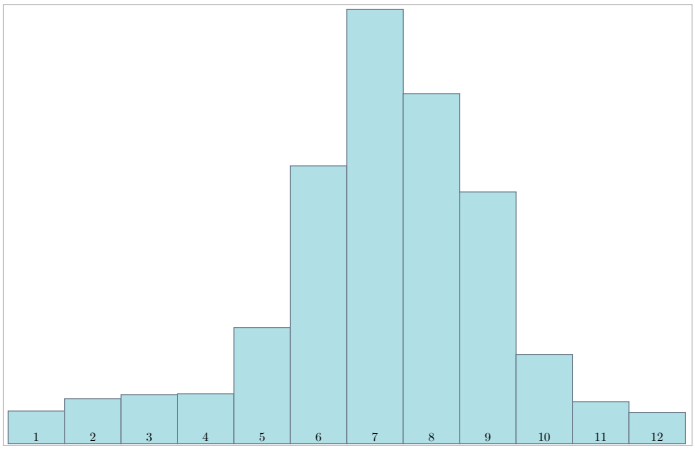


Fig. 8.11 Total number of records per month

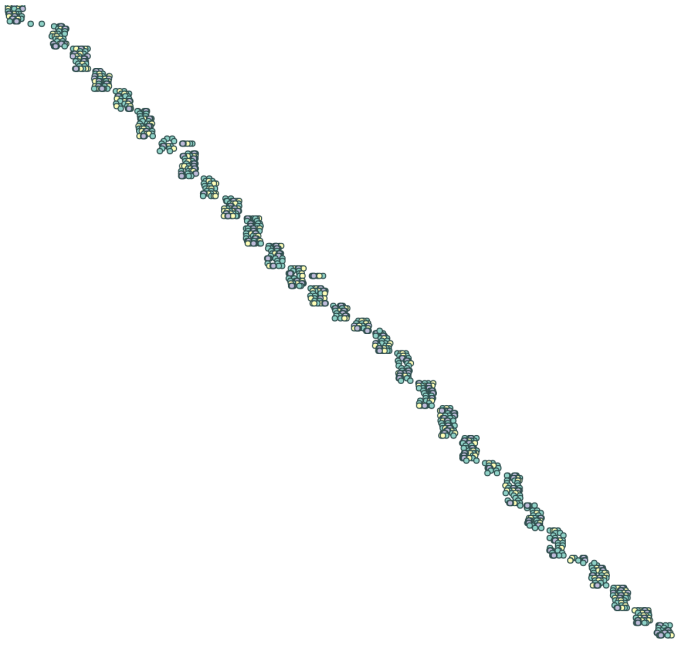


Fig. 8.12 Ineffective view with $N((\textbf{CarId},L_1),N((\textbf{Day},L_2),(\textbf{Hour},L_3)))$, after filtering by month and vehicle type.

Considering daily patterns and the type of data that is used — timestamped records, another approach is to extract *intervals* from the records, based on the earliest and latest records of the day. An initial relevant visualisation to have an overview of the intervals is the *triangular model* Qiang et al. (2012). In this visualisation, absolute time is mapped to the horizontal axis, while duration is mapped to the vertical axis. The midpoint of each interval is projected on the horizontal position; longer intervals are closer to the middle of the view horizontally. As the duration is mapped to the other

Triangular model			CARTESIAN, INTERVAL						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**CarId**	–	–	–	-	**CarId**	-	-	SPINE
2	**intervalId**	*N*	**Group**	–	-	**HM**:mp	**HM**:d	SCALE	POINT

Table 8.5 Specification of the triangular model in fig. 8.13. Signature: $N((\mathbf{CarId},L_1),(\mathbf{intervalId},L_2))$; **HM** = HourMinute; mp = midpoint and d = duration.

position, longer intervals are closer to the top of the view as well. As each interval is mapped to a single point, it is possible to compare temporal aspects of multiple temporal intervals. As an example of *spatially constrained* layout in relation to time, the visualisation has a limitation on the available channels for other attributes; colour could be used for vehicle category, for example.

The configuration of a visualisation depends on which variable is used to identify an interval. In the example of this case study, the only change to the original records was the addition of an *intervalId* variable that identifies, for each vehicle, the unique intervals that were found. This means that in order to distinguish vehicles and intervals, the hierarchical structure must begin with the CarId. As the triangular model has every point in the same view, *within level superimposition* is used in the first level; the second level contains the definition of the triangular model. The specification of this is in table 8.5, with the visualisation itself in fig. 8.13.

The visualisation does not display details about the gates that vehicles pass through, but it enables comparing the distribution of the categories over time and the duration of their daily activities. In the visualisation, the points with red hue are ranger vehicles. It is easy to see that, for each vehicle, their activity is longer during the whole day compared to other categories. Additionally, rangers usually begin their daily activities at approximately 6:00 AM; this is contrast with the other categories of vehicles that have a consistent activity throughout the night.

An interesting pattern that the visualisation shows is the starting time of longer activities in the reserve: most non-rangers vehicles that stay longer than two to three hours begin their activity in the morning. This is noticeable from the fact that, with a few exceptions, there are no midpoints of intervals in the 200-600 minutes range after 12:00 PM (highlighted region in red in fig. 8.14). Beyond this, it is difficult to extract more information as there is a large number of cluttered points in the 0-100 minutes

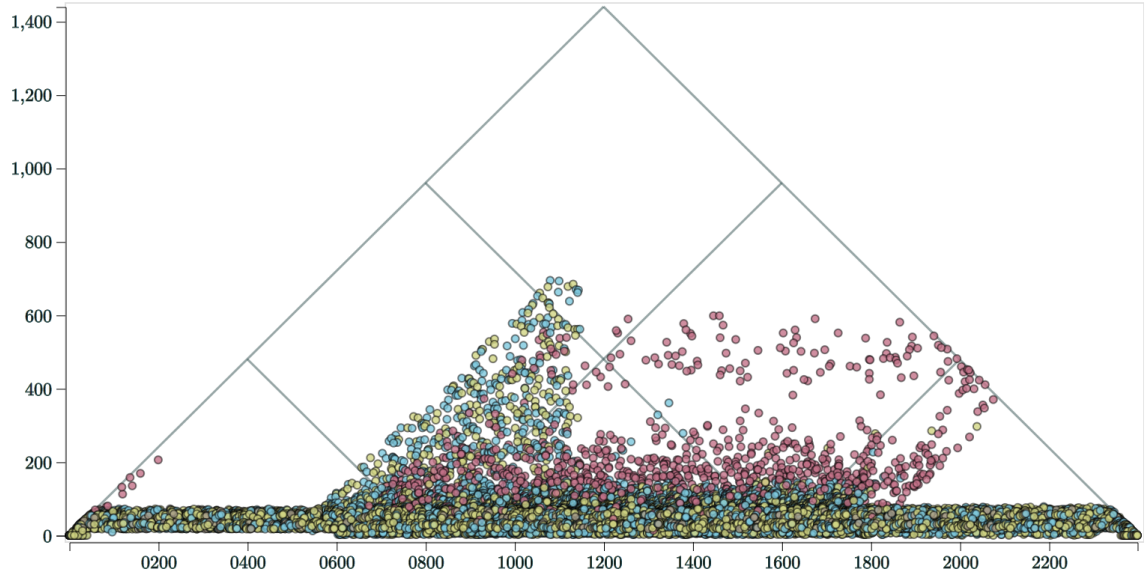


Fig. 8.13 Triangular model view of N((**CarId**,L_1),(**intervalId**,L_2)). buses, cars/motorbikes, rangers, trucks

Triangular model			CARTESIAN, INTERVAL						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Group**	–	–	–	-	-	**Group**	SCALE	SPINE
2	**CarId**	*N*	**Group**	–	-	**HM**:mp	**HM**:d	SCALE	POINT

Table 8.6 Triangular model with new level added to partition the space by **Group** for fig. 8.15. Signature: N((**Group**,L_1),(**CarId**,L_2)); **HM** = HourMinute; mp = midpoint and d = duration.

range. Besides colouring, vehicle types can be used in visual composition, conditioning the intervals to show the distribution for each group of vehicle.

Table 8.6 describes the specification of the triangular model view after a new level has been added at the top of the tree, with the resulting visualisation shown in fig. 8.15. The groups ordered from top to bottom are buses, cars/motorbikes rangers and trucks. The new view emphasises the difference between buses and the other categories, which due to the cluttered view was difficult to see in the single triangular view; the similarity between cars/motorbikes and trucks is also confirmed in this view.

It is possible to identify longer scale patterns by adding a new level for a time domain configured to *Month* between the two existing levels, as specified in table 8.7 and

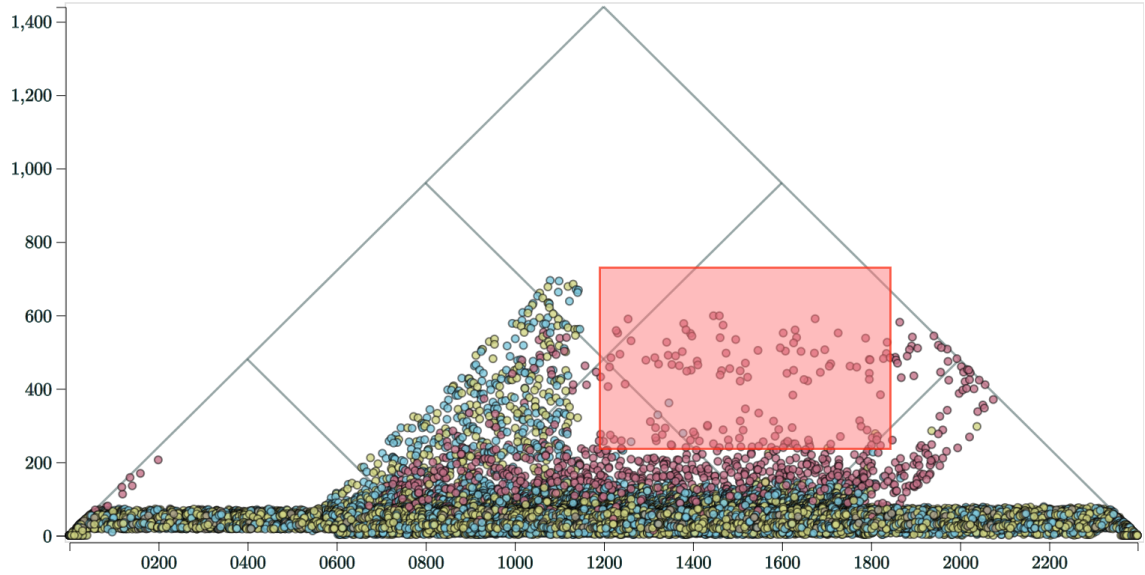


Fig. 8.14 Highlighted region in the triangular model view. Vehicles inside the region are mostly rangers.

displayed in fig. 8.16. In the new view, it is possible to see the increase in longer daily activities during the summer months (the views near the centre) for cars and motorbikes and trucks. Buses, on the other hand, don't see any change in the duration of the intervals. Ranger activity is regular throughout the year.

There are other ways to visualise intervals, but they are again limited by the large number of data points. Visualisations that map the duration to the length of a bar, for example, would require applying multiple filters or analytical methods, such as clustering the intervals that contain the same sequence of gates, for example.

Triangular model			CARTESIAN, INTERVAL						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Group**	–	–	–	-	-	**Group**	SCALE	SPINE
2	**Month**	*N*	–	–	-	**Month**	-	SCALE	SPINE
3	**CarId**	*N*	**Group**	–	-	**HM**:mp	**HM**:d	SCALE	POINT

Table 8.7 Triangular model with new level added to partition the space by **Month** between the previous two levels for fig. 8.16. Signature: $N((\textbf{Group},L_1),(\textbf{CarId},L_2))$; **HM** = HourMinute; mp = midpoint and d = duration.

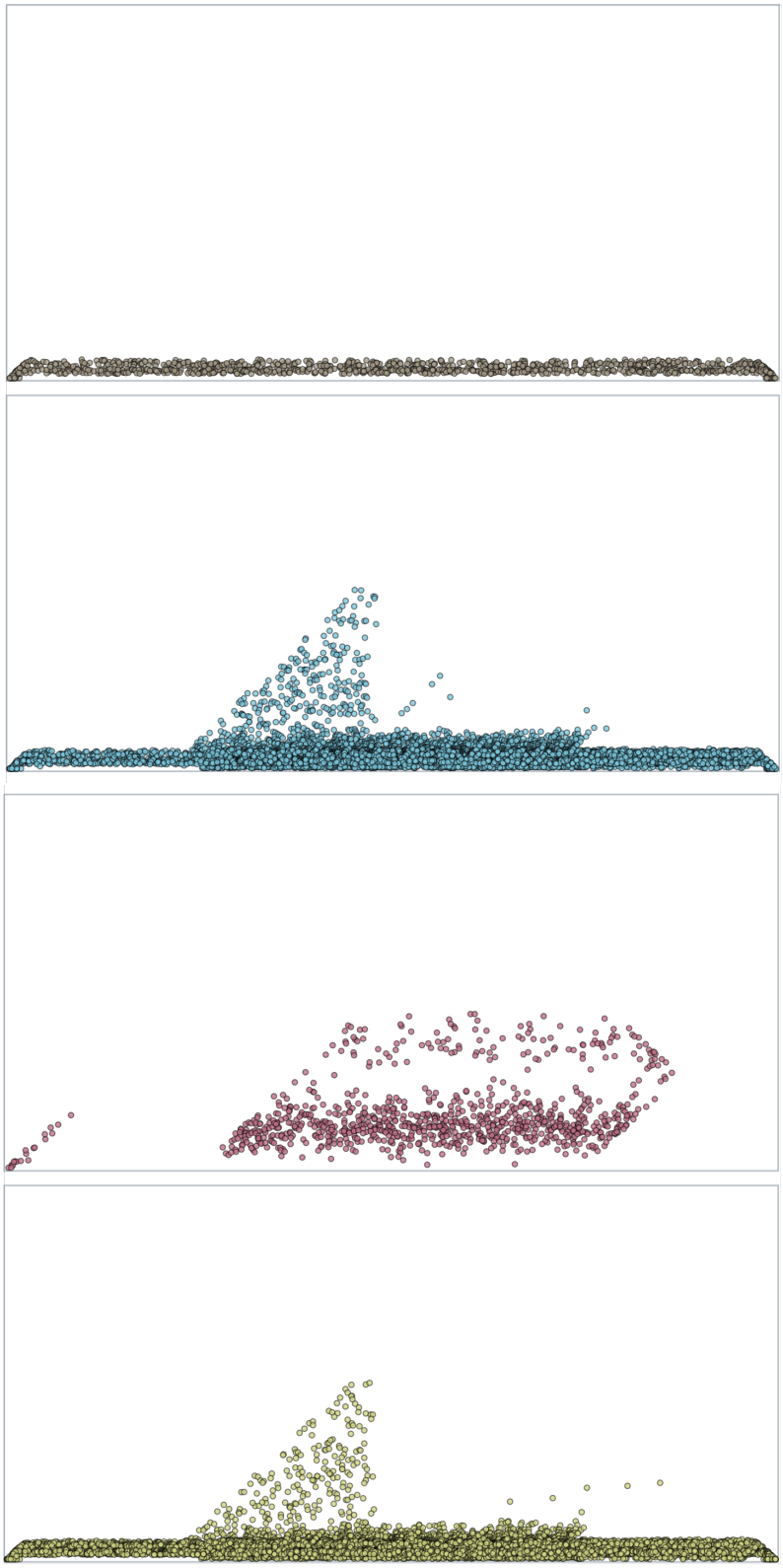


Fig. 8.15 Triangular model with added level of Group, $N((\textbf{Group},L_1),N((\textbf{CarId},L_1),(\textbf{intervalId},L_2)))$.

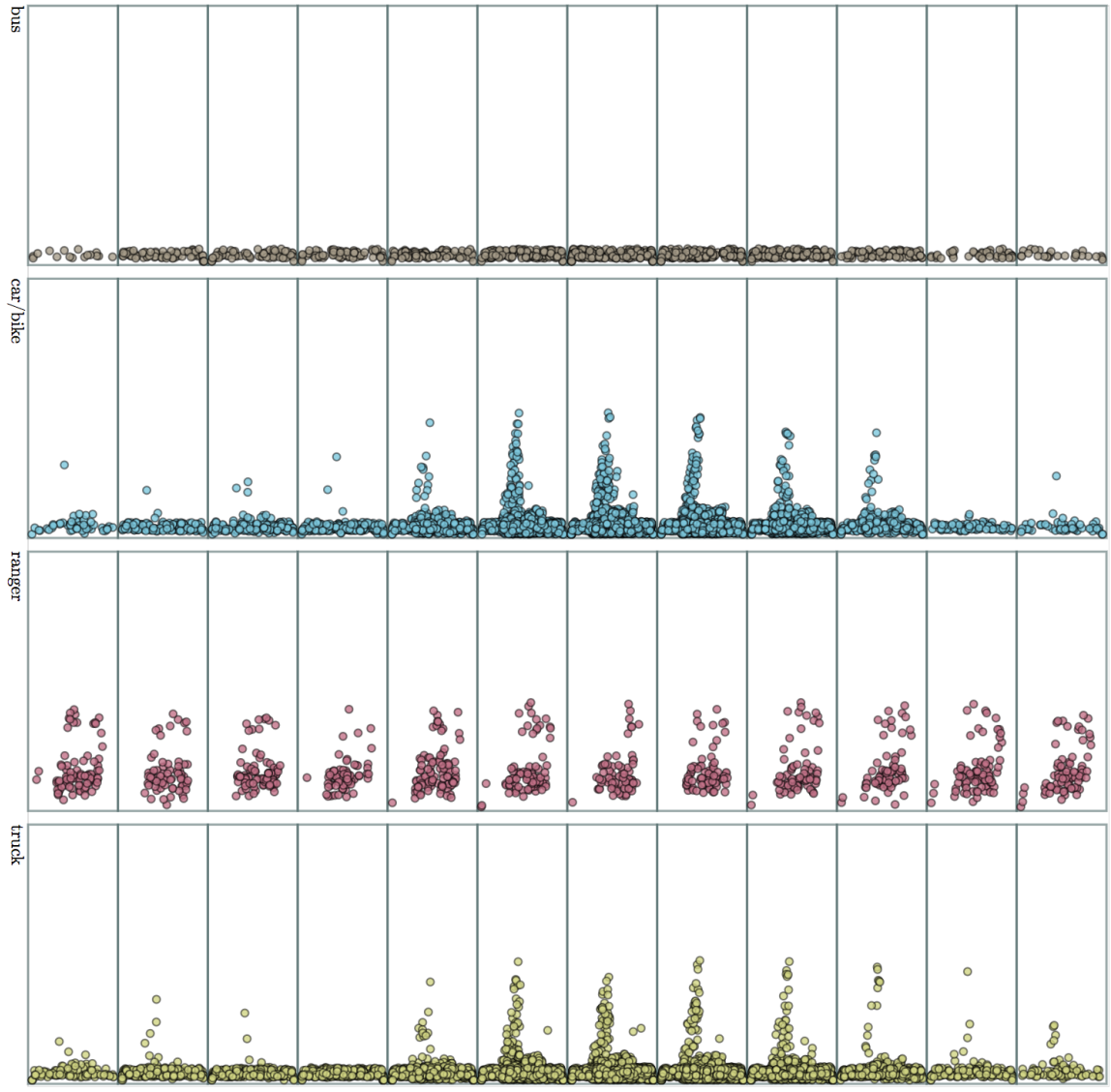


Fig. 8.16 Triangular model view with a new level for *Month* added between the two previous levels.. Signature: $N((\textbf{Group},L_1),N((\textbf{Month},L_2)\ (\textbf{CarId},L_3)))$.

8.3.1 Summarising the answers

The answers to the challenge were discussed as part of the visual exploration process that used the components of the framework. The limitations of the framework were also considered, particularly the scalability of some visualisations in the context of a large dataset. The two tasks were concerned with *daily* and *multi-scale* patterns of vehicles in the reserve, in addition to unusual patterns. The following conclusions were found regarding groups of vehicles:

Cars and motorbikes

- Cars and motorbikes were the most common types of vehicle passing through the reserve;
- Their daily activity was higher between approximately 6:00 to 18:00;
- Campsites were entered or exited from 5:00 to 22:00, with no suspicious activity registered;
- In summer months, there was an increase in both activity and the duration of their daily activities;

Trucks

- Trucks were the second most common type of vehicle passing through the reserve;
- Their daily activity was higher between approximately 6:00 to 18:00, but they also had more activity registered through a 24-hour period compared to cars and motorbikes;
- In summer months, there was an increase in both activity and the duration of their daily activities;
- Trucks were spotted in *ranger-stops*, which are gates with access restricted to rangers;

Buses

- Buses were the least most common type of vehicle passing through the reserve;

- Their daily activity was regular throughout the day;
- Their activity did not change throughout the months;

Rangers

- Rangers have a mostly regular working schedule which spans;
- Individual ranger vehicles have longer *active* daily hours compared to other vehicles;
- As the number of rangers in the park does not vary, their activity is also regular throughout the year;
- Rangers do not venture outside the park and only inspect campsites during regular hours;

Regarding the gates in the reserve, the following conclusions were found:

- There is an increase in the usage of campsites in the reserve in the summer months, while in the rest of the months they are comparably low, with no apparent seasonal change from spring/autumn to winter;
- Gate *entrance3* is the most commonly used entrance compared to the other entrance gates;
- Campsite 1 is the least used campsite; campsite 8 is the most popular campsite;

8.4 Visual exploration paths

One of the main features of the framework is the ability to describe the navigation strategies that are used to engage in the visual exploration process, matching transformations in each component to the steps in Card-Norman's model. It enables relating hypotheses, insights and answers to the different configurations that were used. In the example of the case study, this starts with the first specified *overview* of the dataset — fig. 8.17 shows the history of visual exploration for instant time exploration. Each node corresponds to a final visualisation specified in the case study, the edges represent the framework's operations that were used in the exploration.

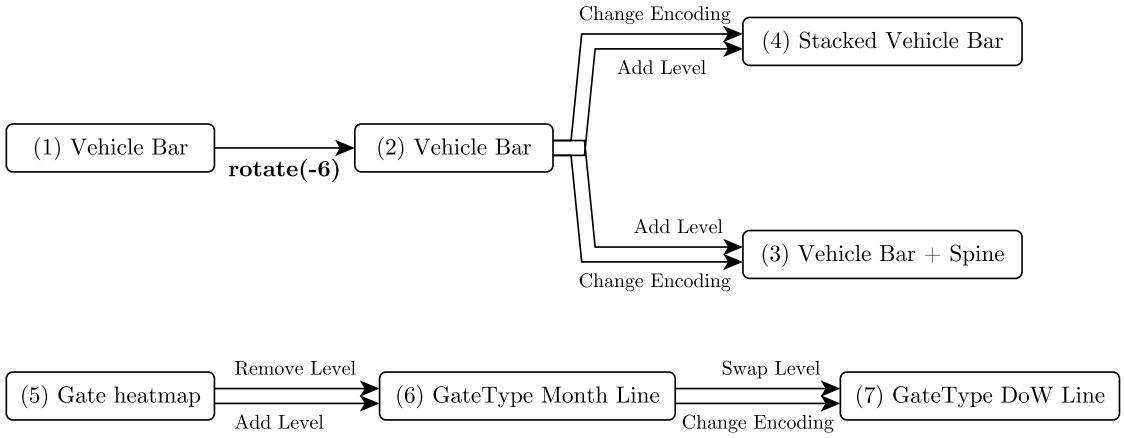


Fig. 8.17 Exploration strategy for instant time with no intermediate visualisation steps and composite transformations.

Alternatively, a diagram containing the individual steps that lead to compositions between layers can also be extracted. In fig. 8.18, each node indicates an intermediate or full visualisation and the numbered arrows indicate the order of transformations that lead to each visualisation. The corresponding figures in the thesis are pointed in the yellow markers. The diagram does not indicate individual layers that are *shared* across visualisations, but the layers that serve as a base for multiple views, such as the spine for *GateType*, are easy to spot. The full navigation, starting from an empty visualisation to the last task for instant time contains 19 operations. The chosen level of detail for the operations combines adding a new level to the hierarchy and a specification for this new level; the operations could be broken down even further if these operations were separated.

In the second diagram, it is possible to identify when the visual exploration had a significant change in direction. From the full visualisation for fig. 9.7, two levels were removed at once. From that point, variations of visualisations based on the GateType spine were explored. Comparing it to the rest of the diagram enables analysing the parts of the domain that were being explored: in the first part, vehicle types were at the centre of the exploration; in the second part, the types of gate became the focus.

The diagram also enables relating the generated visualisations, the visualisation strategies and the answers to the tasks. In the current case study, it is quite obvious which dimensions of the dataset relate to the different visualisations as the meaning of dimensions is clear. It is likely, though, that in other domains the connection will not

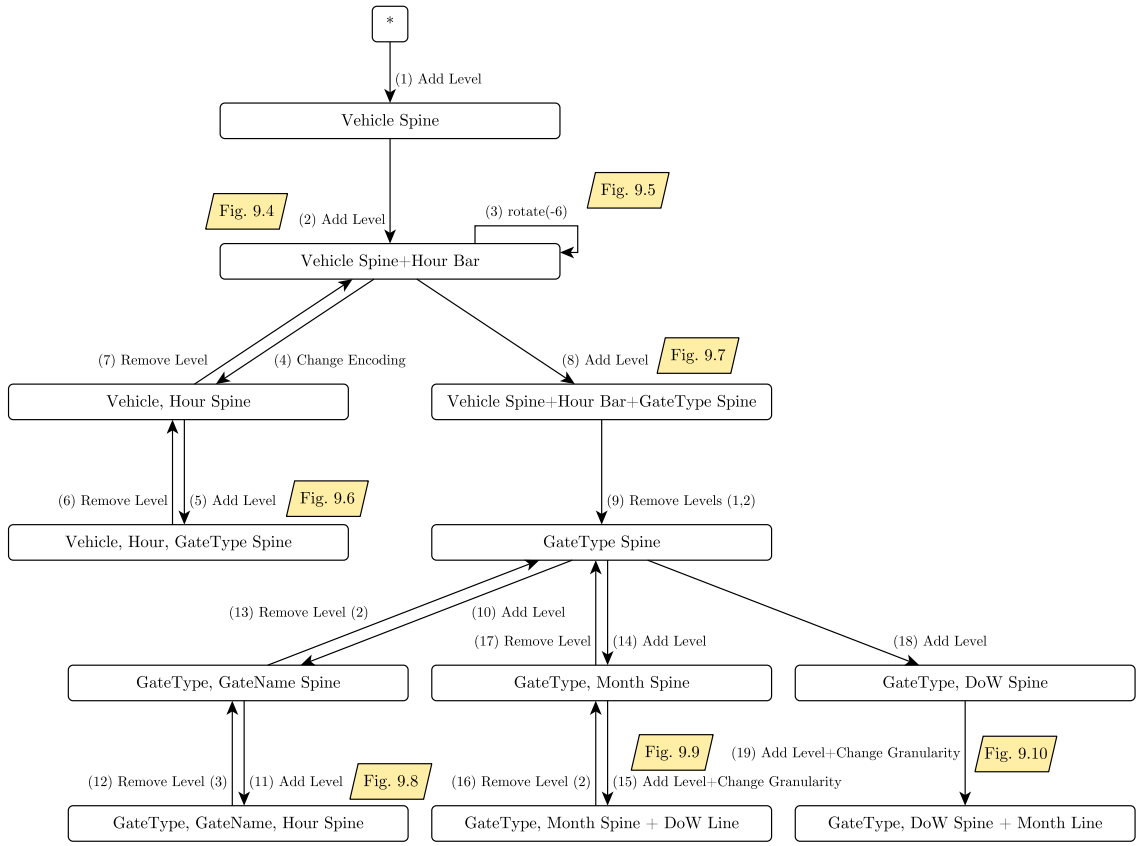


Fig. 8.18 Exploration strategy for instant time with intermediate visualisation steps and unique transformations.

be straightforward, when the tasks are more open-ended or the exact meaning of some dimensions is not clear.

Besides higher level analysis of the paths taken in the visual exploration, lower level similarity between views can also be analysed. For this, it would be beneficial to move from concrete variable mappings to abstract categories. Figure 8.19 is an example of such conversion: the diagram becomes more convoluted, but there are fewer nodes. In comparison with the previous diagram, it is also difficult to find out the *endpoints* of the analysis. The diagrams, however, do not describe detailed use of visual channels. Considering the number of attributes for each level, other methods are required to display and compare the changes for the full specifications. One way is to show the tables that were used throughout the chapter in order, highlighting what is changing in each table from the previous one.

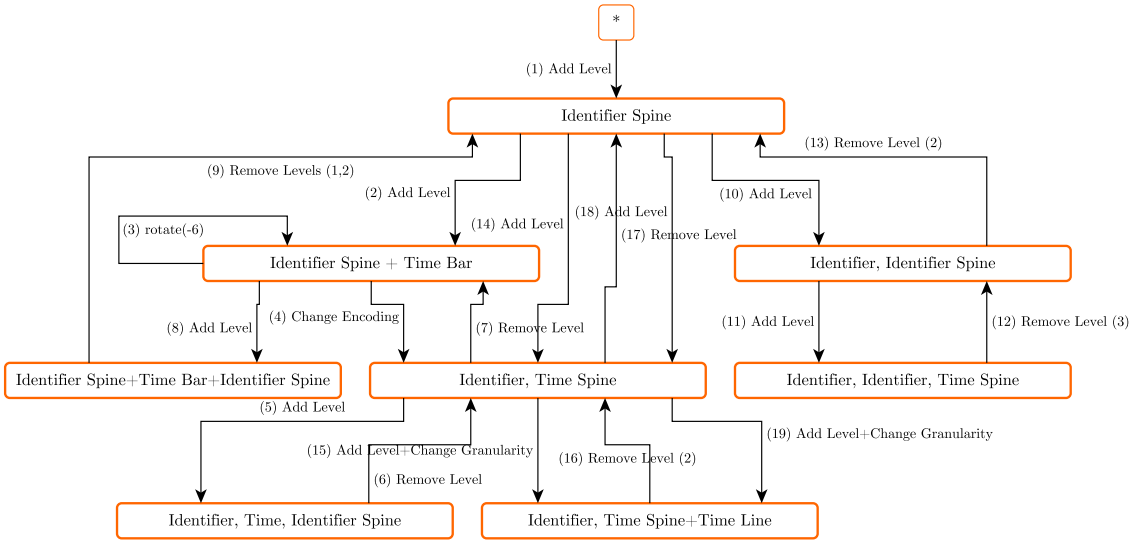


Fig. 8.19 Exploration strategy for instant time with intermediate visualisation steps and unique transformations.

The suggested composition grammar also enables a higher level analysis of the composition methods that were used, without including information about the visual mappings. The following series of visual compositions were used throughout the chapter:

- $N((\textbf{Group},L_1),(\textbf{Hour},L_2))$
- $N(\textbf{Group},L_1),N((\textbf{Hour},L_2),(\textbf{GateType},L_3)))$
- $N(\textbf{GateType},L_1),N((\textbf{GateName},L_2),(\textbf{Hour},L_3)))$
- $N(\textbf{GateType},L_1),N((\textbf{Month},L_2),(\textbf{DayOfWeek},L_3)))$
- $N(\textbf{GateType},L_1),N((\textbf{DayOfWeek},L_2),(\textbf{Month},L_3)))$

These different methods for describing the evolution of configurations demonstrate the framework's potential for provenance analysis, in which records of *actions* can be used to replicate or recover analysis and review the exploration strategies (Ragan et al., 2015). The components of the framework and their properties enable different levels of granularity (or detail) to be captured and analysed, such as the visual composition arrangements, visual mappings and general transformations that were used. The actions can also be identified back in Card-Norman's model, both in the visualisation pipeline and the interaction loop.

8.5 Chapter summary

This chapter explained the application of the framework in a case study based on a data visualisation challenge. The use of each component in the design of visualisations for exploratory analysis was described. Afterwards, the framework was used to address the tasks that were part of the challenge, describing the rationale for reconfiguring the visualisations. This was followed by analysis of the exploration strategies used in the challenge. The next chapter discusses the contributions presented in the thesis and concludes it by discussing the limitations and potential for future research.

Chapter 9

Conclusion

This chapter concludes the thesis by discussing the contributions and limitations of the research, leading to a critical reflection of the completed work and ending with suggestions for future research that are enabled by the work presented in this thesis.

9.1 Contributions

The primary contribution of the thesis is the **framework for hierarchical time-oriented data visualisations**, which is presented as an outcome to addressing the research question *how can interactive hierarchical visualisations help explore temporal data?*. Breaking down this question into three secondary questions allowed different aspects of visual exploration with hierarchical and time-oriented data visualisations to be investigated:

- **RQ1: What are the interactive visualisation methods used for temporal data?**
- **RQ2: How can hierarchical composition techniques be combined with the surveyed visualisations?**
- **RQ3: What are the temporal interactions that facilitate exploring temporal data in hierarchical visualisations?**

RQ1 and RQ2 were jointly addressed by the use of the *view* and *composition* components in appendix A, where it is demonstrated that the framework can successfully describe the techniques found in literature of time-oriented data visualisations that are within

the scope of this research. RQ3 was addressed in chapter 6, which described the transformations that enable the exploration of multiple perspectives based on the aspects of time, with appendix D documenting the process that led to the definition of the operators.

Chapter 7 summarised the combination between the components of the framework, leading to the answer to the primary research question: a framework that supports *systematic* exploration of visual and temporal aspects facilitate the use of hierarchical visualisations to help explore temporal data. Chapter 8 demonstrated how the framework can be used for this, with all aspects of the framework being explored: multiple perspective were defined by the use of the view and composition components, while the temporal transformations partly guided the exploration based on the different tasks that were investigated as part of the case study.

As outlined in the first chapter, both the *layered specifications of interactive visualisations for temporal data* appendix A and the temporal transformations introduced in chapter 6 are also presented as novel secondary contributions of this thesis in light of the literature examined in chapter 2.

In relation to the HiVE notation, described in section 2.1.3, the framework represents a significant conceptual advance. Although the initial idea of composing visualisations with false hierarchies was inspired by HiVE, the original work did not investigate how the various composition methods can be combined with different layouts and shapes. Additionally, this work explored temporal data visualisation and related concepts which were included in the framework, such as the temporal transformations. In contrast with HiVE, however, this thesis did not conceive a notation or extended the notation. This remains as a challenge for future work.

9.1.1 Addressed gaps

Two main gaps, described in chapters 1 and 2, motivated the research presented in this thesis and are addressed as follows:

Inclusion of temporal aspects in hierarchical visualisations Existing frameworks and models for hierarchical visualisations do not include provisions to support aspects of temporal data as part of the visualisation process. The consequence is that the application of hierarchical concepts for time-oriented visualisations is limited from a theoretical point-of-view – as it can be seen in the survey in appendix A, only a small

number of visualisations employ a hierarchical view on time. The combination of the transformation and composition components enables future interactive visualisations to designed focussing on the use of hierarchical views for the exploration of multiple perspectives on temporal data.

Inclusion of temporal aspects in general visualisation and interaction design The transformation component maps out the possibilities for the *conceptual* characteristics of temporal data, which had not been previously considered in frameworks for time-oriented visualisations. Existing frameworks take a broad view of how the properties of time can be used to inform visualisation design, suggesting *which* aspects should be considered but not enumerating *how* to incorporate these aspects in the design process. The enumeration of operators as functions with parameters allows visualisation designers to fully consider how these transformations can implemented in applications, such as which modes of interaction can be used to trigger each transformation.

9.2 Benefits, limitations and future work

Future work based on this thesis opens up from both benefits and the limitations that emerged during the research.

9.2.1 Benefits

Comparing effectiveness of configurations The framework enables a systematic evaluation of effectiveness of configurations as part of the visual exploration paradigm. Although there is a number of common encodings that have been thoroughly investigated in terms of their effectiveness and perception problems (Adnan et al., 2016; Albers et al., 2014; Harrison et al., 2014; Javed et al., 2010), there is still great potential in this area. The design space formed by the framework enables planning experiments to identify effective configurations based on its three aspects: encodings, composition and the temporal transformations. An advantage compared to the literature described in chapter 2, the *generative* aspect of the framework enables layers that are common among various techniques to be compared in terms of their effects on the effectiveness of encodings.

Evaluating visual exploration strategies Roth (2013) argued that a taxonomy of interactions can inform experiments to "diagnose suboptimal operator strategies" given a particular task. In the context of the framework proposed in this thesis, this idea can be extended to all aspects of the framework, rather than only the operators in the transformation component, as discussed in the exploration paths in chapter 8. For encodings, the low level specifications can be compared across domains and tasks, whereas for temporal transformations, independently or combined with composition methods, the categories and different parameters can be analysed. This is also applicable in *provenance* analysis, in which records of *actions* can be used to replicate or recover analysis and review the exploration strategies (Ragan et al., 2015). Likewise, the granularity of the interaction process in Card-Norman's model is appropriate for provenance analysis, enabling considering the changes in encodings *and* the use of interaction in various steps. Shrinivasan and van Wijk (2008) also proposed a framework to support an *analytic reasoning process* that is compatible with the structure of the framework presented in this thesis, separating data views, provenance analysis views, where the exploration path is displayed, and knowledge-support views, where users can externalise their thoughts.

Supporting visual analysis Aigner et al. (2007b) identified the lack of a "visualisation framework that can handle all types of times and data, or provides a broader selection of possible representations" and called for "an open framework fed with pluggable visual and analytical components for analysing time-oriented data". This was stated in the context of visual methods that were suitable for particular tasks and data; such a framework would be able to cover more representations to support more tasks. As seen in chapter 2, this has yet to be achieved; however, the framework presented in this thesis is an important step in that direction that enables the configuration of visualisation methods to support multiple tasks as part of visual exploration. Although this thesis has not covered *analytical components* as part of this, future research can include analytical methods for temporal data, such as the derivation of *autocorrelation* Box et al. (1994) to enable forming hypotheses about underlying *cyclic* aspects of time with a strong statistical support.

9.2.2 Limitations

The scope of the framework, described in chapter 1, defined the limits of the research presented in this thesis. Nevertheless, there are also other limitations that arose due

to the choices taken in the design of the framework and that can serve as the basis for future work.

Types of visualisation

Two limitations are related to the types of visualisations: 2D and static, without complex *glyphs*. In order to include 3D visualisations in the framework, research is needed to find out how to include hierarchical compositions for temporal data in 3D views; Hadlak et al. (2010) experimented with juxtaposed layers in one of the three dimensions, but the work raised even more questions, including *how can data attributes and structural aspects be combined more efficiently?*, which the framework can help to answer. Animation was also excluded from the research; there are various ways to include it in the framework, such as the use of a special layout or as one type of composition. For future work, an interesting question is how the transformation component can be used to interactively modify animation parameters.

Complex glyph shapes are not supported by the framework. Borgo et al. (2013) defines a glyph as a multi-channel *visual object*; in certain cases, this definition applies to *nested* views; Munzner (2014) argues indeed that a glyph can be considered a *view* depending on the *size* of the glyph. In the context of the framework, the *unsupported* glyphs are those that employ unusual layouts and spatial arrangements and are used as normal non-nested visual marks – *growth ring maps* by Bak et al. (2009) are are an example of that. In the context of visual composition, the main characteristic of such shape is that their uniqueness makes it very difficult to incorporate them in a *multiple perspectives* exploratory approach. It is another case that could be addressed in future work; in contrast with other scope limitations, though, it should not be a priority.

Another limitation is the exclusion of *explicit* hierarchical visualisations. Although a *connecting path* element was introduced only for connecting points *within* level, the framework, in theory, should be able to support explicit encodings *between* levels as well. The main issue is investigating how linking items between the different levels work for all types of composition and keeping the framework consistent. Another aspect connected to the *connecting path* is the visual properties of the path itself, such as *thickness* and *colour*, which are unexplored in this thesis. Although there are no issues that prevent explicit encodings being supported by the framework, they were not included because the focus was on the different channels that can convey the order of time, rather than including every possible visual property of a visualisation. As an example of the possibilities for future work, Buchin et al. (2014) introduced connecting paths that encode the *duration* of an interval as a *distortion* of a straight line.

Composition methods

Regarding composition methods, the *overloading* method is not included in the framework. It is defined as *superimposition* or *nesting* of a *client* specification on top or within a *host* encoding; in this case, the client specification does not contain any positional variables, which are derived from the host encoding. The aim of the framework is to support basic compositions that are common in the literature; as there were no entries in the survey that cannot be supported due to the omission of overloading, it was not included in the framework. However, there are no conceptual incompatibilities that prevent it from being included in future works.

An important limitation is the absence of data-driven filtering as part of composition methods. The description of a visualisation always consider the hierarchical structure being fully displayed at the abstraction stage, with the exception of extent operators. The use of extent operators, however, is not defined for use with composition methods or visualisation patterns such as *overview plus detail* (Cockburn et al., 2008), which is the "simultaneous display of both an overview and detailed view of an information space, each in a distinct presentation space". With the operators, the overview is the original time domain and the trimmed domain is displayed in the detailed view. It is possible to extend the framework in the future by re-thinking the relationship between the hierarchical structure and the composition methods and allowing the definition of *sub-hierarchies* that can be visualised along with the main hierarchy.

Spatiotemporal visualisations

The last limitation defined by the scope is about spatiotemporal visualisations, which are primarily investigated in geographical information science and cartography. Although the framework does not exclude spatial data, there is a huge number of design choices for spatiotemporal visualisations and maps especially regarding the different shapes and symbols. Nevertheless, the framework already includes an important starting point to support this type of visualisation: *superimposition* and *nesting*. As many spatial data include hierarchical aspects, such as *countries and cities*, and the framework can fully support non-spatial views of this type of data, future work should include finding out the necessary shapes and layouts that are required for maps. The hierarchical aspects of spatial data is also related to the spatial transformations that can also be designed and implemented in a similar way to the transformation component of the framework. As some temporal data is collected at a arbitrary resolutions, geographical data is also referenced using arbitrary boundaries which might hide or reveal processes

at different scales and introduce biases in the data (Openshaw, 1983). Operators for spatial data may help with systematically exploring these.

9.2.3 Research agenda

As discussed in the previous two sections, this thesis opens up several research paths. Some of them are a direct result of the benefits of the thesis for visual exploration of temporal data, such as evaluating the effectiveness of various configurations of encodings, compositions and interactions. Further research can also expand the work presented here due to the current limitations, which exist as part of the scope and as part of design choices. The new research paths can be defined as part of a research agenda following up from this thesis and are summarised in fig. 9.1.

The first step to continue the research and realise the benefits of it is an implementation of the framework, which will enable in practice several of the outlined benefits. The implementation can take two forms: an open-ended visualisation environment for visual exploration or a prototyping/sketching application that uses the framework to design new systems. One challenge associated with the development of a visualisation environment is implementing encodings using the best practices in the literature, in terms of colours, spacing, size, among other properties. An alternative to this is to include concepts of the framework into existing libraries and languages. As discussed in chapter 2, the Vega-Lite grammar includes some degree of composition; an interesting future path would be to replace the existing composition techniques with the ones proposed in this thesis. The prototyping application would make it possible to explore new visualisation designs based on the ideas of framework. Chapter 8 demonstrated how the combination of existing techniques with the concepts of the framework allow this exploration. However, the designs were targeted towards the analytical tasks of the data challenge; a systematic exploration of designs should go hand-in-hand with empirical evaluations.

Extending the framework in light of its limitations could be done in parallel or as a second step. Research into the support for complex glyphs, animation and 3D visualisations does not depend on the implementation. Glyphs would greatly expand the scope of the framework; animation would allow for new types of visualisations as well as different types of analyses, such as with real time data; supporting 3D visualisations would also expand the scope – in comparison with glyphs, it would

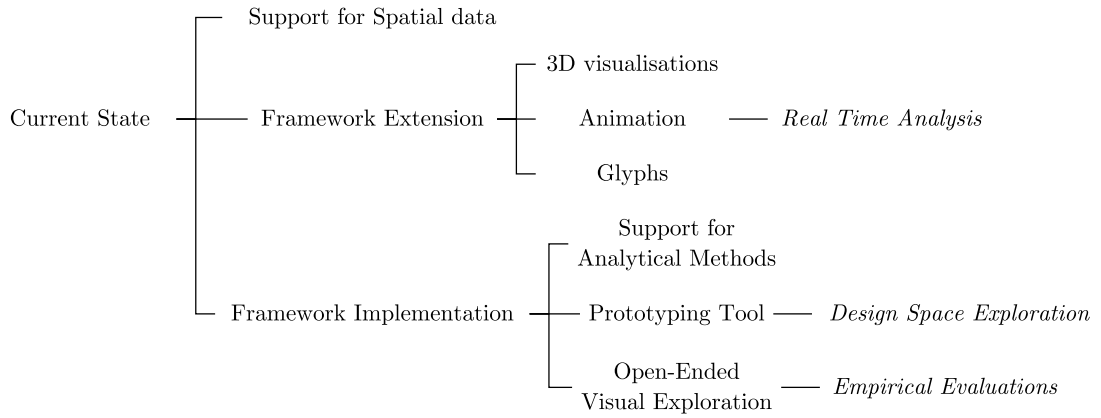


Fig. 9.1 Summary of the research agenda. Beyond the current state, the three potential avenues of future work can be done in parallel. Within each area, specific research paths can also be done in parallel. The highlighted benefits in italic are enabled by the associated future work.

require a rethink of composition methods and visualisation and would likely take a longer research project to address it.

A third possibility is to focus on expanding the scope of the framework towards different areas, such as complementing it with spatial aspects, or more support for analytical methods. As discussed, there are several aspects of spatial data that would require consideration when integrating with the framework; an interesting question is finding out how to adapt existing techniques for time-oriented visualisations for spatiotemporal tasks and how does the framework help with that. Including support for analytical methods will likely be easier if the framework is implemented, because there are no major conceptual limitations for this to happen – it is important to see in practice how the transformations can be combined with the application of such methods.

9.3 Conclusion

This thesis investigated which aspects of hierarchical visualisations and time-oriented visualisations help with visual exploration of temporal data. The result of the research is a framework that allows a structured design of visualisations at the intersection of the two areas. The development of the framework was motivated by the gaps that were identified at this intersection, namely the absence of temporal data in frameworks for hierarchical visualisation and the sub-optimal consideration of interactions based

on the properties of temporal data in frameworks for time-oriented visualisations. By addressing the gaps and combining aspects of the two areas, the thesis opens new visual exploration and research pathways to stimulate further investigation of hierarchical time-oriented visualisations. The methodology that was used to design the framework also enabled the generation of self-contained secondary contributions that can be applied outside the main scope of thesis.

References

Adnan, M., Just, M., and Baillie, L. (2016). Investigating Time Series Visualisations to Improve the User Experience. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*, pages 5444–5455.

Aigner, W., Bertone, A., Miksch, S., Tominski, C., and Schumann, H. (2007a). Towards a conceptual framework for visual analytics of time and time-oriented data. In *2007 Winter Simulation Conference*, pages 721–729. IEEE.

Aigner, W., Miksch, S., Müller, W., Schumann, H., and Tominski, C. (2007b). Visualizing time-oriented data - A systematic view. *Computers & Graphics*, 31(3):401–409.

Aigner, W., Miksch, S., Schumann, H., and Tominski, C. (2011). *Visualization of Time-Oriented Data*. Springer London.

Albers, D., Correll, M., and Gleicher, M. (2014). Task-driven evaluation of aggregation in time series visualization. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*, pages 551–560, New York, New York, USA. ACM Press.

Allen, J. F. and Hayes, P. J. (1985). A Common-Sense Theory of Time. *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 528–531.

André, P., Wilson, M. L., Russell, A., Smith, D. A., Owens, A., and Schraefel, M. (2007). Continuum: designing timelines for hierarchies, relationships and scale. In *Proceedings of the 20th annual ACM symposium on User interface software and technology - UIST '07*, page 101, New York, New York, USA. ACM Press.

Andrienko, G., Andrienko, N., Bak, P., Keim, D., Kisilevich, S., and Wrobel, S. (2011). A conceptual framework and taxonomy of techniques for analyzing movement. *Journal of Visual Languages and Computing*, 22(3):213–232.

Andrienko, N. and Andrienko, G. (2006). *Exploratory analysis of spatial and temporal data: a systematic approach*. Springer Berlin Heidelberg.

Andrienko, N. and Andrienko, G. (2012). Visual analytics of movement: An overview of methods, tools and procedures. *Information Visualization*, 12(1):3–24.

Bach, B., Dragicevic, P., Archambault, D., Hurter, C., and Carpendale, S. (2016a). A Descriptive Framework for Temporal Data Visualizations Based on Generalized Space-Time Cubes. *Computer Graphics Forum*.

Bach, B., Shi, C., Heulot, N., Madhyastha, T., Grabowski, T., and Dragicevic, P. (2016b). Time Curves: Folding Time to Visualize Patterns of Temporal Evolution in Data. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):559–568.

Bak, P., Mansmann, F., Janetzko, H., and Keim, D. (2009). Spatiotemporal Analysis of Sensor Logs using Growth Ring Maps. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):913–920.

Bale, K., Chapman, P., Barraclough, N., Purdy, J., Aydin, N., and Dark, P. (2007). Kaleidomaps: a new technique for the visualization of multivariate time-series data. *Information Visualization*, 6(2):155–167.

Baudel, T. and Broeskema, B. (2012). Capturing the design space of sequential space-filling layouts. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2593–2602.

Beard, K., Deese, H., and Pettigrew, N. R. (2008). A framework for visualization and exploration of events. *Information Visualization*, 7(2):133–151.

Bettini, C., Dyreson, C. E., Evans, W. S., Snodgrass, R. T., and Wang, X. S. (1998). A Glossary of Time Granularity Concepts. *Temporal Databases: Research and Practice LNCS 1399*, pages 406–413.

Borgo, R., Kehrer, J., Chung, D. H. S., Maguire, E., Laramee, R. S., Hauser, H., Ward, M., and Chen, M. (2013). Glyph-based Visualization: Foundations, Design Guidelines, Techniques and Applications. In *Eurographics State of the Art Reports*.

Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (1994). *Time Series Analysis: Forecasting & Control*. Prentice Hall.

Boyandin, I., Bertini, E., Bak, P., and Lalanne, D. (2011). Flowstrates: An approach for visual exploration of temporal origin-destination data. *Computer Graphics Forum*, 30(3):971–980.

Brehmer, M., Lee, B., Bach, B., Riche, N. H., and Munzner, T. (2017). Timelines Revisited: A Design Space and Considerations for Expressive Storytelling. *IEEE Transactions on Visualization and Computer Graphics*, 23(9):2151–2164.

Buchin, K., van Goethem, A., Hoffmann, M., van Kreveld, M., and Speckmann, B. (2014). Travel-Time Maps: Linear Cartograms with Fixed Vertex Locations. In Duckham, M., Pebesma, E., Stewart, K., and Frank, A. U., editors, *Geographic Information Science*, pages 18–33, Cham. Springer International Publishing.

Buchin, M., Kruckenberg, H., and Kölzsch, A. (2013). Segmenting trajectories by movement states. In *Advances in Geographic Information Science*, pages 15–25.

Buja, A., Cook, D., and Swayne, D. F. (1996). Interactive high-dimensional data visualization. *Journal of computational and graphical statistics*, 5(1):78–99.

Burch, M., Beck, F., and Diehl, S. (2008). Timeline trees: visualizing sequences of transactions in information hierarchies. In *Proceedings of the working conference on Advanced visual interfaces - AVI '08*, page 75, New York, New York, USA. ACM Press.

Card, S. K. and Mackinlay, J. (1997). The structure of the information visualization design space. *Information Visualization, 1997. Proceedings., IEEE Symposium on*, pages 92–99.

Card, S. K., Mackinlay, J. D., and Shneiderman, B. (1999). Readings in Information Visualization: Using Vision to Think. In *Information Display*, page 686. Morgan Kaufmann.

Carlis, J. V. and Konstan, J. A. (1998). Interactive Visualization of Serial Periodic Data. *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology*, pages 29–38.

Chi, E. (2000). A taxonomy of visualization techniques using the data state reference model. In *Proceedings of IEEE Symposium on Information Visualization 2000*, pages 69–75. IEEE Comput. Soc.

Cho, M., Kim, B., Bae, H. J., and Seo, J. (2014). Stroscope: Multi-scale visualization of irregularly measured time-series data. *IEEE Transactions on Visualization and Computer Graphics*, 20(5):808–821.

Chuah, M. and Roth, S. (1996). On the semantics of interactive visualizations. *Proceedings IEEE Symposium on Information Visualization '96*, pages 29–36.

Cleveland, W. S. (1993). *Visualizing Data*. Hobart Press.

Cockburn, A., Karlson, A., and Bederson, B. B. (2008). A review of overview+detail, zooming, and focus+context interfaces. *ACM Computing Surveys*, 41(1):1–31.

Cousins, S. B. and Kahn, M. G. (1991). The visual display of temporal information. *Artificial Intelligence in Medicine*, 3(6):341–357.

Daassi, C., Nigay, L., and Fauvet, M. (2005). A taxonomy of temporal data visualization techniques. *Information-Interaction-Intelligence*, 5(2):41–63.

Demšar, U., Buchin, K., Cagnacci, F., Safi, K., Speckmann, B., Van de Weghe, N., Weiskopf, D., and Weibel, R. (2015). Analysis and visualisation of movement: an interdisciplinary review. *Movement Ecology*, 3(1):1–24.

Dix, A. and Ellis, G. (1998). Starting Simple - adding value to static visualisation through simple interaction. In *Advanced Visual Interfaces AVI98*, pages 124–134.

Dyreson, C., Evans, W., Lin, H., and Snodgrass, R. (2000). Efficiently supporting temporal granularities. *IEEE Transactions on Knowledge and Data Engineering*, 12(4):568–587.

Elmqvist, N. and Fekete, J. D. (2010). Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):439–454.

Fereday, J. and Muir-Cochrane, E. (2006). Demonstrating Rigor Using Thematic Analysis: A Hybrid Approach of Inductive and Deductive Coding and Theme Development. *International Journal of Qualitative Methods*, 5(1):80–92.

Ferreira, N., Poco, J., Vo, H. T., Freire, J., and Silva, C. T. (2013). Visual exploration of big spatio-temporal urban data: A study of New York city taxi trips. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2149–2158.

Fischer, F., Fuchs, J., and Mansmann, F. (2012). ClockMap: Enhancing circular treemaps with temporal glyphs for time-series data. *Proceedings of the Eurographics Conference on Visualization (EuroVis 2012)*, pages 97–101.

Frank, A. U. (1998). Different Types of "Times " in GIS. In Egenhofer, M. J. and Golledge, R. G., editors, *Spatial and Temporal reasonning in Geographic Information Systems*, pages 40–61. Oxford University Press, New York.

Funkhouser, H. G. (1936). A Note on a Tenth Century Graph. *Osiris*, 1(1):260.

Furnas, G. W. (1986). Generalized fisheye views. *ACM Conference on Human Factors in Computing Systems*, pages 16–23.

Gad, S., Javed, W., Ghani, S., Elmqvist, N., Ewing, T., Hampton, K. N., and Ramakrishnan, N. (2015). ThemeDelta: Dynamic segmentations over temporal topic models. *IEEE Transactions on Visualization and Computer Graphics*, 21(5):672–685.

Gleicher, M., Albers, D., Walker, R., Jusufi, I., Hansen, C. D., and Roberts, J. C. (2011). Visual comparison for information visualization. *Information Visualization*, 10(4):289–309.

Goralwalla, I. a., Özsu, M. T., and Szafron, D. (1998). An Object-Oriented Framework for Temporal Data Models. *Temporal Databases: Research and Practice*, pages 1–35.

Gschwandtner, T., Aigner, W., Kaiser, K., Miksch, S., and Seyfang, A. (2011). CareCruiser: Exploring and visualizing plans, events, and effects interactively. *IEEE Pacific Visualization Symposium 2011, PacificVis 2011 - Proceedings*, pages 43–50.

Gschwandtner, T., Gärtner, J., Aigner, W., and Miksch, S. (2012). A Taxonomy of Dirty Time-Oriented Data. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7465 LNCS, pages 58–72. Springer, Berlin, Heidelberg.

Guerra-Gomez, J., Pack, M. L., Plaisant, C., and Shneiderman, B. (2013). Visualizing change over time using dynamic hierarchies: Treeversity2 and the stemview. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2566–2575.

Guo, D., Chen, J., MacEachren, A. M., and Liao, K. (2006). A Visualization System for Space-Time and Multivariate Patterns (VIS-STAMP). *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1461–1474.

Haber, R. B. and Mcnabb, D. A. (1990). Visualization ldioms : A Conceptual Model Visualization for Scientific Systems. *Visualization in scientific computing*, 74:93.

Hadlak, S., Tominski, C., Schulz, H.-J., and Schumann, H. (2010). Visualization of attributed hierarchical structures in a spatiotemporal context. *International Journal of Geographical Information Science*, 24(10):1497–1513.

Halberg, F. (1969). Chronobiology. *Annual Review of Physiology*, 31(1):675–726.

Haroz, S., Kosara, R., and Franconeri, S. L. (2016). The Connected Scatterplot for Presenting Paired Time Series. *IEEE Transactions on Visualization and Computer Graphics*, 22(9):2174–2186.

Harris, R. L. (1999). *Information Graphics: A Comprehensive Illustrated Reference*. Oxford University Press, Inc., New York, NY, USA.

Harrison, B., Owen, R., and Baecker, R. (1994). Timelines: an interactive system for the collection and visualization of temporal data. In *Graphics Interface*, pages 141–141.

Harrison, L., Yang, F., Franconeri, S., and Chang, R. (2014). Ranking Visualizations of Correlation Using Weber's Law. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1943–1952.

Havre, S., Hetzler, E., Whitney, P., and Nowell, L. (2002). ThemeRiver: Visualizing thematic changes in large document collections. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):9–20.

Heer, J. and Shneiderman, B. (2012). Interactive dynamics for visual analysis. *Communications of the ACM*, 55(4):45–54.

Hochheiser, H. and Shneiderman, B. (2004). Dynamic query tools for time series data sets: Timebox widgets for interactive exploration. *Information Visualization*, 3(1):1–18.

Jankun-Kelly, T. J., Ma, K. L., and Gertz, M. (2007). A model and framework for visualization exploration. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):357–368.

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

Javed, W. and Elmqvist, N. (2013). Stack Zooming for Multifocus Interaction in Skewed-Aspect Visual Spaces. *IEEE Transactions on Visualization and Computer Graphics*, 19(8):1362–1374.

Javed, W., McDonnel, B., and Elmqvist, N. (2010). Graphical perception of multiple time series. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):927–934.

Keim, D. A., Schneidewind, J., and Sips, M. (2004). CircleView. In *Proceedings of the working conference on Advanced visual interfaces - AVI '04*, page 179, New York, New York, USA. ACM Press.

Keogh, E., Chu, S., Hart, D., and Pazzani, M. (1993). Segmenting Time Series: A Survey and Novel Approach. *Data Mining in Time Series Databases*, pages 1–21.

Kincaid, R. (2010). SignalLens: Focus plus Context Applied to Electronic Time Series. *Ieee Transactions on Visualization and Computer Graphics*, 16(6):900–907.

Kothur, P., Sips, M., Unger, A., Kuhlmann, J., and Dransch, D. (2013). Interactive visual summaries for detection and assessment of spatiotemporal patterns in geospatial time series. *Information Visualization*, 13(3):283–298.

Kraak, M.-j. (2008). Time – New Opportunities for the Space – Time Cube. *Geographic Visualization: Concepts, Tools and Applications*.

Krstajic, M., Bertini, E., and Keim, D. (2011). CloudLines: Compact Display of Event Episodes in Multiple Time-Series. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2432–2439.

Lammarsch, T., Aigner, W., Bertone, A., Gärtner, J., Mayr, E., Miksch, S., and Smuc, M. (2009). Hierarchical temporal patterns and interactive aggregated views for pixel-based visualizations. *Proceedings of the International Conference on Information Visualisation*, pages 44–50.

Lawvere, F. W. and Rosebrugh, R. (2003). *Abstract Sets and Mappings*, pages 1–25. Cambridge University Press.

Lei, S. T. and Zhang, K. (2010). A visual analytics system for financial time-series data. *Proceedings of the 3rd International Symposium on Visual Information Communication - VINCE '10*, page 1.

Li, X. and Kraak, M.-J. (2013). The Time Wave. A New Method of Visual Exploration of Geo-data in Time–space. *The Cartographic Journal*, 45(3):193–200.

Loorak, M. H., Perin, C., Kamal, N., Hill, M., and Carpendale, S. (2016). TimeSpan: Using Visualization to Explore Temporal Multi-dimensional Data of Stroke Patients. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):409–418.

MacNeil, S. and Elmqvist, N. (2013). Visualization mosaics for multivariate visual exploration. *Computer Graphics Forum*, 32(6):38–50.

McLachlan, P., Munzner, T., Koutsofios, E., and North, S. (2008). LiveRAC: Interactive Visual Exploration of System Management Time-Series Data. In *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, page 1483, New York, New York, USA. ACM Press.

Muller, W. and Schumann, H. (2003). Visualization methods for time-dependent data - an overview. *Proceedings of the 2003 Winter Simulation Conference, 2003.*, 1.

Munzner, T. (2014). *Visualization Analysis and Design*. CRC Press.

Nathan, R., Getz, W. M., Revilla, E., Holyoak, M., Kadmon, R., Saltz, D., and Smouse, P. E. (2008). A movement ecology paradigm for unifying organismal movement research. *Proceedings of the National Academy of Sciences of the United States of America*, 105(49):19052–9.

Norman, D. A. (1988). *The Design of Everyday Things*. Basic Books, New York, NY.

Openshaw, S. (1983). The modifiable area unit problem. *Concepts and Techniques in Modern Geography*, 38:1–41.

Pani, A. K. and Bhattacharjee, G. P. (2001). Temporal representation and reasoning in artificial intelligence: A review. *Mathematical and Computer Modelling*, 34(1-2):55–80.

Peuquet, D. J. (1994). It's About Time: A Conceptual Framework for the Representation of Temporal Dynamics in Geographic Information Systems. *Annals of the Association of American Geographers*, 84(3):441–461.

Pike, W. A., Stasko, J., Chang, R., and O'Connell, T. A. (2009). The Science of Interaction. *Information Visualization*, 8(4):263–274.

Qiang, Y., Delafontaine, M., Versichele, M., De Maeyer, P., and Van de Weghe, N. (2012). Interactive analysis of time intervals in a two-dimensional space. *Information Visualization*, 11(4):255–272.

Qlik (2018). Qlik Sense.

Ragan, E. D., Endert, A., Sanyal, J., and Chen, J. (2015). Characterizing Provenance in Visualization and Data Analysis : An Organizational Framework of Provenance Types and Purposes. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–10.

Rind, A., Lammarsch, T., Aigner, W., Alsallakh, B., and Miksch, S. (2013). TimeBench: A data model and software library for visual analytics of time-oriented data. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2247–2256.

Rit, J. (1986). Propagating Temporal Constraints for Scheduling. In *Proceedings of the Fifth AAAI National Conference on Artificial Intelligence*, volume 86, pages 383–388, Philadelphia. AAAI Press.

Roberts, J. C. (2007). State of the art: Coordinated & multiple views in exploratory visualization. In *Proceedings - Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization, CMV 2007*, pages 61–71.

Roth, R. E. (2012). Cartographic Interaction Primitives: Framework and Synthesis. *The Cartographic Journal*, 49(4):376–395.

Roth, R. E. (2013). An empirically-derived taxonomy of interaction primitives for interactive cartography and geovisualization. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2356–2365.

Satyanarayan, A., Moritz, D., Wongsuphasawat, K., and Heer, J. (2016). Vega-Lite: A Grammar of Interactive Graphics. *IEEE Transactions on Visualization and Computer Graphics*, 2626(c):1–1.

Schulz, H. J., Hadlak, S., and Schumann, H. (2011). The design space of implicit hierarchy visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 17(4):393–411.

Shahar, Y., Goren-Bar, D., Boaz, D., and Tahan, G. (2006). Distributed, intelligent, interactive visualization and exploration of time-oriented clinical data and their abstractions. *Artificial Intelligence in Medicine*, 38(2):115–135.

Shen, Z. and Kwan-Liu, M. (2008). MobiVis: A visualization system for exploring mobile data. In *IEEE Pacific Visualisation Symposium 2008, PacificVis - Proceedings*, pages 175–182.

Shneiderman, B. (1996). The eyes have it: a task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pages 336–343. IEEE Comput. Soc. Press.

Shrinivasan, Y. B. and van Wijk, J. J. (2008). Supporting the analytical reasoning process in information visualization. *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, page 1237.

Silva, S. and Catarci, T. (2000). Visualization of linear time-oriented data: a survey. *Proceedings of the First International Conference on Web Information Systems Engineering*, 1(June).

Sips, M., Kothur, P., Unger, A., Hege, H. C., and Dransch, D. (2012). A visual analytics approach to multiscale exploration of environmental time series. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2899–2907.

Slingsby, A., Dykes, J., and Wood, J. (2009). Configuring hierarchical layouts to address research questions. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):977–984.

Slonneger, K. and Kurtz, B. L. (1995). *Formal Syntax and Semantics of Programming Languages: A Laboratory Based Approach*. Addison-Wesley.

St. John, M., Cowen, M. B., Smallman, H. S., and Oonk, H. M. (2001). The Use of 2D and 3D Displays for Shape-Understanding versus Relative-Position Tasks. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 43(1):79–98.

Stolte, C., Tang, D., and Hanrahan, P. (2002). Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):52–65.

Tableau Software (2018). Tableau.

Tan, P.-N., Steinbach, M., and Kumar, V. (2005). *Introduction to data mining*. Addison-Wesley.

TIBCO Software Inc. (2018). TBICO Spotfire.

Tominski, C. and Schumann, H. (2008). Enhanced Interactive Spiral Display. *The Annual SIGRAD Conference Special Theme: Interaction*, pages 53–56.

Tominski, C., Schumann, H., Andrienko, G., and Andrienko, N. (2012). Stacking-based visualization of trajectory attribute data. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2565–2574.

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

Tukey, J. W. (1977). *Exploratory Data Analysis.* Pearson.

van der Corput, P. and van Wijk, J. J. (2017). Comparing Personal Image Collections with PICTuReVis. *Computer Graphics Forum*, 36(3):295–304.

Van Wijk, J. and Van Selow, E. (1999). Cluster and calendar based visualization of time series data. In *Proceedings 1999 IEEE Symposium on Information Visualization (InfoVis'99)*, pages 4–9,. IEEE Comput. Soc.

Van Wijk, J. J. (2005). The value of visualization. *Proceedings of the IEEE Visualization Conference*, page 11.

Wang, T., Plaisant, C., Shneiderman, B., Spring, N., Roseman, D., Marchand, G., Mukherjee, V., and Smith, M. (2009). Temporal Summaries: Supporting Temporal Categorical Searching, Aggregation and Comparison. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1049–1056.

Wang Baldonado, M. Q., Woodfruss, A., and Kuchinsky, A. (2000). Guidelines for using multiple views in information visualization. In *AVI '00 Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 110 – 119.

Weber, M., Alexa, M., and Muller, W. (2001). Visualizing time-series on spirals. In *IEEE Symposium on Information Visualization 2001 (INFOVIS 2001)*, pages 7–13. IEEE.

Wickham, H. (2010). A Layered Grammar of Graphics. *Journal of Computational and Graphical Statistics*, 19(1):3–28.

Wickham, H. and Hofmann, H. (2011). Product plots. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2223–2230.

Wilkinson, L. (2005). *The grammar of graphics.* Springer, New York.

Wongsuphasawat, K. and Shneiderman, B. (2009). Finding comparable temporal categorical records: A similarity measure with an interactive visualization. In *2009 IEEE Symposium on Visual Analytics Science and Technology*, pages 27–34. IEEE.

Yi, J. S., ah Kang, Y., Stasko, J. T., and Jacko, J. A. (2007). Toward a Deeper Understanding of the Role of Interaction in Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1224–1231.

Zhao, J., Chevalier, F., and Balakrishnan, R. (2011). KronoMiner. In *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11*, page 1737, New York, New York, USA. ACM Press.

Zhao, J., Drucker, S. M., Fisher, D., and Brinkman, D. (2012). TimeSlice. In *Proceedings of the International Working Conference on Advanced Visual Interfaces - AVI '12*, page 433, New York, New York, USA. ACM Press.

Zhou, M. and Feiner, S. (1998). Visual task characterization for automated visual discourse synthesis. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 392–399.

Appendix A

Survey and specifications of encodings

This appendix contains the references found from the survey described in chapter 5. Each reference is divided into two parts: the description of the visual mappings and the specifications according to the framework and the use of the conditioning variables and composition method. They are formatted in the same manner that was done throughout the thesis. Entries are in alphabetical order by the author's surname and title.

The following abbreviations are used as headers in the tables:

- **C**: Colour
- **S**: Size
- **P**: Path
- **Cond.**: Conditioning
- **L**: Layer number
- **CV**: Conditioning variable
- **CM**: Composition method

Continuum (André et al., 2007)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Time	SCALE	–	–	–	✓	-	BAR	1-to-1	✓
Attribute	SCALE	–	✓	–	-	✓	-	1-to-1	-

Layers:

Time histogram			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Time**	–	–	$[-, \mathbf{A}]$	-	**T**	**A**	SCALE	BAR

Table A.1 Signature: $(\mathbf{T}, L_1)$; $\mathbf{A}$ = Item count

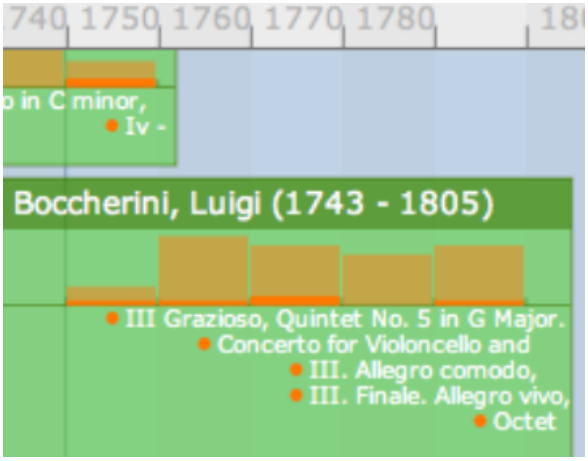


Fig. A.1 Continuum (André et al., 2007)

Time Curves (Bach et al., 2016b)

Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Spatial	MDS	–	–	–	✓	✓	-	2-to-2	-
Time	SCALE	✓	–	✓	-	-	POINT	1-to-1	✓

Layers:

Time Curves			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Time**	–	**T**	–	**T**	**S**	**S**	SCALE	POINT

Table A.2 Signature: (**T**,L_1); **S** = MDS coordinate

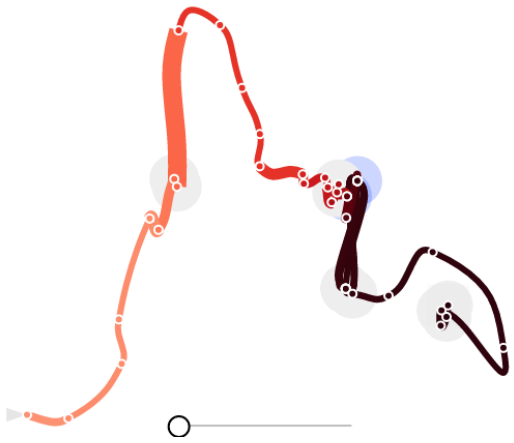


Fig. A.2 Time Curves (Bach et al., 2016b)

Kaleidomaps (Bale et al., 2007)
See fig. 5.8.
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Identifier	SCALE	–	–	–	✓	-	SPINE	1-to-1	✓
Time	SCALE	–	–	–	✓	-	SPINE	1-to-1	✓
Time	SCALE	–	–	–	-	✓	SPINE	1-to-1	✓
Attribute	SCALE	✓	–	–	-	-	-	-	-

Layers:

Kaleidomaps specification			POLAR, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Identifier**	–	–	–	-	**I**	-	SCALE	SPINE
2	**Time 1**	*nesting*	–	–	-	$\mathbf{T_1}$	-	SCALE	SPINE
3	**Time 2**	*nesting*	**A**	–	-	-	$\mathbf{T_2}$	SCALE	SPINE

Table A.3 Signature: $N((\mathbf{I},L_1),N((\mathbf{T},L_2),(\mathbf{T},L_3)))$

Flowstrates (Boyandin et al., 2011)

Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Identifier	SCALE	–	–	–	-	✓	SPINE	1-to-1	✓
Time	SCALE	–	–	–	✓	-	SPINE	1-to-1	✓
Attribute	SCALE	–	✓	–	-	✓	-	1-to-1	-

Layers:

Flowstrates heat map			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Identifier**	–	–	–	-	-	**I**	SCALE	SPINE
2	**Time**	*nesting*	**A**	–	-	**T**	-	SCALE	SPINE

Table A.4 Signature: $N((\mathbf{I},L_1),(\mathbf{T},L_2))$

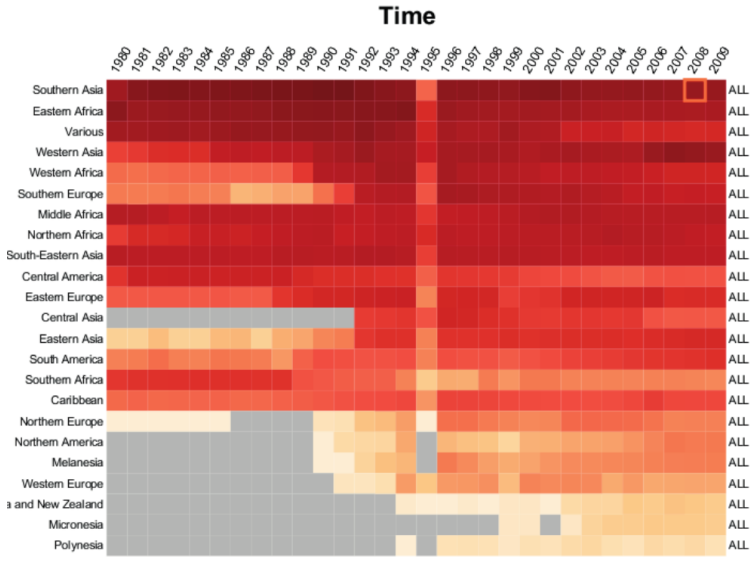


Fig. A.3 Flowstrates (Boyandin et al., 2011)

Timeline Trees (Burch et al., 2008)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Identifier	SCALE	–	–	–	-	✓	SPINE	1-to-1	✓
Time	SCALE	✓	–	–	✓	-	BAR	1-to-1	✓
Attribute	SCALE	–	✓	–	-	-	-	1-to-1	-

Layers:

Timeline Trees			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Identifier**	–	–	–	-	-	**I**	SCALE	BAR
2	**Time**	*nesting*	**T**	[**A**, –]	-	**T**	-	SCALE	SPINE

Table A.5 Signature: $N((\mathbf{I},L_1),(\mathbf{T},L_2)$

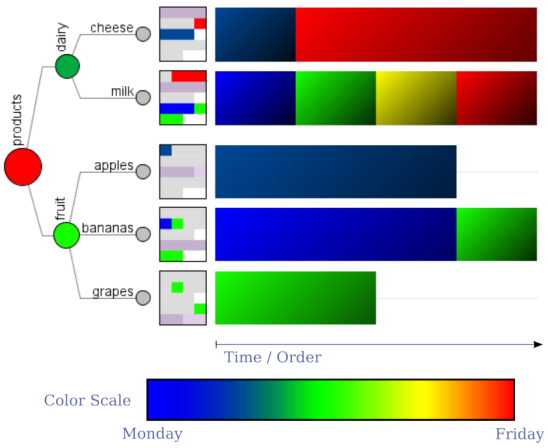


Fig. A.4 Timeline Trees (Burch et al., 2008)

Spiral (Carlis and Konstan, 1998)

Carlis and Konstan (1998) suggested two types of spirals: one for instant time and one for interval time.

Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Time 1	SPIRAL	–	–	–	✓	✓	POINT	n-to-2	✓
Time 2	SPIRAL	–	–	–	✓	✓	POINT	n-to-2	✓

Layers:

Spiral			POLAR, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	$[\mathbf{T_1}, \mathbf{T_2}]$	–	–	–	-	$[\mathbf{T_1}, \mathbf{T_2}]$	$[\mathbf{T_1}, \mathbf{T_2}]$	SPIRAL	POINT

Table A.6 Signature: ($[\mathbf{T_1}, \mathbf{T_2}]$,$L_1$)

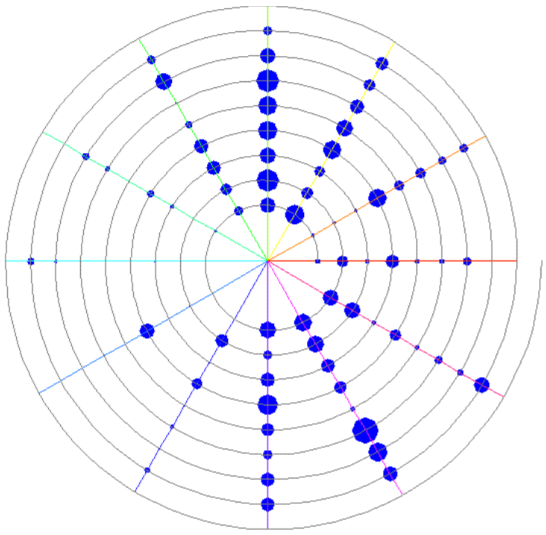


Fig. A.5 Instants spiral (Carlis and Konstan, 1998)

Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Time 1	SPIRAL	–	–	–	✓	✓	-	n-to-2	✓
Time 2	SPIRAL	–	–	–	✓	✓	-	n-to-2	✓
Time 3	SPIRAL	–	✓	–	-	-	BAR	n-to-2	-

Layers:

Spiral			POLAR, INTERVAL						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	$[\mathbf{T_1}, \mathbf{T_2}]$	–	–	$\mathbf{T_3}$	-	$[\mathbf{T_1}, \mathbf{T_2}]$	$[\mathbf{T_1}, \mathbf{T_2}]$	SPIRAL	BAR

Table A.7 Signature: $([\mathbf{T_1}, \mathbf{T_2}], L_1).\mathbf{T_3}$ = Interval duration

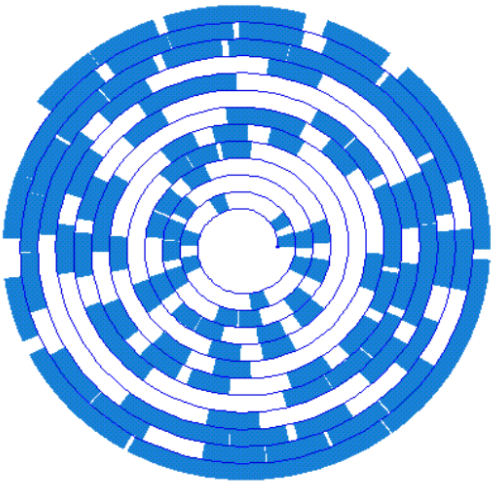


Fig. A.6 Interval spiral (Carlis and Konstan, 1998)

Cycle plot (Cleveland, 1993)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Time 1	SCALE	–	–	–	✓	-	SPINE	1-to-1	✓
Time 2	SCALE	–	–	✓	✓	-	POINT	1-to-1	✓
Attribute	SCALE	–	–	–	-	✓	-	1-to-1	-

Layers:

Cycle plot			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Time 1**	–	–	–	-	$\mathbf{T_1}$	-	SCALE	SPINE
2	**Time 2**	*nesting*	–	–	$\mathbf{T_2}$	$\mathbf{T_2}$	**A**	SCALE	POINT

Table A.8 Signature: $N((\mathbf{T_1},L_1),(\mathbf{T_2},L_2))$; $\mathbf{T_2}$ is *coarser* than $\mathbf{T_1}$

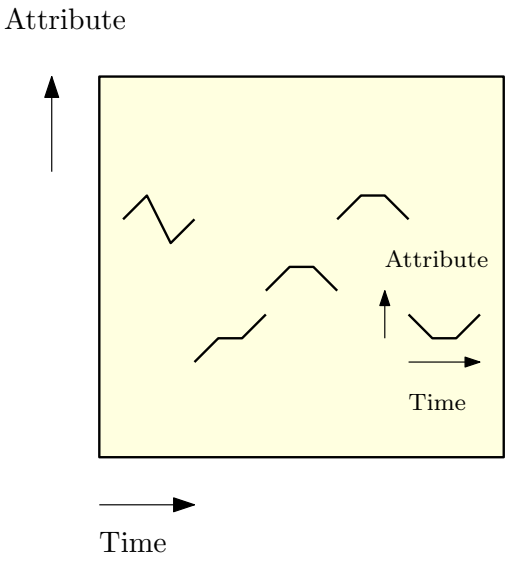


Fig. A.7 Cycle plot

ClockMap (Fischer et al., 2012)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Identifier	SCALE	–	–	–	-	-	CIRCLE	1-to-1	✓
Attr 1	ALGORITHM	–	✓	–	✓	✓	-	n-to-2	-
Time	SCALE	–	–	–	✓	-	SPINE	1-to-1	✓
Attr 2	SCALE	✓	–	–	-	-	-	1-to-1	-

Layers:

ClockMap			POLAR, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Identifier**	–	–	**A***	-	**A***	**A***	ALGORITHM	CIRCLE
2	**Time**	*nesting*	**A**	–	-	**T**	-	SCALE	SPINE

Table A.9 Signature: $N((\mathbf{I},L_1),(\mathbf{T},L_2))$; **A*** = Algorithm-generated size and position

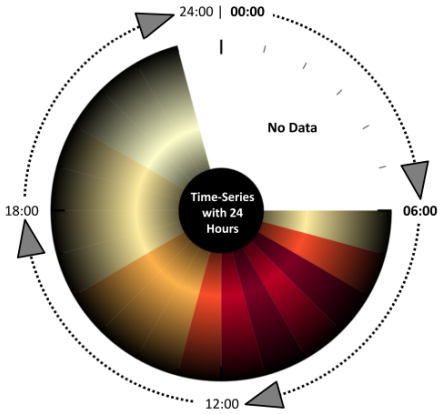


Fig. A.8 ClockMap (Fischer et al., 2012)

ThemeDelta (Gad et al., 2015)

The full replication of this visualisation requires the use of a connecting path *across* nodes, which the framework currently does not support.

Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Time	SCALE	–	–	–	✓	-	SPINE	1-to-1	✓
Identifier	SCALE	–	–	–	✓	-	POINT	1-to-1	✓
Attribute	ALGORITHM	–	–	–	-	✓	-	1-to-1	-

Layers:

ThemeDelta			CARTESIAN, INTERVAL						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Time**	–	–	–	-	**T**	-	SCALE	SPINE
2	**Identifier**	*nesting*	–	–	-	-	**A***	ALGORITHM	POINT

Table A.10 $N((\mathbf{T},L_1),(\mathbf{I},L_2))$; **A*** = Algorithm-generated position

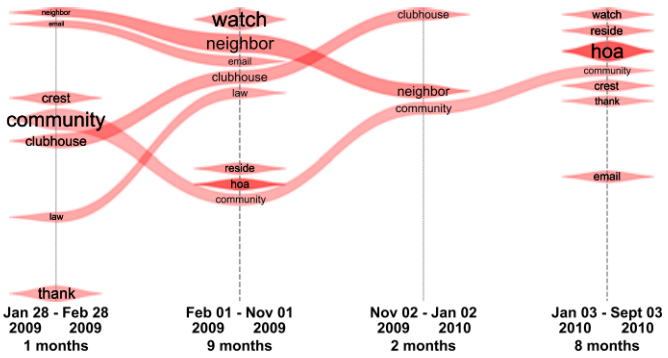


Fig. A.9 ThemeDelta (Gad et al., 2015)

VIS-STAMP (Guo et al., 2006)

Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Identifier	SCALE	–	–	–	-	✓	SPINE	1-to-1	✓
Time	SCALE	–	–	–	✓	-	SPINE	1-to-1	✓
Attribute	SCALE	✓	–	–	-	✓	-	1-to-1	-

Layers:

VIS-STAMP			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Identifier**	–	–	–	-	-	**I**	SCALE	SPINE
2	**Time**	*nesting*	**A**	–	-	**T**	-	SCALE	SPINE

Table A.11 Signature: $N((\mathbf{I},L_1),(\mathbf{T},L_2))$

Fig. A.10 VIS-STAMP (Guo et al., 2006)

CareCruiser (Gschwandtner et al., 2011)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Identifier	SCALE	–	–	–	-	✓	SPINE	1-to-1	✓
Time	SCALE	–	–	✓	✓	-	POINT	1-to-1	✓
Attribute	SCALE	–	–	–	-	✓	-	1-to-1	-

Layers:

CareCruiser			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Identifier**	–	–	–	-	-	**I**	SCALE	SPINE
2	**Time**	*nesting*	–	–	**T**	**T**	**A**	SCALE	POINT

Table A.12 Signature: $N((\mathbf{I},L_1),(\mathbf{T},L_2))$

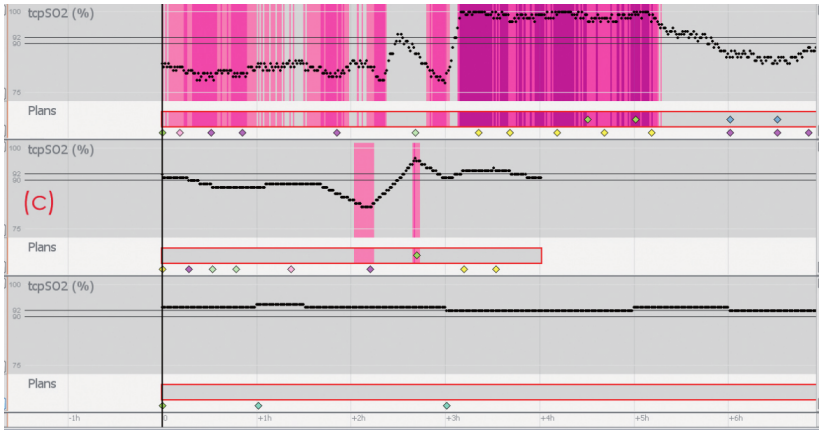


Fig. A.11 CareCruiser (Gschwandtner et al., 2011)

Connected scatterplot (Haroz et al., 2016)

Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Time	SCALE	–	–	✓	-	-	POINT	1-to-1	✓
Attribute 1	SCALE	–	✓	–	-	✓	-	1-to-1	-
Attribute 2	SCALE	–	✓	–	-	-	✓	1-to-1	-

Layers:

Connected scatterplot			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Time**	–	–	–	**T**	**A**	**A**	SCALE	POINT

Table A.13 Signature: (**T**,L_1)

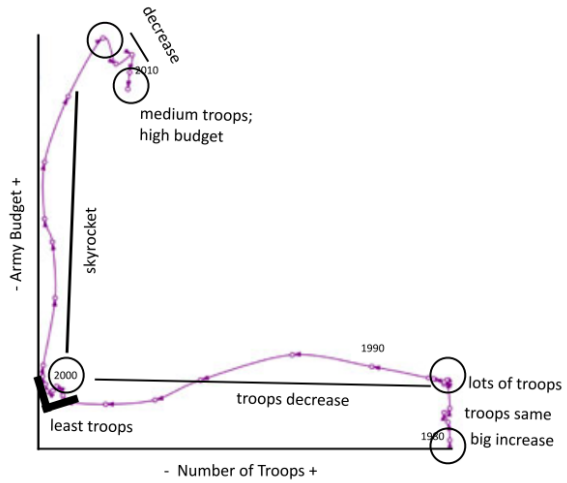


Fig. A.12 Connected scatterplot (Haroz et al., 2016)

Point chart (Harris, 1999)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Time	SCALE	–	–	–	✓	-	POINT	1-to-1	✓
Attribute	SCALE	–	–	–	-	✓	-	1-to-1	-

Layers:

Point chart			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Time**	–	–	–	-	**T**	**A**	SCALE	POINT

Table A.14 Signature: ($\mathbf{T}$,L_1)

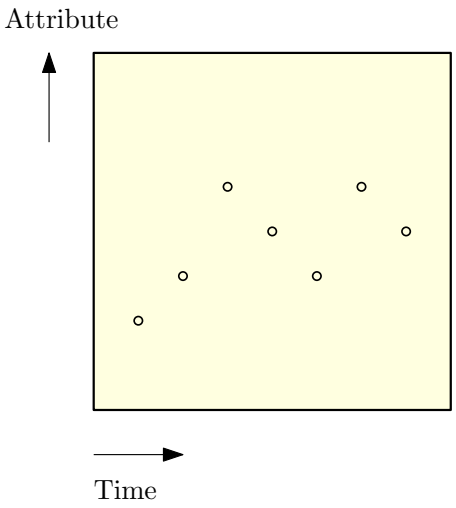


Fig. A.13 Point chart

Bar chart (Harris, 1999)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Time	SCALE	–	–	–	✓	-	BAR	1-to-1	✓
Attribute	SCALE	–	✓	–	-	✓	-	1-to-1	-

Layers:

Bar chart			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Time**	–	–	[–, **A**]	-	**T**	**A**	SCALE	BAR

Table A.15 Signature: (**T**,L_1)

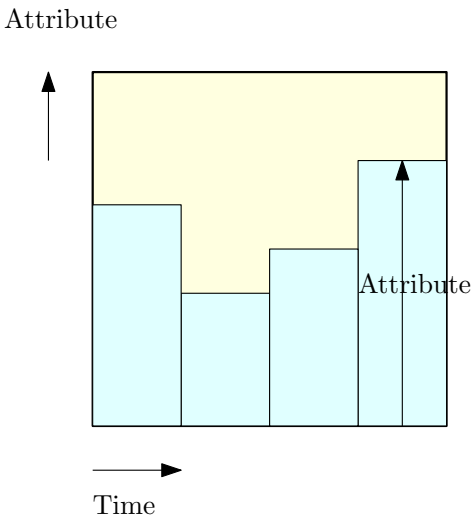


Fig. A.14 Bar chart

Line chart (Harris, 1999)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Time	SCALE	–	–	✓	✓	-	POINT	1-to-1	✓
Attribute	SCALE	–	–	–	-	✓	-	1-to-1	-

Layers:

Line chart			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Time**	–	–	–	**T**	**T**	**A**	SCALE	POINT

Table A.16 Signature: (**T**,L_1)

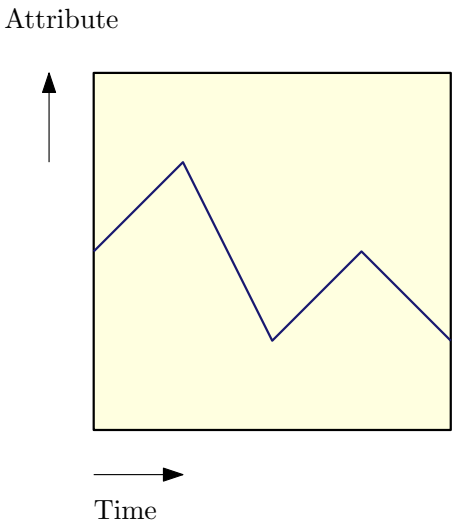


Fig. A.15 Line chart

Polar point chart (Harris, 1999)

Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Time	SCALE	–	–	✓	✓	-	POINT	1-to-1	✓
Attribute	SCALE	–	–	–	-	✓	-	1-to-1	-

Layers:

Point chart			POLAR, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Time**	–	–	–	-	**T**	**A**	SCALE	POINT

Table A.17 Signature: (**T**,L_1)

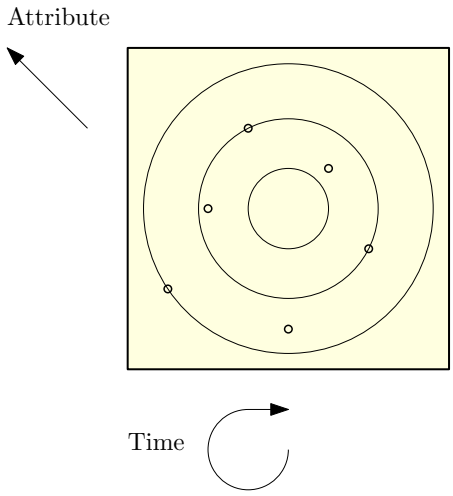


Fig. A.16 Polar point chart

Polar line chart (Harris, 1999)

Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Time	SCALE	–	✓	✓	✓	-	POINT	1-to-1	✓
Attribute	SCALE	–	–	–	-	✓	-	1-to-1	-

Layers:

Line chart			POLAR, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Time**	–	–	–	**T**	**T**	**A**	SCALE	POINT

Table A.18 Signature: (**T**,L_1)

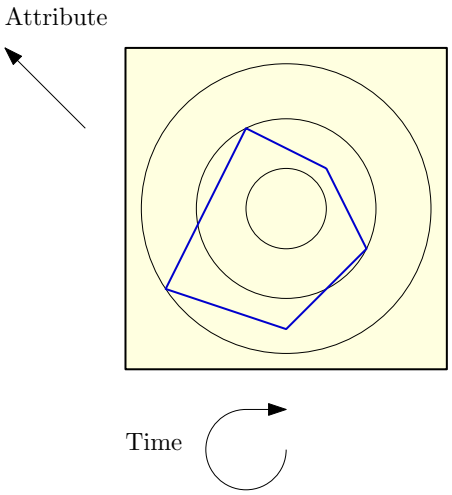


Fig. A.17 Polar line chart

Angular bar chart (Harris, 1999)

Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Identifier	SCALE	–	–	–	-	✓	SPINE	1-to-1	✓
Time	SCALE	–	✓	–	✓	-	BAR	1-to-1	✓

Layers:

Bar chart			POLAR, INTERVAL						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Identifier**	–	–	-	-	-	**I**	SCALE	SPINE
2	**Time**	*nesting*	–	[**T**:duration,-]	-	**T**:start	-	SCALE	BAR

Table A.19 Signature: N((**I**,L_1),(**T**,L_2))

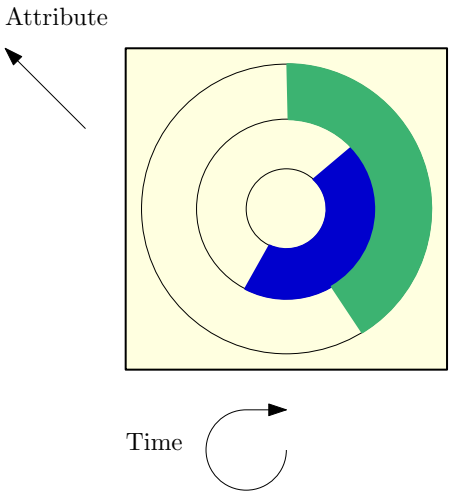


Fig. A.18 Angular bar chart

Radial pie chart (Harris, 1999)

Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Time	SCALE	–	–	–	✓	-	BAR	1-to-1	✓
Attribute	SCALE	–	✓	–	-	✓	-	1-to-1	-

Layers:

Radial pie chart			POLAR, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Time**	–	–	[–, **A**]	-	**T**	**A**	SCALE	BAR

Table A.20 (**T**,L_1)

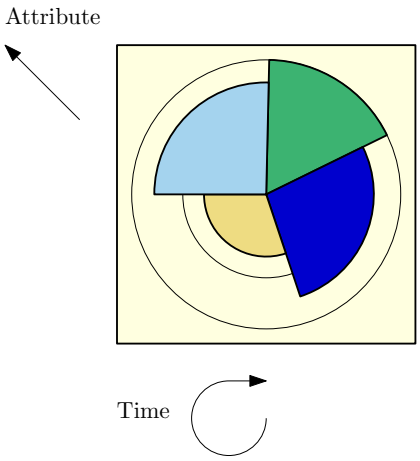


Fig. A.19 Radial pie chart

Dot plot (point histogram) (Harris, 1999)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Time	SCALE	–	–	–	✓	-	POINT	1-to-1	✓
Identifier	SCALE	–	–	–	-	✓	-	1-to-1	-

Layers:

Dot plot histogram			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Time**	–	–	–	-	**T**	**I**	SCALE	POINT

Table A.21 (**T**,L_1); Points are stacked; the sorting order does not matter but an *ordinal scale* is used.

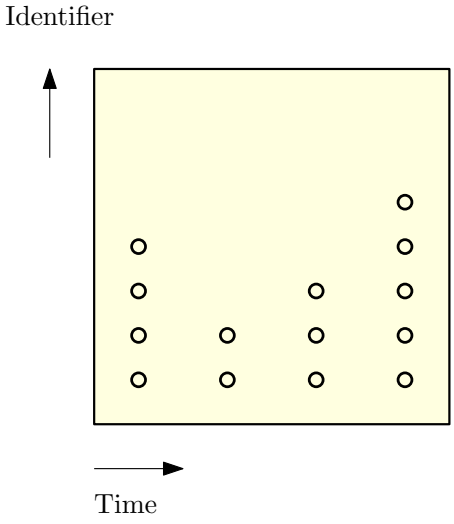


Fig. A.20 Dot plot

Timeline (Harrison et al., 1994)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Identifier	SCALE	–	–	–	-	✓	SPINE	1-to-1	✓
Time	SCALE	–	✓	–	✓	-	BAR	1-to-1	✓

Layers:

Timeline			CARTESIAN, INSTANT-INTERVAL						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Identifier**	–	–	–	-	-	**I**	SCALE	SPINE
2	**Time**	*nesting*	–	**T**:duration	-	**T**:start	-	SCALE	BAR

Table A.22 $N((\mathbf{I},L_1),(\mathbf{T},L_2))$

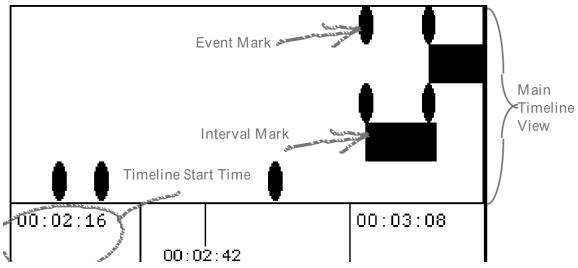


Fig. A.21 Timeline (Harrison et al., 1994)

ThemeRiver (Havre et al., 2002)

Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Identifier	ALGORITHM	–	–	–	✓	✓	POLYGON	1-to-1	✓
Time	SCALE	–	–	–	-	✓	POLYGON	n-to-2	-
Attribute	SCALE	–	✓	–	-	✓	-	-	-

Layers:

ThemeRiver			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Identifier**	–	–	[–, **A**]	-	-	**A**	ALGORITHM	POLYGON
1	**Time**	*nesting*	–	[–, **A**]	-	**T**	**A**	SCALE	POLYGON

Table A.23 $N((\mathbf{I},L_1),(\mathbf{T},L_2))$

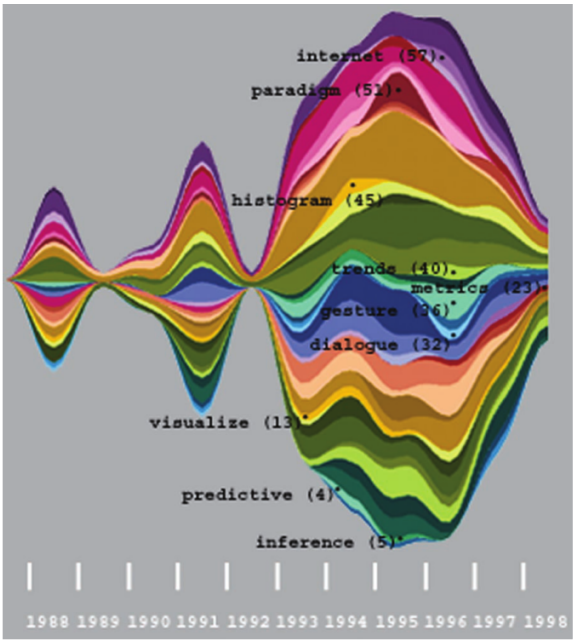


Fig. A.22 ThemeRiver (Havre et al., 2002)

Circle View (Keim et al., 2004)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Identifier	SCALE	–	–	–	✓	-	SPINE	1-to-1	✓
Time	SCALE	–	–	–		✓	SPINE	1-to-1	✓
Attribute	SCALE	✓	–	–	-	-	-	1-to-1	-

Layers:

Circle View			POLAR, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Identifier**	–	–	–	-	**I**	-	SCALE	SPINE
2	**Time**	*nesting*	**A**	–	-	-	**T**	SCALE	SPINE

Table A.24 $N(\mathbf{I},L_1),(\mathbf{T},L_2))$

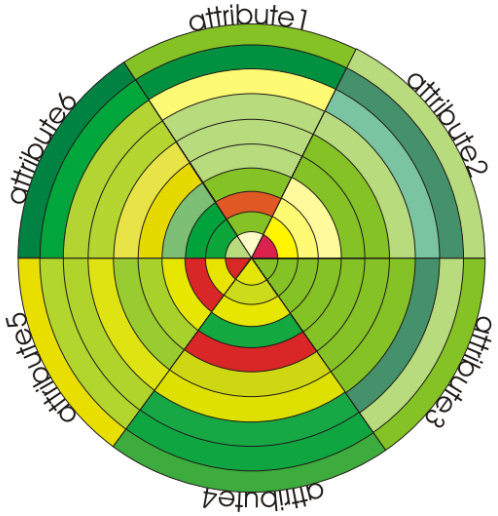


Fig. A.23 Circle View (Keim et al., 2004)

Cloudlines (Krstajic et al., 2011)

Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Identifier	SCALE	–	–	–	-	✓	SPINE	1-to-1	✓
Time	SCALE	–	–	–	✓	-	POINT	1-to-1	✓
Attribute	SCALE	–	✓	–	-	✓	-	1-to-1	-

Layers:

Cloudlines			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Identifier**	–	–	–	-	-	**I**	SCALE	SPINE
2	**Time**	*nesting*	–	**A**	-	**T**	-	SCALE	POINT

Table A.25 Signature: $N((\mathbf{I},L_1)(\mathbf{T},L_2))$

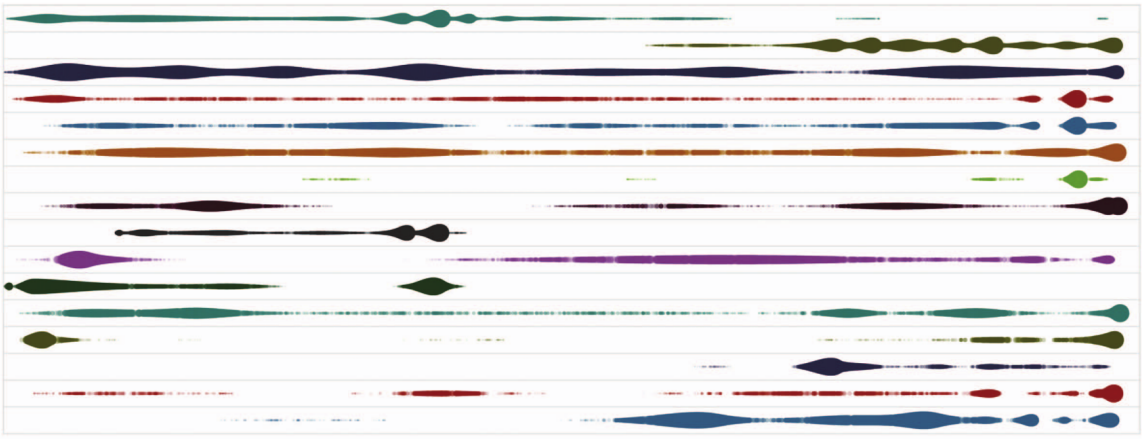


Fig. A.24 Cloudlines (Krstajic et al., 2011)

GROOVE (Lammarsch et al., 2009)

Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Time 1	SCALE	–	–	–	✓	-	SPINE	1-to-1	✓
Time 2	SCALE	–	–	–	-	✓	SPINE	1-to-1	✓
Time 3	SCALE	–	–	–	✓	-	SPINE	1-to-1	✓
Time 4	SCALE	–	–	–	-	✓	SPINE	1-to-1	✓
Attribute	SCALE	–	✓	–	-	-	-	1-to-1	-

Layers:

GROOVE			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Time 1**	–	–	–	-	$\mathbf{T_1}$	-	SCALE	SPINE
2	**Time 2**	*nesting*	–	–	-	-	$\mathbf{T_2}$	SCALE	SPINE
3	**Time 3**	*nesting*	–	–	-	$\mathbf{T_3}$	-	SCALE	SPINE
4	**Time 4**	*nesting*	**A**	–	-	-	$\mathbf{T_4}$	SCALE	SPINE

Table A.26 Signature: $N((\mathbf{T_1},L_1),N(\mathbf{T_2},L_1),N((\mathbf{T_3},L_3),(\mathbf{T_4},L_4))))$

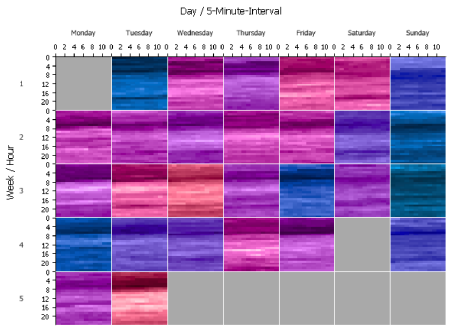


Fig. A.25 GROOVE (Lammarsch et al., 2009)

QTrade (Lei and Zhang, 2010)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Identifier	SCALE	–	–	–	-	-	POINT	1-to-1	✓
Time	SCALE	–	–	✓	-	-	POINT	1-to-1	✓
Attribute 1	SCALE	–	–	–	✓	-	-	1-to-1	-
Attribute 2	SCALE	–	–	–	-	✓	-	1-to-1	-

Layers:

Scatterplot			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Identifier**	–	**T**	–	-	**A**	**A**	SCALE	POINT

Table A.27 Signature: (**I**,L_1)

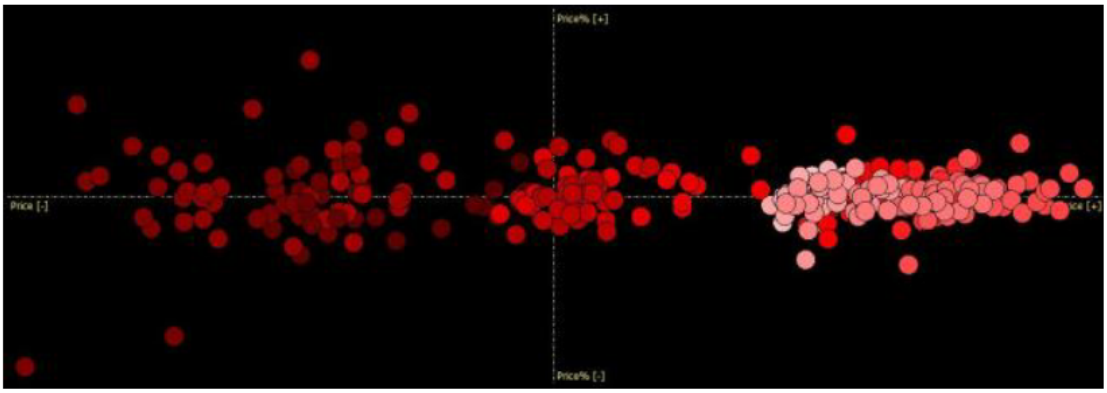


Fig. A.26 QTrade (Lei and Zhang, 2010)

Time Wave (Li and Kraak, 2013)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Time 1	SINE	–	✓	–	✓	✓	BAR	n-to-2	✓
Time 2	SINE	–	–	–	✓	✓	BAR	n-to-2	✓

Layers:

Time Wave			CARTESIAN, INSTANT-INTERVAL						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	$[\mathbf{T_1}, \mathbf{T_2}]$	–	–	**T**:duration	-	$[\mathbf{T_1}, \mathbf{T_2}]$	$[\mathbf{T_1}, \mathbf{T_2}]$	SINE	BAR

Table A.28 $N(\mathbf{T},L_1)$

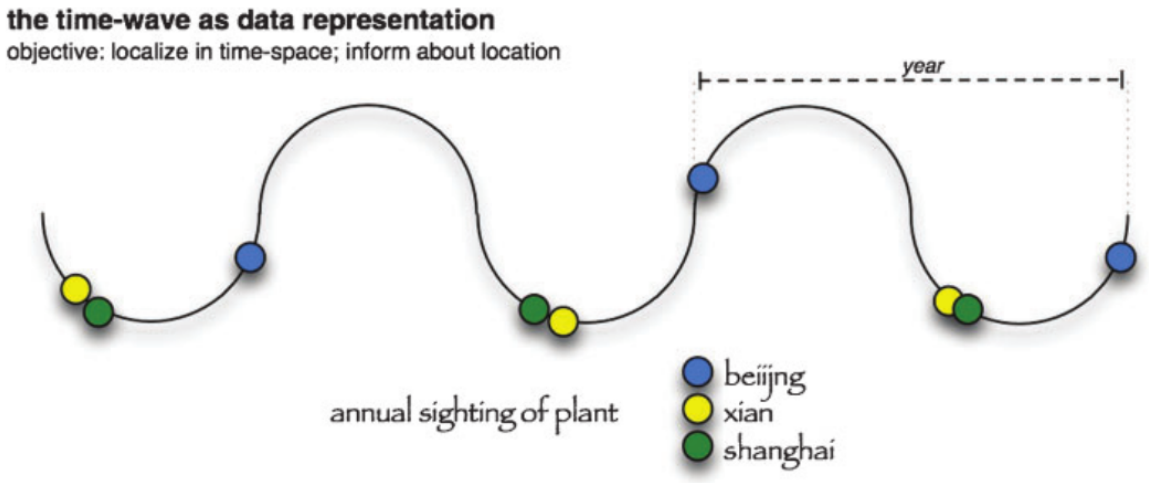


Fig. A.27 Time Wave (Li and Kraak, 2013)

LiveRAC (McLachlan et al., 2008)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Identifier 1	SCALE	–	–	–	✓	-	SPINE	1-to-1	✓
Identifier 2	SCALE	–	–	–	-	✓	SPINE	1-to-1	✓
Time	SCALE	–	–	✓	✓	-	POINT	1-to-1	✓
Attribute	SCALE	–	–	–	-	✓	-	1-to-1	-

Layers:

LiveRAC			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Identifier 1**	–	–	–	-	$\mathbf{I_1}$	-	SCALE	SPINE
2	**Identifier 2**	*nesting*	–	–	-	-	$\mathbf{I_2}$	SCALE	SPINE
3	**Time**	*nesting*	–	–	**T**	**T**	**A**	SCALE	POINT

Table A.29 Signature: $N((\mathbf{I_1},L_1),N((\mathbf{I_2},L_2),(\mathbf{T},L_3)))$

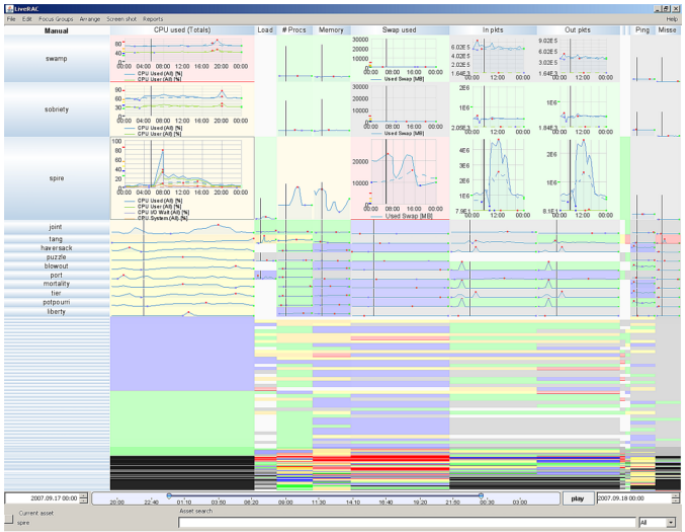


Fig. A.28 LiveRAC (McLachlan et al., 2008)

GeoTM (Qiang et al., 2012)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Identifier	SCALE	–	–	–	✓	-	BAR	1-to-1	✓
Time	SCALE	–	–	–	✓	✓	POINT	1-to-2	-

Layers:

Triangular model			CARTESIAN, INTERVAL						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Identifier**	–	–	–	-	**T**:Midpoint	**T**:Duration	SCALE	POINT

Table A.30 Signature: (**I**,L_1);

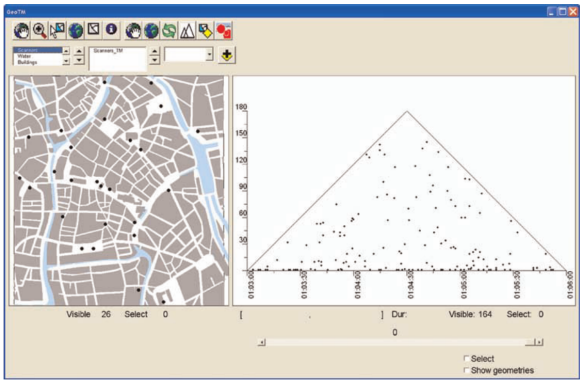


Fig. A.29 GeoTM (Qiang et al., 2012)

MobiVis (Shen and Kwan-Liu, 2008)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Time 1	SCALE	–	–	–	✓	-	SPINE	1-to-1	✓
Time 2	SCALE	–	–	–	-	✓	SPINE	1-to-1	✓
Attribute	SCALE	✓	–	–	-	-	-	1-to-1	-

Layers:

MobiVis			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Time 1**	–	–	–	-	$\mathbf{T_1}$	-	SCALE	SPINE
2	**Time 2**	*nesting*	**A**	–	-	-	$\mathbf{T_2}$	SCALE	SPINE

Table A.31 Signature: $N((\mathbf{T_1}, L_1), (\mathbf{T_2}, L_2))$

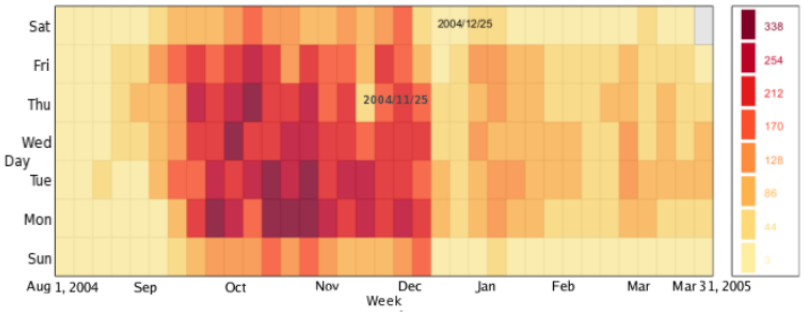


Fig. A.30 MobiVis (Shen and Kwan-Liu, 2008)

Two-tone Spiral (Tominski and Schumann, 2008)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Time 1	SPIRAL	–	–	–	✓	✓	POINT	n-to-2	✓
Time 2	SPIRAL	–	–	–	✓	✓	POINT	n-to-2	✓
Attribute	SCALE	✓	–	–	-	-	-	1-to-1	-

Layers:

Spiral			POLAR, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	$[\mathbf{T_1}, \mathbf{T_2}]$	–	**A***	–	-	$[\mathbf{T_1}, \mathbf{T_2}]$	$[\mathbf{T_1}, \mathbf{T_2}]$	SPIRAL	SPINE

Table A.32 Signature: ($[\mathbf{T_1}, \mathbf{T_2}]$,$L_1$); **A*** = Two-tone pseudo colouring

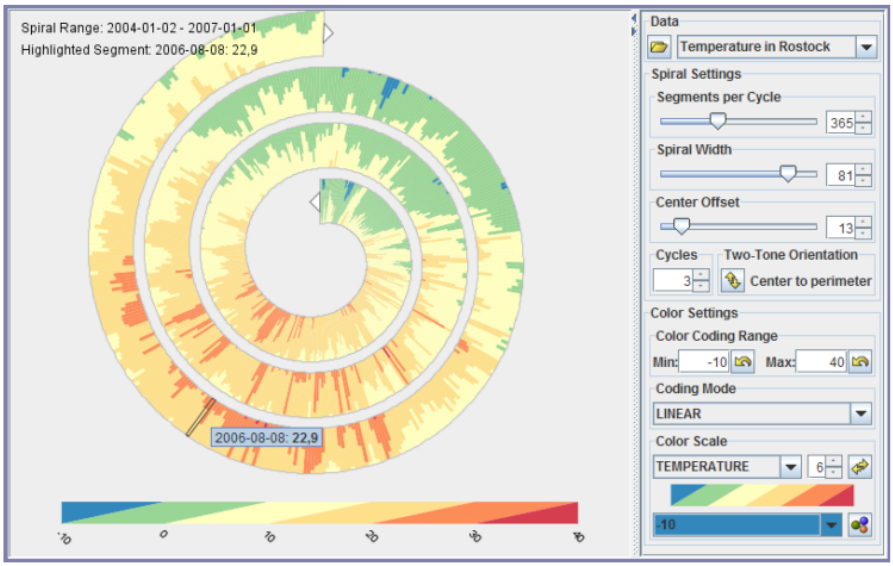


Fig. A.31 Two-tone spiral (Tominski and Schumann, 2008)

Stacking-based horizon graphs (Tominski et al., 2012)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Identifier	SCALE	–	–	–	✓	-	SPINE	1-to-1	✓
Time	SCALE	–	–	–	✓	-	SPINE	1-to-1	✓
Attribute	SCALE	✓	–	–	-	✓	-	1-to-1	-

Layers:

Stacking-based horizon graphs			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Identifier**	–	–	–	-	**I**	-	ALGORITHM	SPINE
2	**Time**	*nesting*	**A***	–	-	**T**	-	SCALE	SPINE

Table A.33 Signature: $N((\mathbf{I},L_1),(\mathbf{T},L_2))$; **A*** = Two-tone pseudo colouring

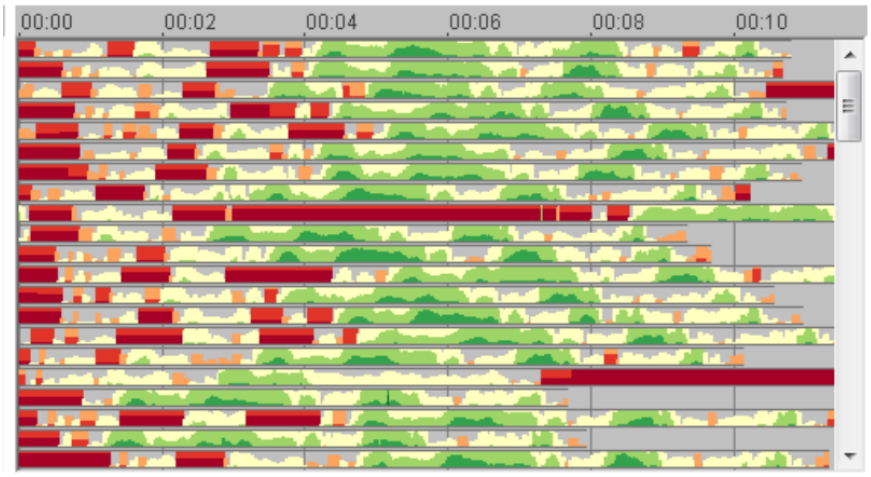


Fig. A.32 Stacking-based horizon graphs (Tominski et al., 2012)

Interval timeline (Tufte, 1983)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Identifier	SCALE	–	–	–	-	✓	SPINE	1-to-1	✓
Time	SCALE	–	✓	–	✓	-	BAR	1-to-1	-

Layers:

Interval timeline			CARTESIAN, INTERVAL						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Identifier**	–	–	–	-	-	**I**	SCALE	SPINE
2	**Time**	*nesting*	–	**T**:duration	-	**T**:start	-	SCALE	BAR

Table A.34 Signature: $N(\mathbf{I},L_1),(\mathbf{T},L_2)$

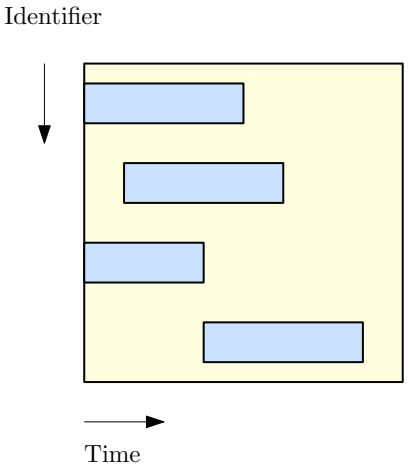


Fig. A.33 Interval timeline

Temporal summaries (Wang et al., 2009)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Time	SCALE	–	–	–	✓	-	BAR	1-to-1	✓
Attribute	SCALE	–	✓	–	-	✓	-	1-to-1	-

Layers:

Temporal summaries			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Time**	–	–	[–, **A**]	-	**T**	**A**	SCALE	BAR

Table A.35 Signature: (**T**,L_1); **A** = Item count

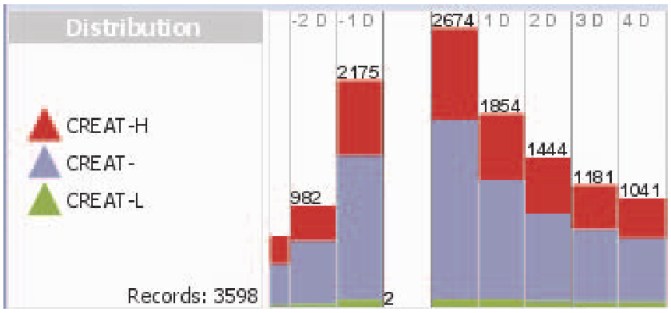


Fig. A.34 Temporal summaries (Wang et al., 2009)

Spiral 2 (Weber et al., 2001)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Time 1	SPIRAL	–	–	–	✓	✓	POINT	n-to-2	✓
Time 2	SPIRAL	–	–	–	✓	✓	POINT	n-to-2	✓
Attribute	SCALE	–	✓	–	-	-	-	1-to-1	-

Layers:

Spiral			POLAR, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	$[\mathbf{T_1}, \mathbf{T_2}]$	–	**A**	–	-	$[\mathbf{T_1}, \mathbf{T_2}]$	$[\mathbf{T_1}, \mathbf{T_2}]$	SPIRAL	POINT

Table A.36 Spiral specification, with two temporal variables being used for conditioning and position. Signature: ($[\mathbf{T_1}, \mathbf{T_2}]$,$L_1$)

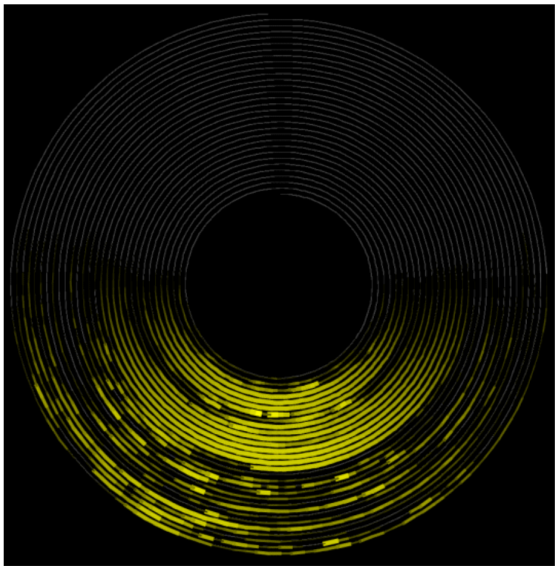


Fig. A.35 Spiral 2 (Weber et al., 2001)

Calendar cluster view (Van Wijk and Van Selow, 1999)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Time 1	SCALE	–	–	–	✓	-	SPINE	1-to-1	✓
Time 2	SCALE	–	–	–	-	✓	SPINE	1-to-1	✓
Time 3	SCALE	–	–	–	✓	-	SPINE	1-to-1	✓
Time 4	SCALE	–	–	–	-	✓	SPINE	1-to-1	✓
Attribute	SCALE	✓	–	–	-	-	-	1-to-1	-

Layers:

Calendar cluster view			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Time 1**	–	–	–	-	$\mathbf{T_1}$	-	SCALE	SPINE
2	**Time 2**	*nesting*	–	–	-	-	$\mathbf{T_2}$	SCALE	SPINE
3	**Time 3**	*nesting*	–	–	-	$\mathbf{T_3}$	-	SCALE	SPINE
4	**Time 4**	*nesting*	**A**	–	-	-	$\mathbf{T_4}$	SCALE	SPINE

Table A.37 Signature: $N((\mathbf{T_1},L_1),N(\mathbf{T_2},L_1),N((\mathbf{T_3},L_3),(\mathbf{T_4},L_4)))$; $\mathbf{A}$ = Cluster assignment

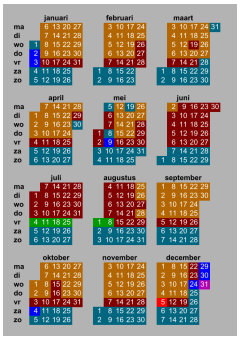


Fig. A.36 Calendar cluster view (Van Wijk and Van Selow, 1999)

Similan (Wongsuphasawat and Shneiderman, 2009)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Identifier 1	SCALE	–	–	–	-	✓	SPINE	1-to-1	✓
Time	SCALE	–	–	–	✓	-	SPINE	1-to-1	✓
Identifier 2	SCALE	–	–	–	✓	-	SPINE	1-to-1	✓
Attribute	SCALE	✓	–	–	-	-	-	1-to-1	-

Layers:

Similan			CARTESIAN, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Identifier 1**	–	–	–	-	-	$\mathbf{I_1}$	SCALE	SPINE
2	**Time**	*nesting*	–	–	-	**T**	-	SCALE	SPINE
3	**Identifier 2**	*nesting*	**A**	–	-	$\mathbf{I_2}$	-	SCALE	SPINE

Table A.38 Signature: $N((\mathbf{I_1},L_1),N((\mathbf{T},L_2),(\mathbf{I_2},L_3)))$; **A** = Existence of event

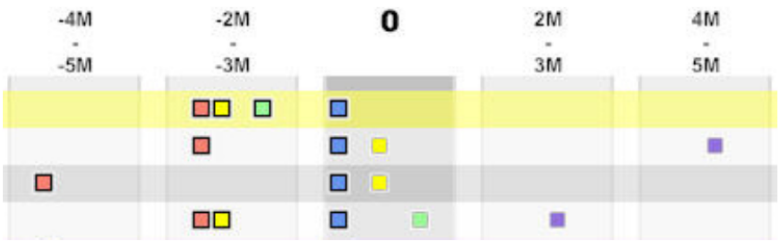


Fig. A.37 Similan (Wongsuphasawat and Shneiderman, 2009)

KronoMiner (Zhao et al., 2011)

Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Identifier	SCALE	–	–	–	✓	-	SPINE	1-to-1	✓
Time	SCALE	–	–	✓	✓	-	POINT	1-to-1	✓
Attribute	SCALE	–	–	–	-	✓	-	1-to-1	-

Layers:

KronoMiner			POLAR, INSTANT						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Identifier**	–	–	–	-	**I**	-	SCALE	SPINE
2	**Time**	*nesting*	–	–	**T**	**T**	**A**	SCALE	POINT

Table A.39 Signature: $N((\mathbf{I},L_1),(\mathbf{T},L_2))$

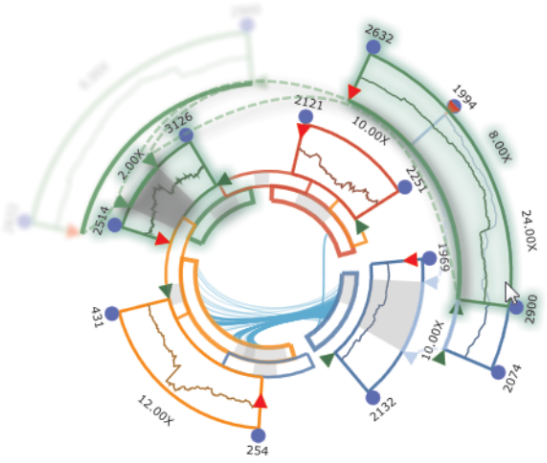


Fig. A.38 KronoMiner (Zhao et al., 2011)

TimeSlice (Zhao et al., 2012)
Mappings:

Variable	Layout	C	S	P	x/θ	y/r	Shape	Case	Cond.
Identifier	ALGORITHM	–	–	–	✓	-	SPINE	1-to-1	✓
Time	SCALE	–	✓	–	✓	-	BAR	1-to-1	✓

Layers:

TimeSlice			CARTESIAN, I NTERVAL						
L	CV	CM	C	S	P	x/θ	y/r	Layout	Shape
1	**Identifier**	-	–	–	-	**I**	-	ALG.	SPINE
2	**Time**	*nesting*	–	**T**:duration	-	**T**:start	-	SCALE	BAR

Table A.40 Signature: $N((\mathbf{I},L_1),(\mathbf{T},L_2))$

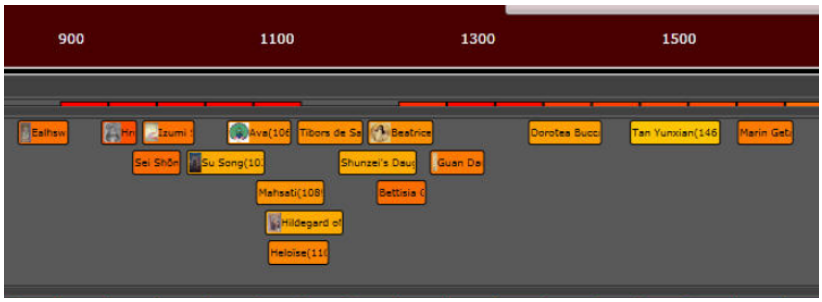


Fig. A.39 TimeSlice (Zhao et al., 2012)

Appendix B

Composition grammar

A grammar for specifying view compositions was defined to support the research. While it has not been implemented as a *parser* to generate the visualisations, it helps with the description of visualisations and the separation of the visual channels from the hierarchical structure of views. The grammar is described by the following rules using Backus normal form (BNF):

```
<visualisation> ::= <view>
                  | <composition_method> "(" < views> "," <views> ")"

<view> ::= "(" <variable> "," <composition> ")"
         | "(" <variable> "," <encoding> ")"

<composition> ::= <composition_method> "(" <encoding> <list_of_encodings>
")"
<list_of_encodings> ::= "," <encoding> <list_of_encodings> | ""
<composition_method> ::= "J" | "S" | "N"
```

`Variable` is any variable name and `encoding` is a set of visual mappings.

The above grammar covers all cases of compositions supported by the framework through the composition component, as seen in chapter 4.

- No composition: (**Time 1**, E_1)
- Same level composition: (**Time 1**, $J(E_1,E_2)$)
- Between level composition: J((**Time 1**, E_1),(**Time 2**, E_2))

Appendix C

Pseudocode descriptions of temporal transformations

In this chapter, pseudocode descriptions of the temporal transformations introduced in chapter 6 are provided to facilitate understanding of the operators and future implementation of the framework. The aim is not to describe *optimal* implementation of the transformations, as it is likely that in some programming languages, more efficient versions can be written.

C.1 Segmentation operators

Algorithm 1 Segment at instants

1: **procedure** $\mathscr{S}_t(\mathcal{T}, \{t_1, ..., t_n\})$ ▷ Segment domain $\mathcal{T}$ at instants $\{t_1, ..., t_n\}$
2: $P_{\text{tmp}} \leftarrow P_{\mathcal{T}}$ ▷ Retrieve set P of time points of $\mathcal{T}$
3: $A \leftarrow$ List-of-TimeDomain ▷ Initialise list of new time domains
4: $i \leftarrow 1$
5: **while** $i < N$ **do**
6: $P_{\text{new}} \leftarrow$ SliceArrayUpTo(P_{tmp}, t_i) ▷ Slice array of time points *lower than* t_i
7: $A_i \leftarrow$ InitDomain($\mathcal{T}, P_{\text{new}}$) ▷ Initialise a copy of time domain $\mathcal{T}$
8: $i \leftarrow i + 1$
9: **return** A ▷ Returns the list of segmented time domains

Algorithm 2 Segment by duration

1: **procedure** $\mathscr{S}_d(\mathcal{T}, d)$ ▷ Segment domain $\mathcal{T}$ by duration d
2: $N \leftarrow \text{Ceil}(E_{\mathcal{T}}/d)$ ▷ Calculate number of segments
3: $P_{\text{tmp}} \leftarrow P_{\mathcal{T}}$ ▷ Retrieve set P of time points of $\mathcal{T}$
4: $A \leftarrow$ List-of-TimeDomain ▷ Initialise list of new time domains
5: $i \leftarrow 1$
6: **while** $i < N$ **do**
7: $P_{\text{new}} \leftarrow \text{SliceArrayBy}(P_{\text{tmp}},\text{d})$ ▷ Slice array by number of elements
8: $A_i \leftarrow \text{InitDomain}(\mathcal{T}, P_{\text{new}})$ ▷ Initialise a copy of time domain $\mathcal{T}$
9: $i \leftarrow i + 1$
10: **return** A ▷ Returns the list of segmented time domains

Algorithm 3 Segment matching granularity

1: **procedure** $\mathscr{S}_G(\mathcal{S}, G)$ ▷ Segment domain $\mathcal{T}$ matching granularity G
2: $\mathcal{T}' \leftarrow \mathscr{C}_G(\mathcal{T}, G)$ ▷ Change granularity and copy time domain
3: $P_{\text{tmp}} \leftarrow P_{\mathcal{T}'}$ ▷ Retrieve set P of time points of $\mathcal{T}$
4: $A \leftarrow$ List-of-TimeDomain ▷ Initialise list of new time domains
5: **for all** p in P_{tmp} **do** ▷ Iterate over the points with new granularity
6: $A_i \leftarrow \text{Match}(P_{\mathcal{T}},\text{G}(p))$ ▷ Retrieve time points that match p_G
7: **return** A

Algorithm 4 Segment relative to

1: **procedure** $\mathscr{S}_{t,f}(\mathcal{T}, t, f)$ ▷ Segment time domain $\mathcal{T}$ relative to point t and distance function f
2: $N \leftarrow |P_{\text{tmp}}|$
3: $P_{\text{tmp}} \leftarrow P_{\mathcal{T}}$ ▷ Retrieve set P of time points of $\mathcal{T}$
4: $A \leftarrow$ List-of-TimeDomain ▷ Initialise list of new time domains
5: **while** $N > 0$ **do**
6: $size \leftarrow f(\mathcal{T}, t)$
7: $P_{new} \leftarrow \text{ArraySliceLength}(P_{\text{tmp}}, \text{size})$ ▷ Segment points of domain
8: $\text{Push}(A,\text{InitDomain}(\mathcal{T}, P_{\text{new}}))$
9: $N \leftarrow N - size$
10: **return** A ▷ Return domain

Algorithm 5 Join

1: **procedure** $\mathscr{J}([\mathcal{T}_1, [\mathcal{T}_2], \ldots])$ ▷ Join segments
2: $N \leftarrow \text{count}([\mathcal{T}_1, [\mathcal{T}_2], \ldots])$ ▷ Get the number of segments being joined
3: $i \leftarrow 1$
4: $\mathcal{T}' \leftarrow \text{InitDomain}()$ ▷ Initialise domain
5: **while** $i < N$ **do**
6: $P_{\text{tmp}} \leftarrow P_{\mathcal{T}_i}$ ▷ Retrieve time points of current segment
7: $\text{Push}(\mathcal{T}', P_{\text{tmp}})$ ▷ Add time points to new time domain
8: $i \leftarrow i + 1$
9: **return** $\mathcal{T}'$

C.2 Granularity operators

Algorithm 6 Bin at instants

1: **procedure** $\mathscr{B}_t(\mathcal{T}, \{t_1, ..., t_n\})$ ▷ Bin time points of domain $\mathcal{T}$ at instants $\{t_1, ..., t_n\}$
2: $i \leftarrow 1$
3: $P_{\text{copy}} \leftarrow P_{\mathcal{T}}$ ▷ Retrieve set P of time points of $\mathcal{T}$
4: $P_{\text{new}} \leftarrow \text{InitPoints}(\mathcal{T})$
5: **while** $i < N$ **do**
6: $p \leftarrow \text{FindPoint}(P_{\text{copy}}, t_i)$
7: $p_{new} \leftarrow \text{ArraySlice}(P_{\text{copy}}, p\ )$ ▷ Bin all time points earlier than p
8: $\text{Push}(P_{\text{new}}, p_{new})$
9: $i \leftarrow i + 1$
10: $P_{\mathcal{T}} \leftarrow P_{new}$ ▷ Replaces the old set of time points
11: **return** T ▷ Return domain

Algorithm 7 Bin by duration

procedure $\mathscr{B}_d(\mathcal{T}, d)$ ▷ Bin time points of domain $\mathcal{T}$ by duration d
  $i \leftarrow 1$
  $N \leftarrow \lceil |P|/d \rceil$
  $P_{\text{tmp}} \leftarrow P_{\mathcal{T}}$ ▷ Retrieve set P of time points of $\mathcal{T}$
  $P_{\text{new}} \leftarrow \text{InitPoints}(\mathcal{T})$
  while $i < N$ **do**
    $p_{new} \leftarrow \text{ArraySliceLength}(P_{\text{tmp}}, d)$ ▷ Bin d time points
    $\text{Push}(P_{\text{new}}, p_{new})$
    $i \leftarrow i + 1$
  $P_{\mathcal{T}} \leftarrow P_{new}$ ▷ Replaces the old set of time points
  return T ▷ Return domain

Algorithm 8 Bin relative to

procedure $\mathscr{B}_{t,f}(\mathcal{T}, t, f)$▷ Bin domain $\mathcal{T}$ relative to point t and binning function f
  $P_{\text{tmp}} \leftarrow P_{\mathcal{T}}$ ▷ Retrieve set P of time points of $\mathcal{T}$
  $P_{\text{new}} \leftarrow \text{InitPoints}(\mathcal{T})$
  while (**do**$|P_{\text{tmp}}| > 0$)
    $p_{new} \leftarrow \text{ArraySliceLength}(P_{\text{tmp}}, f(\mathcal{T}, t))$ ▷ Bin time points
    $\text{Push}(P_{\text{new}}, p_{new})$
  $P_{\mathcal{T}} \leftarrow P_{new}$ ▷ Replaces the old set of time points
  return T ▷ Return domain

Algorithm 9 Expand

procedure $\mathscr{X}(\mathcal{T})$ ▷ Reverse any binning on time domain $\mathcal{T}$
  $P_{\text{new}} \leftarrow \text{InitPoints}(G_{\mathcal{T}})$
  $P_{\text{tmp}} \leftarrow P_{\mathcal{T}}$ ▷ Retrieve set P of time points of $\mathcal{T}$
  for all p in P_{tmp} **do** ▷ Iterate over the original points
    for all t in p **do** ▷ Iterate over the instants of each point
      $p_{\text{tmp}} \leftarrow t$ ▷ Assign instant to unique time point
      $\text{Push}(P_{\text{new}}, p_{\text{tmp}})$ ▷ Add point to list of new points
  $P_{\mathcal{T}} \leftarrow P_{\text{new}}$ ▷ Assign new set of time points
  return $\mathcal{T}$ ▷ Return time domain

Algorithm 10 Change Granularity

procedure $\mathscr{C}_G(\mathcal{T}, G)$ ▷ Change granularity used to label domain $\mathcal{T}$ to G
    $P_{\text{new}} \leftarrow \text{InitPoints}(G)$
    for all p in $P_{\mathcal{T}}$ **do**
        $p_{\text{tmp}} \leftarrow G(p)$
        if p_{tmp} **not in** P_{new} **then**
            $\text{Push}(P_{\text{new}}, p_{\text{tmp}})$ ▷ Add point to list of new points
    $P_{\mathcal{T}} \leftarrow P_{\text{new}}$ ▷ Assign new set of time points
    return $\mathcal{T}$ ▷ Return time domain

Algorithm 11 Rollup

procedure $\mathscr{C}_{\text{RollUp}}(\mathcal{T})$ ▷ Rollup granularity of domain $\mathcal{T}$
    $H \leftarrow \text{CoarserThan}(G_T)$
    $P_{\text{new}} \leftarrow \text{InitPoints}(H)$
    for all p in $P_{\mathcal{T}}$ **do**
        $p_{\text{tmp}} \leftarrow H(p)$
        if p_{tmp} **not in** P_{new} **then**
            $\text{Push}(P_{\text{new}}, p_{\text{tmp}})$ ▷ Add point to list of new points
    $P_{\mathcal{T}} \leftarrow P_{\text{new}}$ ▷ Assign new set of time points
    return $\mathcal{T}$ ▷ Return time domain

Algorithm 12 Drill down

procedure $\mathscr{C}_{\text{DrillDown}}(\mathcal{T})$ ▷ Drill down granularity of domain $\mathcal{T}$
    $H \leftarrow \text{FinerThan}(G_T)$
    $P_{\text{new}} \leftarrow \text{InitPoints}(H)$
    for all p in $P_{\mathcal{T}}$ **do**
        $p_{\text{tmp}} \leftarrow H(p)$
        if p_{tmp} **not in** P_{new} **then** ▷ Prevent duplicate points included
            $\text{Push}(P_{\text{new}}, p_{\text{tmp}})$ ▷ Add point to list of new points
    $P_{\mathcal{T}} \leftarrow P_{\text{new}}$ ▷ Assign new set of time points
    return $\mathcal{T}$ ▷ Return time domain

Algorithm 13 Rotate

procedure $\mathscr{R}_t(\mathcal{T}, D)$ ▷ Rotate domain $\mathcal{T}$ in direction D
  $P_{\text{tmp}} \leftarrow P_{\mathcal{T}}$ ▷ Retrieve set P of time points of $\mathcal{T}$
  if D = RIGHT **then**
    lastElement ← Tail(P_{tmp}) ▷ Remove and retrieve last time point
    Prepend(P_{tmp},lastElement) ▷ Insert at the beginning of array
  else if D = LEFT **then**
    firstElement ← Head(P_{tmp}) ▷ Remove and retrieve first time point
    Append(P_{tmp},firstElement) ▷ Insert at end of array
  $\mathcal{T}' \leftarrow$ InitDomain(P_{tmp}) ▷ Initialise domain with time points
  return $\mathcal{T}'$ ▷ Return time domains

Algorithm 14 Align

procedure $\mathscr{A}_t(t, \{\mathcal{T}_1[, \mathcal{T}_2[, \ldots]]\})$ ▷ Align domains relative to point t
  $N \leftarrow \text{count}(\{\mathcal{T}_1[, \mathcal{T}_2[, \ldots]]\})$ ▷ Count the number of domains being aligned
  $i \leftarrow 1$
  while $i < N$ **do**
    $origin \leftarrow$FindPoint($P_{\mathcal{T}_i}, t$)
    for all p in $P_{\mathcal{T}_i}$ **do**
      $p \leftarrow \text{Dist}(p, origin)$
    $i \leftarrow i + 1$
  return $\{\mathcal{T}_1[, \mathcal{T}_2[, \ldots]]\}$ ▷ Return aligned time domains

C.3 Extent operators

Algorithm 15 Trim At

procedure $\mathscr{T}_t(\mathcal{T}, [t_1], [t_2])$ ▷ Segment domain $\mathcal{T}$ at instants $\{t_1, ..., t_n\}$
  $p_1 \leftarrow \text{FindPoint}(P_{\mathcal{T}}, t_1)$
  $p_2 \leftarrow \text{FindPoint}(P_{\mathcal{T}}, t_2)$
  for all p in $P_{\mathcal{T}}$ **do**
    if EarlierThan(p, p_1) OR LaterThan(p, p_2) **then**
      Remove(p,$P_{\mathcal{T}}$)
  return $\mathcal{T}$ ▷ Return time domain

Algorithm 16 Trim By

procedure $\mathscr{T}_d(\mathcal{T}, [d_1], [d_2])$ ▷ Segment domain $\mathcal{T}$ at instants $\{t_1, ..., t_n\}$
  $i \leftarrow 1$
  while $i < d_1$ **do**
    Head($P_\mathcal{T}$)
    $i \leftarrow i + 1$
  $i \leftarrow 1$
  while $i < d_2$ **do**
    Tail($P_\mathcal{T}$)
    $i \leftarrow i + 1$
  return $\mathcal{T}$ ▷ Returns time domain

Algorithm 17 Extend To

procedure $\mathscr{E}_t(\mathcal{T}, [t_1], [t_2])$ ▷ Segment domain $\mathcal{T}$ at points $\{t_1, ..., t_n\}$
  while $s_\mathcal{T} > t_1$ **do**
    $p \leftarrow G_\mathcal{T}(t_1 - 1)$
    $s_\mathcal{T} \leftarrow p$
    Prepend($P_\mathcal{T}$,p)
  while $e_\mathcal{T} < t_2$ **do**
    $p \leftarrow G_\mathcal{T}(t_2 - 1)$
    $e_\mathcal{T} \leftarrow p$
    Append($P_\mathcal{T}$,p)
  return $\mathcal{T}$ ▷ Return time domain

Algorithm 18 Extend By

1: **procedure** $\mathscr{E}_d(\mathcal{T}, [d_1], [d_2])$ ▷ Segment domain $\mathcal{T}$ at points $\{t_1, ..., t_n\}$
2: $i \leftarrow 1$
3: **while** $i < d_1$ **do**
4: $p \leftarrow G_{\mathcal{T}}(p_{1_{\mathcal{T}}} - 1)$
5: $s_{\mathcal{T}} \leftarrow p$
6: Prepend($P_{\mathcal{T}}$,p)
7: $i \leftarrow i - 1$
8: $i \leftarrow 1$
9: **while** $i < d_2$ **do**
10: $p \leftarrow G_{\mathcal{T}}(t_2 - 1)$
11: $e_{\mathcal{T}} \leftarrow p$
12: Append($P_{\mathcal{T}}$,p)
13: $i \leftarrow i + 1$
14: **return** $\mathcal{T}$ ▷ Return time domain

C.4 Utility functions

Auxiliary functions were included in the previous sections to support the operators and enable concise descriptions of them. These functions can also be considered as an essential set of basic operations that are required to implement the temporal transformations, with the manipulation of basic data types that are used in the abstractions of the temporal properties, which are the targets of the transformations.

InitDomain(...) Initialise an abstraction of the temporal domain with the passed parameters. The assumption is that the parameters that were not passed are inferred. An example is when a set of time points is passed as a parameter and the extent and starting and ending points are extracted from it.

SliceArrayUpTo(Array,Point) Copy and remove elements from array from the start *up to* the passed parameter.

SliceArrayBy(Array,Count) Copy and remove n elements from array, where n is passed as a parameter.

Append(Array,Element) Add new element to the end of array.

Prepend(Array,Element) Add new element to the beginning of array.

Head(Array) Copy and remove first element of array.

Tail(Array) Copy and remove last element of array.

Remove(Array,Point) Remove parameter from array.

InitPoints() Initialise a new set of time points.

FindPoint(Points,Instant) Retrieve time point containing the instant that is passed as parameter.

Dist(Point,Point) Calculate temporal distance between two time points.

Match(Points,Point) Return all time points that are equal to the reference.

CoarserThanGranularity Returns granularity *immediately* coarser than the parameter.

FinerThan(Granularity) Returns granularity *immediately* finer than the parameter.

EarlierThan(Reference,Point) Returns true if point happens *earlier* than reference.

LaterThan(Reference,Point) Returns true if point happens *later* than reference.

Appendix D

Survey of interactions in time-oriented visualisations

This appendix contains the textual descriptions of the interactions surveyed as part of the development Transformation component. The taxonomy of interactions by Yi et al. (2007) was used as a means to classify the interactions and later map them to the operators described in chapter 6. The survey also includes interactions that are not part of the final set of transformations and are listed here to demonstrate the rigour of the process.

Reference	S	Ex	R	En	A/E
André et al. (2007)		✓			✓
Bach et al. (2016b)			✓		✓
Bale et al. (2007)		✓			✓
Boyandin et al. (2011)	✓	✓	✓		✓
Burch et al. (2008)					✓
Carlis and Konstan (1998)		✓	✓		
Harrison et al. (1994)					✓
Keim et al. (2004)			✓	✓	✓
Krstajic et al. (2011)		✓			
Lammarsch et al. (2009)			✓	✓	✓
Van Wijk and Van Selow (1999)	✓	✓			
Zhao et al. (2011)			✓		✓
Hochheiser and Shneiderman (2004)					
Li and Kraak (2013)	✓				
Qiang et al. (2012)	✓				
Wongsuphasawat and Shneiderman (2009)			✓		
Gad et al. (2015)		✓	✓		
Wang et al. (2009)			✓		✓
Gschwandtner et al. (2011)		✓	✓		
Tominski and Schumann (2008)			✓		
McLachlan et al. (2008)				✓	✓
Tominski et al. (2012)		✓	✓		✓
Sips et al. (2012)					✓
Ferreira et al. (2013)			✓		✓
Shahar et al. (2006)			✓		✓
Loorak et al. (2016)					✓
Shen and Kwan-Liu (2008)	✓				✓
Javed and Elmqvist (2013)					✓
Kincaid (2010)		✓			
Andrienko and Andrienko (2012)					✓
Cho et al. (2014)			✓		
Kothur et al. (2013)					✓
Beard et al. (2008)					✓
Tableau Software (2018)			✓		
van der Corput and van Wijk (2017)					
Guerra-Gomez et al. (2013)					✓
Cousins and Kahn (1991)			✓		✓
TIBCO Software Inc. (2018)					✓

Table D.1 Summary of first step of interaction survey, part 1. **S** = Select, **E** = Explore, **R** = Reconfigure, Encode, **A/E** = Abstract/elaborate

Reference	F	C/R	VC	SC	AC	DT
André et al. (2007)			✓			✓
Bach et al. (2016b)						
Bale et al. (2007)	✓	✓				
Boyandin et al. (2011)	✓					
Burch et al. (2008)						
Carlis and Konstan (1998)	✓		✓	✓		
Harrison et al. (1994)	✓					
Keim et al. (2004)						
Krstajic et al. (2011)						
Lammarsch et al. (2009)						
Van Wijk and Van Selow (1999)	✓	✓			✓	
Zhao et al. (2011)	✓					
Hochheiser and Shneiderman (2004)	✓					
Li and Kraak (2013)	✓					
Qiang et al. (2012)						
Wongsuphasawat and Shneiderman (2009)			✓			✓
Gad et al. (2015)		✓				✓
Wang et al. (2009)	✓	✓				
Gschwandtner et al. (2011)	✓	✓				
Tominski and Schumann (2008)						
McLachlan et al. (2008)		✓				
Tominski et al. (2012)	✓					✓
Sips et al. (2012)						
Ferreira et al. (2013)						✓
Shahar et al. (2006)						
Loorak et al. (2016)						
Shen and Kwan-Liu (2008)						
Javed and Elmqvist (2013)	✓					
Kincaid (2010)						
Andrienko and Andrienko (2012)						
Cho et al. (2014)						
Kothur et al. (2013)						
Beard et al. (2008)						
Tableau Software (2018)						
van der Corput and van Wijk (2017)			✓			
Guerra-Gomez et al. (2013)						
Cousins and Kahn (1991)						
TIBCO Software Inc. (2018)						

Table D.2 Summary of first step of interaction survey, part 2. **F** = Filter, **C/R** = Connect/Relate, **VC** = Visualisation configuration, **SC** = State control, **AC** = Analytical configuration, **DT** = Design transformation

André et al. (2007)

- **Select**:
- **Explore**: Moving window on temporal histogram, Scrolling vertically/horizontally on main screen
- **Reconfigure**:
- **Encode**:
- **Abstract/elaborate**: Compress/expand moving window, Sliders to change size of items on main view based on "prominence"
- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**: Temporal histogram changes according to window size
- **State control**:
- **Analytical configuration**:
- **Design transformation**: Temporal binning

Bach et al. (2016b)

- **Select**:
- **Explore**:
- **Reconfigure**: Slider for changing positioning weight between 1D timeline and timecurve
- **Encode**:
- **Abstract/elaborate**: Tooltip showing details
- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Bale et al. (2007)

- **Select**:
- **Explore**: Zoom in segment of interest
- **Reconfigure**:
- **Encode**:
- **Abstract/elaborate**: Tooltip showing details

- **Filter**: Can select angular sections, Angular and radial markers to show in line graph
- **Connect/relate**: Line graph shows data selected by marker
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Boyandin et al. (2011)

- **Select**: Select origins/destinations
- **Explore**: Panning and zooming in the three different views
- **Reconfigure**: Reorder rows
- **Encode**:
- **Abstract/elaborate**: Aggregate/disaggregate rows by origin or destination
- **Filter**: Filter origins/destinations
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Burch et al. (2008)

- **Select**:
- **Explore**:
- **Reconfigure**:
- **Encode**:
- **Abstract/elaborate**: Collapse/expand nodes in hierarchy, Tooltip showing details
- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Carlis and Konstan (1998)

- **Select**:

- **Explore**: Panning, zooming, rotating and tilting (includes 3D)
- **Reconfigure**: Change number of points per spiral arm
- **Encode**:
- **Abstract/elaborate**:
- **Filter**: Change spiral start and end point, Change time start and end point
- **Connect/relate**:
- **Visualisation configuration**: Interface with settings
- **State control**: Save and load view configuration
- **Analytical configuration**:
- **Design transformation**:

Harrison et al. (1994)

- **Select**:
- **Explore**:
- **Reconfigure**:
- **Encode**:
- **Abstract/elaborate**: Context and zoom
- **Filter**: Zoom selection
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Keim et al. (2004)

- **Select**:
- **Explore**:
- **Reconfigure**: Change time direction
- **Encode**: Change radius of segments
- **Abstract/elaborate**: Change time unit in segment
- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Krstajic et al. (2011)

- **Select**:
- **Explore**: Special fisheye lens with linear or exponential magnification of time points (on screen space)
- **Reconfigure**:
- **Encode**:
- **Abstract/elaborate**:
- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Lammarsch et al. (2009)

- **Select**:
- **Explore**:
- **Reconfigure**: Change order of time units
- **Encode**: Change various color overlays
- **Abstract/elaborate**: Change time units
- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Van Wijk and Van Selow (1999)

- **Select**: Select members of clusters
- **Explore**: Geometric zoom in chart
- **Reconfigure**:
- **Encode**:
- **Abstract/elaborate**:
- **Filter**: Temporal range filter (cut)
- **Connect/relate**: Show similar days
- **Visualisation configuration**:

- **State control**:
- **Analytical configuration**: Change number of clusters, Generate cluster from smoothed data
- **Design transformation**:

Zhao et al. (2011)

- **Select**:
- **Explore**:
- **Reconfigure**: Re-position segments on the circle
- **Encode**:
- **Abstract/elaborate**: Add new segments from MQW
- **Filter**: Movable query widget
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Hochheiser and Shneiderman (2004)

- **Select**:
- **Explore**:
- **Reconfigure**:
- **Encode**:
- **Abstract/elaborate**:
- **Filter**: Movable query widget
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Li and Kraak (2013)

- **Select**: Movable marker to highlight filtered points
- **Explore**:
- **Reconfigure**:
- **Encode**:
- **Abstract/elaborate**:

- **Filter**: Movable selection
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Qiang et al. (2012)

- **Select**: Movable query shapes to highlight filtered points, Instant query shape from menu
- **Explore**:
- **Reconfigure**:
- **Encode**:
- **Abstract/elaborate**:
- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Wongsuphasawat and Shneiderman (2009)

- **Select**:
- **Explore**:
- **Reconfigure**: Select central event to transform to relative scale
- **Encode**:
- **Abstract/elaborate**:
- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**: Temporal histogram changes according to window size
- **State control**:
- **Analytical configuration**:
- **Design transformation**: Temporal binning

Gad et al. (2015)

- **Select**:

- **Explore**: Panning and zooming
- **Reconfigure**: Recalculate positions with selected trendline on top
- **Encode**:
- **Abstract/elaborate**:
- **Filter**:
- **Connect/relate**: Highlight related timelines when hovering
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**: Timeline segmentation, Segments of time are aggregated to single point

Wang et al. (2009)

- **Select**:
- **Explore**:
- **Reconfigure**: Select event to transform to relative scale
- **Encode**:
- **Abstract/elaborate**: Time unit changes reactively with zooming out, Change time units
- **Filter**: Movable query widget
- **Connect/relate**: Highlight filtered events
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Gschwandtner et al. (2011)

- **Select**:
- **Explore**: Panning and zooming
- **Reconfigure**: Transform to relative scale
- **Encode**:
- **Abstract/elaborate**:
- **Filter**: Movable selection widget
- **Connect/relate**: Highlight related events
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:

- **Design transformation**:

Tominski and Schumann (2008)

- **Select**:
- **Explore**:
- **Reconfigure**: Change number of points per spiral arm, change number of arms
- **Encode**:
- **Abstract/elaborate**:
- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

McLachlan et al. (2008)

- **Select**:
- **Explore**:
- **Reconfigure**:
- **Encode**: Change height and width of cell, Show different encoding depending on cell size
- **Abstract/elaborate**: Extend or cut timeline
- **Filter**:
- **Connect/relate**: Highlight same time in different charts
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Tominski et al. (2012)

- **Select**:
- **Explore**: Panning and zooming
- **Reconfigure**: Transform to relative scale
- **Encode**:
- **Abstract/elaborate**: Change time unit
- **Filter**: Movable query widget for special lens
- **Connect/relate**:

- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**: Temporal binning for time lens

Sips et al. (2012)

- **Select**:
- **Explore**:
- **Reconfigure**:
- **Encode**:
- **Abstract/elaborate**: Extend or cut timeline
- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Ferreira et al. (2013)

- **Select**:
- **Explore**:
- **Reconfigure**: RESCALE
- **Encode**:
- **Abstract/elaborate**: CUT, EXTEND
- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**: Temporal binning in “time series" histogram

Shahar et al. (2006)

- **Select**:
- **Explore**:
- **Reconfigure**: Aligning timelines
- **Encode**:
- **Abstract/elaborate**: Change granularity, modify the extent of timelines

- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Loorak et al. (2016)

- **Select**:
- **Explore**:
- **Reconfigure**:
- **Encode**:
- **Abstract/elaborate**: Modify the extent of timeline
- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Shen and Kwan-Liu (2008)

- **Select**: Draggable time window for filtering
- **Explore**:
- **Reconfigure**:
- **Encode**:
- **Abstract/elaborate**: Change time units
- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Javed and Elmqvist (2013)

- **Select**:
- **Explore**:
- **Reconfigure**:

- **Encode**:
- **Abstract/elaborate**: Extend or cut selection
- **Filter**: Movable window to choose segment
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Kincaid (2010)

- **Select**:
- **Explore**: Draggable lens
- **Reconfigure**:
- **Encode**:
- **Abstract/elaborate**:
- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Andrienko et al. (2011)

- **Select**:
- **Explore**:
- **Reconfigure**:
- **Encode**:
- **Abstract/elaborate**: Change aggregation through granularities
- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Cho et al. (2014)

- **Select**:

- **Explore**:
- **Reconfigure**: Select event to transform to relative scale
- **Encode**:
- **Abstract/elaborate**:
- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Kothur et al. (2013)

- **Select**:
- **Explore**:
- **Reconfigure**:
- **Encode**:
- **Abstract/elaborate**: Cut or extend periodicty explorer selecting start/end dates or duration+start date
- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Beard et al. (2008)

- **Select**:
- **Explore**:
- **Reconfigure**:
- **Encode**:
- **Abstract/elaborate**: Assign different time units to stacks/panels/bands
- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Tableau Software (2018)

- **Select**:
- **Explore**:
- **Reconfigure**: Change order of weekdays
- **Encode**:
- **Abstract/elaborate**: Change granularity
- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

van der Corput and van Wijk (2017)

- **Select**:
- **Explore**:
- **Reconfigure**:
- **Encode**:
- **Abstract/elaborate**:
- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**: Temporal histogram changes according to window size
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Guerra-Gomez et al. (2013)

- **Select**:
- **Explore**:
- **Reconfigure**:
- **Encode**:
- **Abstract/elaborate**: Change the granularity in the time box
- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**:

- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Cousins and Kahn (1991)

- **Select**:
- **Explore**:
- **Reconfigure**: Align time lines
- **Encode**:
- **Abstract/elaborate**: Change granularity of time line
- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

TIBCO Software Inc. (2018)

- **Select**:
- **Explore**:
- **Reconfigure**:
- **Encode**:
- **Abstract/elaborate**: Change granularity
- **Filter**:
- **Connect/relate**:
- **Visualisation configuration**:
- **State control**:
- **Analytical configuration**:
- **Design transformation**:

Appendix E

Definitions

Some interactive visualisation terms and concepts used throughout this thesis have various definitions in the literature and their interpretation may, at times, be ambiguous. This appendix serves as a reference for the vision of interactive visualisation in this thesis to facilitate understanding, summarised in fig. E.1.

In the thesis, a **view** or **visual encoding** is a *single* graphic or chart; one **visualisation** refers to a combination of one or more views. The methods and rules that are used to combine the views are referred to as **composition methods** – these can be, as discussed in chapter 2, for example, juxtaposing views.

A view is the result of interpreting and rendering a set of **visual mappings**; these are the association of variables in the dataset with *visual variables*, which are properties of elements drawn in charts, such as height and width of rectangles or position on the screen.

As outlined in the beginning of the thesis, exploratory analysis is a motivation for this research: this involves the use of varying arrangements of composition, visual mappings when analysing data and manipulating the underlying data. These variations are described as **transformations** in the thesis and are considered as internal actions in a system implementing the visualisations. The internal actions are activated by **interactions**, which are external actions, such as using a mouse to click on an object on the screen, made by a user.

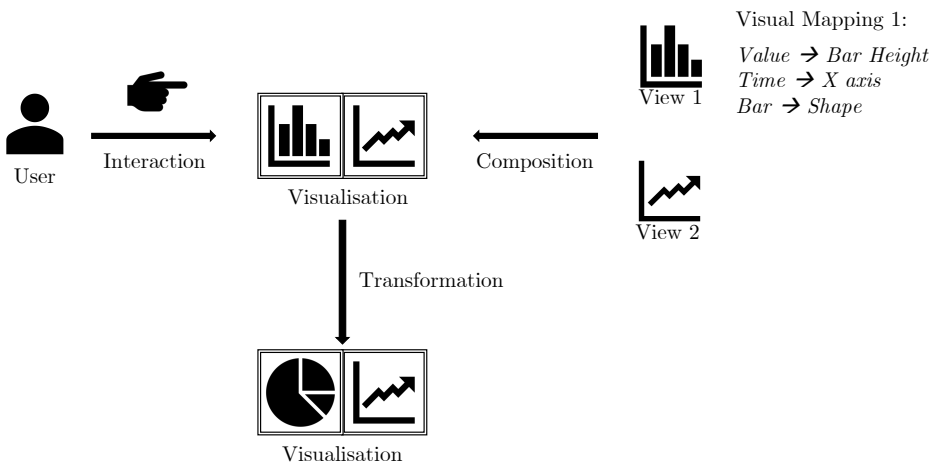


Fig. E.1 Summary of the concepts used in the thesis and how they relate to each other.