Visualisation of Long in Time Dynamic Networks on Large Touch Displays

Alexandra Lee
[redacted]

Submitted to Swansea University in fulfilment
of the requirements for the Degree of Doctor of Philosophy

Swansea University
Prifysgol Abertawe

Department of Computer Science
Swansea University

October 14, 2021

Declaration

This work has not been previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed [redacted]

Date 23/11/2021

Statement 1

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed [redacted]

Date 23/11/2021

Statement 2

I hereby give my consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed [redacted]

Date 23/11/2021

Abstract

Any dataset containing information about relationships between entities can be modelled as a network. This network can be static, where the entities/relationships do not change over time, or dynamic, where the entities/relationships change over time. Network data that changes over time, dynamic network data, is a powerful resource when studying many important phenomena, across wide-ranging fields from travel networks to epidemiology.

However, it is very difficult to analyse this data, especially if it covers a long period of time (e.g, one month) with respect to its temporal resolution (e.g. seconds). In this thesis, we address the problem of visualising long in time dynamic networks: networks that may not be particularly large in terms of the number of entities or relationships, but are long in terms of the length of time they cover when compared to their temporal resolution.

We first introduce Dynamic Network Plaid, a system for the visualisation and analysis of long in time dynamic networks. We design and build for an 84" touch-screen vertically-mounted display as existing work reports positive results for the use of these in a visualisation context, and that they are useful for collaboration. The Plaid integrates multiple views and we prioritise the visualisation of interaction provenance. In this system we also introduce a novel method of time exploration called 'interactive timeslicing'. This allows the selection and comparison of points that are far apart in time, a feature not offered by existing visualisation systems. The Plaid is validated through an expert user evaluation with three public health researchers.

To confirm observations of the expert user evaluation, we then carry out a formal laboratory study with a large touch-screen display to verify our novel method of time navigation against existing animation and small multiples approaches. From this study, we find that interactive timeslicing outperforms animation and small multiples for complex tasks requiring a comparison between multiple points that are far apart in time. We also find that small multiples is best suited to comparisons of multiple sequential points in time across a time interval.

To generalise the results of this experiment, we later run a second formal laboratory study

in the same format as the first, but this time using standard-sized displays with indirect mouse input. The second study reaffirms the results of the first, showing that our novel method of time navigation can facilitate the visual comparison of points that are distant in time in a way that existing approaches, small multiples and animation, cannot. The study demonstrates that our previous results generalise across display size and interaction type (touch vs mouse).

In this thesis we introduce novel representations and time interaction techniques to improve the visualisation of long in time dynamic networks, and experimentally show that our novel method of time interaction outperforms other popular methods for some task types.

Acknowledgements

Throughout the writing of this dissertation I have received a great deal of support and assistance. I would like to express my sincere gratitude to Dr Archambault for introducing me to the field of network visualisation and being willing to be my supervisor for the past 4 years. It has not always been easy job! I am deeply grateful to Dr Nacenta who, through his valuable collaboration, helped to guide and shape my outlook of the human-centred aspect of this work.

I also owe a great deal to Mike, Joss, Gavin, Ken, and David for welcoming me into their research lab and always being willing to act as test subjects and sounding boards, even though human-centred visualisation research is quite far from their field of machine learning. I'd particularly like to thank Mike for always being happy to correct bibliographies close to a deadline.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Contents

A graph consists of a set of nodes and edges. The entities in the graph (companies, people, cities...) are the nodes of the graph while the relationships between those entities (transactions, friendships, direct flights...) are the edges of the graph. Any type of dataset containing information about connections can be modelled and visualised as a graph, including relationships between companies [33], the spread of communicable disease [204], social networks [59], communication systems [120], and travel networks [114], as well as in fields across the biological [142] and physical sciences [45].

There are two main types of graphs: static graphs, where relationships and nodes do not evolve, and dynamic graphs, where there is some change in the graph structure or composition over time. The time component of a dynamic graph can be continuous, where graph events [229] are recorded with the exact timestamp of the event occurrence, or discrete, where time is binned into constant intervals and the graph structure is only recorded at the start or end of each uniformly sampled period.

Regardless of the inclusion of a temporal component, an essential tool for understanding a graph is visualisation. Chen et al. [73] use Anscombe's quartet [14] to demonstrate that networks can have the same numerical properties but radically different structures in terms of connectivity, planarity, and symmetry (shown in Figure 1.1). If multiple networks can have the same properties while being structurally distinct, it is therefore vital that graph visualisation is a component of graph analysis, as it will be impossible to tell the difference between numerically identical graphs without visualisation. For this reason, visualisation is an essential early step for the exploratory data analysis [307] of network data.

The visualisation of static network data can be problematic if the dataset is large or highly connected. Adding a temporal dimension dramatically increases the complexity of the visualisation of network-type data. Where there are large numbers of interactions between elements of the dataset this translates to a large number of edges in the representation, resulting in the all too frequently encountered 'hairball graph', where edge density is such that there is no good graph layout and there can easily be more edges to draw than pixels available on the screen. The 'hairball graph' problem is present in both static and dynamic graph representations but is compounded by the inclusion of the time dimension, particularly when a dynamic graph spans a long a period of time (e.g. months) with respect to a very small temporal resolution (e.g. seconds).

Given that temporality is a key component of dynamic network data, without the time component the data is simply a static network, it follows that it is an essential dimension for

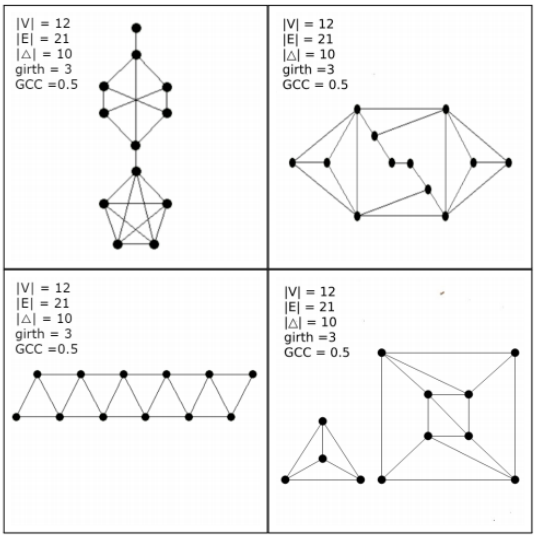


Figure 1.1: Chen et al. [73] demonstrate how four graphs with the same 5 common properties – |V| = 12, |E| = 21, number of triangles | 4 | = 10, girth = 3 and global clustering coefficient GCC= 0.5 – have distinct graph structures.

the visualisation of this type of data. By including temporal information in our visualisations we can uncover network evolution events [87, 125], and potentially unpick the order of earlier events that lead to a chosen network state. In some cases, understanding the order of events in a network can be key to not only understanding the network as a whole, but can also provide insight into ways in which networks can be manipulated at key stages to give a desired result [23, 55].

Existing methods of dynamic graph visualisation mainly opt to encode time either in the time dimension, such as in animation, or in the space dimension by mapping the time value to some physical space on the screen or page, which is the method used by small multiples. As scalability, in many contexts, is a significant issue in the visualisation of dynamic graphs, each of these approaches attempt to reduce the graph in at least one dimension as a mitigation strategy. Previous work finds that small multiples nearly always outperforms animation in terms of both time and correctness when tasks relating to graph analysis are involved [18, 21, 102]. This result is partly due to the very high memory cost of animation – only one state of the graph is shown on the screen at each point. Ergo to compare between two states, the user must memorise the graph at at least one point.

However, there are legitimate use cases for animated approaches – they are compelling for communicating high-level information about the graph [255]. Archambault et al. [15, 19]

find that animation can help where the mental map cannot be preserved, and that animation is best for detecting the appearance or disappearance of nodes and edges. While small multiples are the more usable option for analysis there are severe scalability problems when trying to examine dynamic graphs that are large in any dimension; the primary constraints for small multiples are the available screen size and the interaction cost of the approach if all timeslices cannot be simultaneously displayed on the screen. For the comparison of time points that are far apart in time, and therefore far apart in the representation, there is also a costly memory requirement that reflects animated approaches. These limitations are particularly true for long in time dynamic graphs.

From this knowledge, we form the central tenet of the thesis:

- *What are useful representations and interaction techniques that can improve the visualisation of long in time dynamic networks?*

This thesis explores systems and methods to facilitate the visual analysis of dynamic graph data that is long in the temporal dimension, but not necessarily large in terms of the size of vertex or edge sets. In particular, we address the following:

- Exploring the design space of techniques to visualise long in time dynamic networks.
- Designing a new tool for the exploration of long in time dynamic graphs in a public health context with a number of novel features.
- Testing this new tool in two evaluation sessions with expert researchers.
- Empirically testing common existing time interaction methods against our novel method of time selection on large, touch screen, displays.
- Empirically testing existing time interaction methods against our novel method on personal displays with mouse input.

The focus of the thesis is primarily on the visualisation of long in time networks, with little consideration offered to performance in terms of computing power or speed - we decided only to optimise the performance of our novel approach after the results of the empirical evaluations. We also do not work with particularly large graphs (over 1 million edges); developing these algorithms is far beyond this thesis's scope. For the physical display of our system, we focus primarily on supporting large (greater than 30 inches) wall-mounted touch-screen displays. We make this choice because a body of work [10, 32, 86, 160, 301, 302, 325] finds that

physical navigation and embodied interaction techniques for large displays offer performance benefits for the majority of users and that the limitations of visualisation begin to shift away from the display technology and towards the perceptual ability of the user. We also carry out an empirical evaluation on a series of small (sub 30 inches) standard displays to ensure that our novel time navigation method is generalisable to smaller screens and mouse input. Our rationale for this is that these smaller screens are typically available to a broader range of users than specialist large touch-screen displays.

1.1 Thesis Structure

The rest of the thesis is structured as follows. Chapter 2 details the existing work for the visualisation of dynamic graph data, and also examines existing visualisation and experimental work that uses large wall displays. We also discuss of graph tasks and their relative importance when designing and building visualisations.

Chapter 3 exhibits our dynamic graph visualisation system – referred to in shorthand as 'The Plaid'. Chapter 4 details the two evaluation sessions that we carried out with the target users of the interface described in Chapter 3. The work presented in Chapters 3 and 4 is based off work published at ACM SIGCHI 2019 [188].

Chapter 5 is a formal laboratory study that evaluates a number of Plaid-like interfaces to evaluate the efficacy and efficiency of a new interactive timeslicing approach first trialled with The Plaid. This approach is compared experimentally with the existing, common, dynamic graph visualisation techniques of small multiples and animation. The work presented in Chapter 5 is based off work published in IEEE VIS 2020 [189].

Chapter 6 is a second formal laboratory study that evaluates the same set of interfaces as Chapter 5 but in a remote setting. In this chapter we carry out the experiment in a fully remote manner using a standard-sized display with mouse pointer input. This is unlike Chapter 5 where the study was carried out with a large screen wall display and direct touch input.

Chapter 7 discusses the new findings from the previous chapters and positions them within the wider visualisation field. The implications of our research are also considered. In addition, we make recommendations for future work based on the outcomes of the thesis.

Chapter 8 contains concluding remarks and summing up of the thesis contributions, situated in the context of the wider graph visualisation field.

1.2 Thesis Contributions

This thesis makes the following contributions:

- **A new method for the interactive selection of timeslices for long in time graphs.** We propose a new interaction method for the exploration of the temporal component of a dynamic graph by giving the end-user the ability to select and compare multiple time periods of interest. In this thesis we do not introduce a novel representation of the data, but instead focus on an interaction technique that can be used to improve the utility of existing visualisation methods.

- **A new method for interaction provenance visualisation.** We present a system that leverages the strengths of a large touch-screen display to facilitate the consistent visualisation of provenance information. We also provide support for analysis cascades alongside this, where analysis can be repeated on an area of the network that is distant in the time dimension from the existing analysis. We validate this via observation sessions of target users while they carry out an analysis with our new system. Here, again, the primary contribution is an interaction technique, however we also make a small contribution to the area of the representation of interaction provenance information.

- **Empirically testing multiple basic alternative time interaction methods against our new method for time exploration through a controlled experiment.** We carry out a formal experiment to assess the efficiency and effectiveness of our new interactive time selection method on large vertically-mounted displays with direct touch input. We compare our new method to the field standards of small multiples and interactive animation using a filtered real-world dataset and build our task framework around a variety of actual graph tasks. With this, we extend the validity our first contribution to useful interaction techniques for the visualisation of long-in-time dynamic network data.

- **Empirically testing existing time exploration methods against our new method on personal displays with mouse input to ensure the generalisability of the technique.** We repeat the previous formal experiment but modify all interfaces to operate on a desktop display with full HD resolution and to accept mouse pointer input rather than direct touch. We discuss the implications of these results on the generalisability of our new method and further extend the validity of our contribution to interaction techniques that facilitate the better exploration of long-in-time dynamic network data.

Chapter 2

Background and Related Work

Contents

In this chapter, we first introduce and define common terms that will be used in the thesis. Due to the importance of designing visualisations for tasks [215], we then examine task taxonomies for the visualisation of network data. Next, we look at existing visualisation approaches for dynamic network data and discuss the results of existing experiments testing some of these methods. Finally, we discuss existing visualisation approaches for large displays, the impacts of different interaction techniques, and the utility of interaction provenance.

2.1 Common Definitions

This section defines key terms used in the thesis.

Static Graph A graph can be minimally described as an ordered pair $G = (V, E)$ comprising of V a finite set of vertices or nodes, and E a set of edges $E \subseteq V \times V$, each being an unordered pair i, j of distinct nodes. Each node $i \in V$ corresponds to an entity. The edge set E consists of edges (i, j), such that i and $j \in V$. Each edge (i, j) represents an interaction between entities i and j. Graphs can be either directed, where the edges have orientations, or un-directed. Both nodes and edges of a static graph can be assigned weights. If both nodes and edges have weights the graph definition is further extended to

$$G = (V, E, f, g) \tag{2.1}$$

where f and g are functions, $f : V \to N$ and $g : E \to N$ where $N \in R$ is some number system which assigns a weight.

Dynamic Graph The minimal definition of a static graph involves two sets: V (a set of nodes) and E (a set of edges). Therefore, a minimal dynamic graph is obtained when either of these entities change over time. Ergo we can define a minimal dynamic graph as

$$G = (V, E, T) \tag{2.2}$$

where T represents some temporal component of the graph – this could be the time range covered by the graph. In addition, the definitions of V and E must extend to this new temporal component:

$$V = (u, t), (v, t), \cdots, (u^n, t^n) \tag{2.3}$$

$$E = (u, v, t) \tag{2.4}$$

where t always represents the time at which some change in value or structure occurred.

Event We discuss within the thesis the importance of 'events' within a dynamic graph. For our purposes an 'event' is some action that led to a change within the graph, usually timestamped at an atomic time and not as part of a flattened period of time. In a social network graph, for example, an event would be user i 'liking' the photo of user j at tx, adding an entry to the edge set such that

$$E = \{\cdots (i, j, t_x)\} \tag{2.5}$$

Event-based dynamic graph We define an event-based dynamic graph $D = (V, E)$ to be a graph whose attributes are a function of time. Let T be the time domain defined as an interval in $\mathbb{R}$. Attributes are functions defined in the domain $V \times T$ for nodes, and $E \times T$ for edges.

Timeslice In the wider computing community 'time slicing' is the period of time for which a process is allowed to run in a preemptive multitasking system. This is not the intended meaning in both this thesis and the dynamic graph visualisation community. When we refer to a 'timeslice' we refer to a fixed period of time which maps to a dynamic graph. Imagine a continuous timeline T, beginning at t_y and ending at t_{y+n}. Uniform timeslicing of the timeline T involves binning the entire timeline into equal intervals of time x such that each subsequent time point begins at the end point of the previous timeslice. For example,

$$T = \{(t_y, t_{y+x}), (t_{y+x}, t_{y+2x}), (t_{y+2x}, t_{y+3x}), \cdots\} \tag{2.6}$$

Non-uniform timeslicing While uniform timeslicing, as the name suggests, involves binning a graph with time components into set time sizes, non-uniform timeslicing does the opposite. With non-uniform timeslicing it is possible to instead bin the graph by using vertex or edge set size. For example, consider a dynamic graph $D = (V, E, T)$. Rather than uniformly dividing time into a set of k timeslices, we might instead timeslice by the size of the edge set, choosing to have $|E|/k$ edges per timeslice instead. Thus, having a uniform number of events, but a non-uniform distribution of timeslices across time. In areas of the dynamic graph where activity is high this may mean that the time interval for a timeslice is small – maybe 1 hour. By contrast, in areas of low graph activity this interval may be large, for example 3 days.

Both uniform and non-uniform timeslicing are continuous and cover the whole area of the timeline; that is all timeslices are temporally connected and the complete timeslice set begins and ends at the same point as the main timeline T.

User-defined timeslices In this thesis we introduce the notion of 'user-defined timeslices'. As the name suggests, this is a timeslice whose size and position in time is defined by the user. Therefore a user-defined timeslice A:

$$A = (t_x, t_y, V\star, E\star) \tag{2.7}$$

Where t_x, t_y defines start and end values of the timeslice and

$$t_x, t_y \in T \tag{2.8}$$

$$t_x < t_y \tag{2.9}$$

$$V\star = \{(v,t) | t_x >= t \text{ and } t_y <= t\} \tag{2.10}$$

$$E\star = \{(u,v,t) | u \in V* \text{ and } v \in V*\} \tag{2.11}$$

Interaction Provenance Provenance information is information about how a certain state has been attained. This can take the form of interaction provenance, where the information is about the interactions that have taken place with an interface. It can also be data provenance – information about the processes applied to the data and its source(s) that lead to its creation and current representation. In this thesis we focus on interaction provenance.

Node-link Visualisation Node-link diagrams show nodes in the network as points (dots) and (multiple) relations between nodes as straight lines. An example of this is shown on the left hand of Figure 2.1.

Matrix Visualisation Adjacency matrices typically place the full distinct set of node identifiers on both the x- and y-axis, with each node pair associated with a square within the matrix. Where there is a link between nodes the corresponding matrix square is then differentiated in some way, generally with colour, from the squares where there is no link between node pairs. An example of this is shown on the right hand of Figure 2.1.

2.2 Visualisation Task Taxonomies

Visualisation systems for complex data, such as temporal graphs, are frequently used for exploratory data analysis (EDA), particularly in cases where datasets are large, opaque, or otherwise difficult to sufficiently describe via purely numerical or statistical methods. Typically,

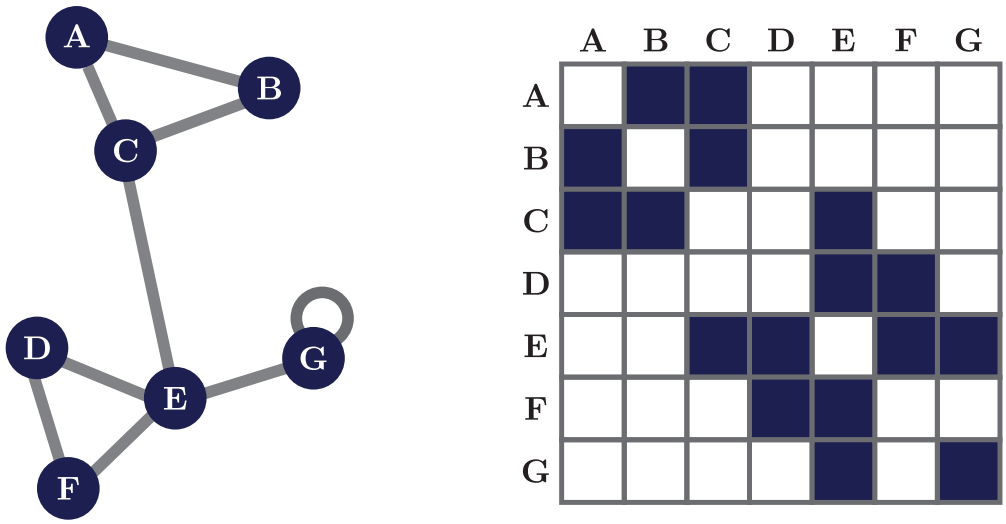


Figure 2.1: Example node-link visualisation (left) and matrix visualisation (right) of the same dataset. Notice how item G is a self-referential node.

data that is difficult to describe numerically is also challenging to visualise in a good way that facilitates EDA. Exploratory data analysis is nearly always task-driven (e.g. do we see this node act in a certain way? Does the structure of this cluster of nodes change over time?), and the completion or failure of these tasks will generally inform the further direction of the analysis. Further, the outcomes of earlier tasks will frequently influence later choices made concerning data filtering or analysis methods and may influence the interpretation of these results [3, 13].

Despite the importance of tasks to EDA, no single visualisation technique can support all types of data and tasks that an individual might aim to carry out during data analysis. Therefore we must specify tasks when designing visualisation systems for any use, particularly when designing visualisation systems for EDA or other early forms of data exploration. We must also understand appropriate techniques to facilitate the completion of the tasks that users consider key to their understanding of the data.

Task taxonomies aim to provide a complete classification or description of the possible range of tasks that can occur with a given data type, and some extend this by categorising tasks via common themes. However, task taxonomies rarely consider the cognitive load of the end-user in their enumerated tasks, and so using any task taxonomy as a task selection database (or interface feature selection list) should be approached with an amount of sympathy for the end-user [3] due to the negative impact that an increased cognitive load can have on the user.

A general drawback within the visualisation field is that systems are designed for sets of tasks but that these tasks are rarely clearly enumerated in accompanying literature and even less frequently discussed concerning their position within the wider task hierarchy. Experiments and formal studies might be more specific in their task selection (e.g. [120], [271], [20]), but they frequently use only a minimal range of tasks when compared to the full taxonomies [12, 285]. Wehrend and Lewis [330] first proposed a method to catalogue visualisation techniques by the tasks that they support, though given the current scale of the information visualisation area, a complete catalogue would be an incredible undertaking.

There exist a number of taxonomies for information visualisation tasks, both abstract [57, 90] and more concrete [7, 8, 11, 168, 280], but we are primarily interested in taxonomies generated for network data. Due to the relative lack of work in this area, we do not further isolate the discussion to only taxonomies of dynamic network data. However, it is the tasks described in these that we use to inform our development of the system described in Chapter 3, our understanding of the evaluation in Chapter 4, and our experiments to validate our new method of time exploration as described in Chapters 5 and 6.

Amar et al. [7] present a low-level task taxonomy for use with static graphs, and Lee et al. [190] extend this, categorising tasks into five graphs: topology, attribute, browsing, overview, and high-level tasks. One high-level task does hint towards the introduction of dynamic data: *how has the graph changed over time?*, yet they do not discuss any sub-tasks that might arise from this; or how the use of this high-level task might have implications on tasks described in other categories.

Shneiderman and Aris [283] classify several tasks as 'high priority', including the structural and attribute-based tasks identified by Lee [40], by first identifying six primary challenges for static network visualisation. They also recognise that, given every possible permutation and combination of tasks and datasets, the number of potential tasks that a user could carry out on a network is unlimited. None of the above [7, 190, 283] give consideration to temporal graph tasks.

For temporal graph data, Yi et al. [341] break down tasks into three main categories, depending on the level at which temporal change in the network is analysed. These are global level tasks; subgroup level tasks, typically based on connectivity or node attributes within a group; and node and edge level, generally the attributes of single nodes or edges or associations between attributes of a single element. However, they provide only a high-level overview of the composition of tasks. They do not consider the subtasks involved in analysing temporal

change for each category and also do not discuss hybrid tasks which may cut across groups or evolve organically during an analysis process.

Bach et al. [26] adapt a geo-temporal task framework from Peuquet [233] to apply to dynamic graph data. The framework from Peuquet defines three dimensions: when, where, and what. The original application of the Peuquet framework was geospatial data; therefore, it worked on the natural assumption that each event or point had a fixed place in the physical space, which could then be reflected more easily in a visualisation, but temporal graph data does not normally have this extra positional encoding. As a result, Bach et al. [26] use these dimensions as the basis for their framework but choose to redefine the what and where dimensions to reflect better the fact that temporal graph data lacks fixed spatial positions. They then formulate tasks using a combination of the three dimensional values, assuming that if two dimensions are known, then the aim of the general task type is the unknown dimensional value (e.g. if *what* and *when* are known, then the general task type is *where*). This framework is similar to the 'lookup' tasks in Andrienko framework [11], but the Bach framework misses more complex tasks such as comparison and finding dependencies.

Ahn et al. [2] provide a detailed taxonomy for temporal graph data, with a focus on the analysis of network evolution. Similar to Bach [26] and Yi [341], they use three dimensions as the basis for their classification. The dimensions from Ahn are property, entity, and temporal feature. 'Entity' is similar to the levels of analysis discussed by Yi [341], and 'property' makes a distinction between domain properties and structural attributes. Both of these describe what should be observed, but they do not completely align with the 'what' described by Bach [26]. The final dimension, temporal features, describes whether the items should be analysed as individual events at single time points or as aggregate events over a period of time. Ahn uses the three dimensions to construct a design space for temporal graph tasks, and they build on this as a tool for characterising existing visual techniques by the tasks they support.

However, during their evaluation of their taxonomy, Ahn et al. [2] find that their dimensions do not fully capture tasks relating to comparison, correlation, or inference, regardless of the general task level [342]. They later add a category for compound tasks to capture these missed items, but they do not seek to discuss how these task types could combine with their other dimensions within the design space they create. It is also logical that the analysis of individual events at single time points should fall under the taxonomy of static graphs, as the temporal component is removed in this situation. We, therefore, believe that the combination of the Ahn [2] taxonomy and some other static graph taxonomy (e.g. [7, 190, 283]) could offer a

more detailed description and a low-level task enumeration for this framework.

Andrienko and Andrienko [11] present a task taxonomy designed to apply to all types of data, and as a result, we can consider its application to temporal graph data. Unlike later work, which provides primarily verbal task descriptions, this taxonomy leverages a formal modelling approach and describes potential task combinations using functional language. The generalisability of this framework means that it is incredibly complex, so a complete description of it is not in the scope of this thesis. Instead, we provide a brief overview of their guiding principles.

Within the Andrienko framework, each task has two components: the unknown information being sought (the target) and the known conditions that the target needs to meet (the constraints). At its highest level, therefore, every task involves target seeking given a set of constraints. Andrienko and Andrienko also provide a data model (the Andrienko Data Model, or ADM) to divide data structures into two component types: referential or characteristic. Building on the ADM, Andrienko introduce the Andrienko Task Framework (the ATF), in which tasks are defined based on the ADM items that participate in them. First, they make distinctions between the level of analysis tasks: elementary tasks capture individual data items (e.g. individual node attributes), and synoptic tasks involve sets of items considered together as a single unified item (e.g. clusters of nodes within a graph structure). They further break down synoptic tasks into tasks that describe the data (descriptive tasks) and tasks that seek connections between data features (connectional tasks). In addition, they distinguish tasks by the role that involved data types take, i.e. whether a data type involved in a task is a target or a constraint. This categorisation leads to three main task types: lookup, comparison, and relation seeking.

However, Kerracher et al. [169, 170, 172] argue that the ATF, while powerful as a basis for task generation, provides task categorisation that is too abstract and generic to be easily applied as a mapping method for visual techniques that fall under its umbrella. Kerracher [172] also finds that the development of the ATF and ADM never considered graph data, and so they propose an extension of both the Framework and the Model to graph data and further apply their extended model to classify the entire set of tasks possible with temporal graph data. This extension introduces a new referrer type (graph) and a new type of relation (linking) to model the edges of a graph. Graph referrer type elements are discrete, unordered, and have distances, and the linking relation type is specific to the graph referrer type. This new referrer type and new relation type give rise to a novel set of structural tasks which apply only to graph referrers and linking relations.

As a result, Kerracher et al. [172] produce four classes of tasks involving very different data items. The vast differences in data items mean that each item type will likely need a different visual representation. The Kerracher classes can include the elementary/synoptic distinction introduced by Andrienko [11] and also describe four variants of synoptic tasks: elementary tasks, tasks considering graph subsets, tasks considering temporal subsets, and tasks considering both graph and temporal subsets. Each of these variants has different relations and components associated with it, and the behaviours participating in each class of task are highly distinct. For each class, there are a total of eight behaviours, split equally between attribute-based and structural.

Of the four synoptic task variants identified by Kerracher, the first two – elementary tasks and tasks considering graph subsets – also apply to static graphs, and so the tasks identified in the taxonomy of Lee et al. [190] can be positioned here. Furthermore, the temporal change level categorisation of Yi et al. [342] is also covered by the Kerracher framework, with node/edge level tasks covered by elementary and temporal subset tasks and the subgroup and global level tasks included in the general graph subset and combined graph and temporal subset tasks.

By extending the formal language definition of task generation first implemented by Andrienko [11], Kerracher extracts all possible task permutations [171], defining 127 possible tasks for dynamic graph data. In using a formal model, it is more easily possible to demonstrate the completeness of the task framework relative to the chosen data model and abstraction level; this approach also ensures that task specifications are at a consistent level of granularity and abstraction. In addition, the formal model also ensures that generated tasks are independent of the application domains. However, while the use of the formal model is robust and ensures completeness, the exclusion of users from the task generation process leads to several limitations, primarily that there is no way of knowing how commonly each identified task might be completed or how practical a particular task is for a specific set of users.

In this thesis, we also seek to make a difference between temporal graph data and event-sequence data, though the former can act as a subset of the latter in some instances. The primary difference that we encounter is that temporal graph data usually seeks to reflect the entire graph, or as much of it as possible, whereas event-sequence data is typically more egocentric (i.e. relating to a single person or process). However, as certain temporal graph tasks can push the analysis into an event-sequence analysis, as we see in Chapter 4, we also include event-sequence data task classifications here.

Plaisant and Shneiderman [236] characterise event data along three dimensions and pro-

pose high-level user tasks. They identify many diverse data characteristics, breaking down tasks into three high-level aims: 1) to heighten awareness, 2) to prepare or select data for further study, and 3) to understand the impact of event patterns and plan further analysis action.

Ruddle et al. [261] also introduce high-level task descriptions for the visualisation of event-sequence data in a healthcare setting. These are: 1) simplifying a sequence; 2) finding a sub-sequence; 3) understanding longitudinal changes; 4) comparing sequences. We observe practitioners carrying out analysis with our tool in Chapter 4 and see many of these high-level tasks reflected in their analysis choices and aims.

It is also important to consider that, within the visualisation literature, the term 'task' is used at multiple levels of abstraction and granularity [57, 214, 252, 275], and there is some discussion regarding the use of abstract task classifications, which offer the ability to generalise beyond a specific use case and facilitate communication between researchers, and the use of data specific task classifications [57], whose primary utility is for the assessment of tools and techniques in a task-focused manner. Table 2.1 describes our application of the existing graph task literature to our work in this thesis.

2.3 Dynamic Network Visualisation

A dynamic network is a network that undergoes some change over time. This change can be related to nodes or edges, and changes can be grouped into specific events. The field of dynamic network visualisation is wide-ranging; and we first examine the design space of the area, moving into surveys providing an overview of the whole research area and justification for our research classification method, before narrowing our focus to more specific works.

2.3.1 The Design Space of Dynamic Network Visualisation

In the general case, a design space can be used to map out "the universe of all possible design choices" [275], with Schulz et al. [274] describing design spaces as a tool to identify "the space of the possible" [274]. We can construct a design space by combining all independent dimensions of a taxonomy to produce all possible combinations and permutations; this method has previously been used to generate possible spaces for visual techniques [162, 274] and tasks [2, 275]. This combinatorial mapping of existing techniques can give rise to unexplored possibilities, offering a direction to inform future work [274].

Topic	References	Relationship to this thesis	Chapters this literature is used in
Static graph tasks / tasks at disjoint time points	[7, 190, 283]	The high-level task classifications were used to identify tasks that users might want to perform at separate time slices so that we could provide functionality to support them.	Chapter 3, Chapter 5, Chapter 6
Static graph tasks / tasks at disjoint time points	[7, 190, 283]	High-level task classifications were used to improve our hypotheses of the impact that distance in time would have on some interfaces and tasks.	Chapter 3, Chapter 5, Chapter 6
The what, where, and when framework	[2, 26, 233]	We use a change in node size as an experimental task, participants are given the 'what' (the node) and the 'where' (four separate time points) and have to find the 'when' (the time point at which the node is smallest).	Chapter 5, Chapter 6
Full dynamic network task taxonomy	[11, 169–172]	We used the Kerracher framework to contextualise the tasks that we observed expert practitioners carry out with our visualisation system, and also selected two tasks of the temporal structural type for our experiments to validate our new method of time exploration.	Chapter 4, Chapter 5, Chapter 6
Event-based dynamic graph tasks	[236, 261]	We observe that some tasks carried out by our expert practitioners stray into the 'event-based' categorisation, and provide support for these task types within our novel visualisation system.	Chapter 3, Chapter 4,

Table 2.1: How we use the task literature throughout the thesis to inform all stages of the product development life cycle.

Design spaces also allow us to map existing visualisation techniques to the tasks they support, uncovering areas that are perhaps under-researched or already well covered in the literature [2]. Efficient use of these mapping techniques can be used to guide further system development and can also signpost areas that would benefit from controlled experiments in the case where multiple systems support the same task(s) as we could then evaluate which technique from a set is the most effective for its claimed task.

Both Amar et al. [8] and Sedig et al. [276] support the idea that task classifications can be used to provide a more systematic basis for a design process, while others [8, 141, 183] instead recommend using classifications as a checklist of items to consider during the design process.

However, for visualisation systems, we do not limit the design space purely to the tasks that a system might support. We can define a design space, though perhaps not a particularly good one, based upon any arbitrary classification that we can apply to all visualisation methods. In terms of dynamic graph visualisation literature, this might be in terms of the treatment of time (timeline basis, real-time basis, neither, or both), the form of visual representation of the data (node-link diagrams, matrices, line charts, scatterplots, or some combination of all of these), a nested version of this (as seen in the classification of Beck et al. [37]), or some other alternative method, such as the space-time cube taxonomy of Bach et al. [24].

There is some discussion within the field regarding the classification method for approaches that visualise dynamic graph data. Hadlak et al. [133] base their categorisation on the reduction techniques used; what element of the graph is reduced (temporal or structural), how the reduction is carried out, or if there is no reduction of any element. Federico et al. [103] use the mapping of the temporal dimension as the basis for their division of techniques, as do Rufiange et al. [263], Kerracher et al. [170], and Beck et al. [37]. von Landesberger et al. [315] use the temporal dimension in a slightly different way, classifying graphs according to their time dependence.

In each of these existing classifications, a common thread is the distinction between the temporal encodings and the structural dimensions of the graph. Little existing design classification work examines the effect of task choice on visual representations or the specific datasets that systems are designed around. A slightly more common discussion relates to the challenges that users face in their visualisation aims. However, these broader challenges do not necessarily map either partially or wholly to tasks specific to dynamic graphs, as we see in the development of our visualisation system detailed Chapter 3.

2.3.2 Surveys

Early work by Herman et al. [147] does not differentiate between static and dynamic data but offers an overview of current methods available for graph visualisation and navigation. They highlight the need for improved graph navigation to mitigate the scalability problem of laying out very large graphs and discuss the importance of the user's cognitive load. Areas particularly highlighted are the need for improved graph navigation, as it is not possible for layout alone to solve the problem of visualising large graphs, and that the size of these graphs is a crucial issue within the area; there comes some point where performance is compromised, or the limits of the viewing platform are reached. It is also noted that the cognitive load of these graph visualisations must be taken into account to be made as helpful as possible for users.

Müller et al. [213] define their focus to the visualisation of temporal network data. They conclude that the time factor requires special treatment to allow the visual exploration of time-dependent data and that many existing visualisations do not consider time a unique factor, but instead treat it as another parameter among many.

Aigner et al. [3] create a systematic view of methods for visualising time-oriented data, concluding that task-orientation should be the primary consideration when designing a visualisation method, similar to other work [11]. They recommend using multiple views for different data aspects, interaction facilities specialised for time-oriented data, and tighter integration of visual and analytical methods.

Large graphs, not necessarily formed with dynamic data, are the focus of von Landesberger et al. [315]. They look at both visualisation methods and graph layout algorithms. The recommendations made for future work align with those identified previously [3, 147, 213].

Kerracher et al. [170] describe a further survey focusing on the visualisation of dynamic data. They classify existing methods using two independent dimensions: graph structural encoding and temporal encoding. They find that the most prevalent method is still node-link, as highlighted in previous work [315], but also note that matrix-based techniques consistently outperform node-link graphs on tasks in a static context [170]. They also find that sequential views are still the most popular for dynamic views, despite juxtaposed views performing better in user tests [170]. Building on this, Beck et al. [37] separate dynamic graph visualisations into two categories based on their method for encoding time. Time-to-time mappings represent time naturally via the temporal dimension, the most common example of this being animation. Time-to-space mappings use one or more spatial dimensions to encode the temporal information; a typical example of this is small multiples. They also find that irrespective of the

treatment of time, node-link diagrams are the most popular representation, supporting previous work [170]. They strongly suggest that matrix-based methods are the solution to dense graphs with many edges due to the nature of their layout, further supported by Kerracher et al. [170]. Suggested future work correlates very heavily with von Landesberger et al. [315].

Bach et al. [24] describe each method as an operation on a conceptual space-time cube, then transforming that cube into a readable 2D visualisation. This approach provides a useful overview of existing methods but was mainly developed as a tool for teaching undergraduate students about temporal graph visualisation. Bruder et al. [60] build on this concept to create a volumetric representation of a graph by stacking each of the adjacency matrices of its time steps, and also facilitate three important analytics methods: data views, aggregation and filtering, and comparison.

Recommendations for future work are mostly consistent over time, with the same research gaps highlighted; primarily the requirement for improved graph navigation and the importance of considering the cognitive load of the visualisation on users [3, 147, 170, 315]. Common suggestions are that visualisations should be designed from a task-orientation standpoint and that multiple views for different data aspects may support more complex tasks [3, 170, 213]. We take the organisation of the below literature is from the temporal graph visualisation technique classification created by Beck et al. [37]. The treatment of time is the main division - time is either mapped as time-to-time (animation) in Section 2.3.3, time-to-space (timeline) in Section 2.3.4, a hybrid approach of these two in Section 2.3.5, or, in rare cases, time is not mapped in either manner.

2.3.3 Time-to-Time Mapping

Animation involves mapping the graph at each time stamp and then running these mappings concurrently as an interactive video, illustrated by Figure 2.2. When the mapping of the graph takes the form of node-link diagrams, the result is a reasonably intuitive dynamic graph visualisation. Node-link diagrams remain the most popular approach [26, 53, 99, 110, 122, 138], with one work combining matrices and animation [264]. A notable exception to this is Elmqvist et al. [97], who use neither node-link diagrams nor matrices for their tool to show information flow in a system of interacting processes.

Bach et al. [26] propose an improvement to animated visualisations. They use animated transitions to highlight changes in the network between time steps to help users identify and understand those changes. Archambault et al. [20] find that using difference maps for this

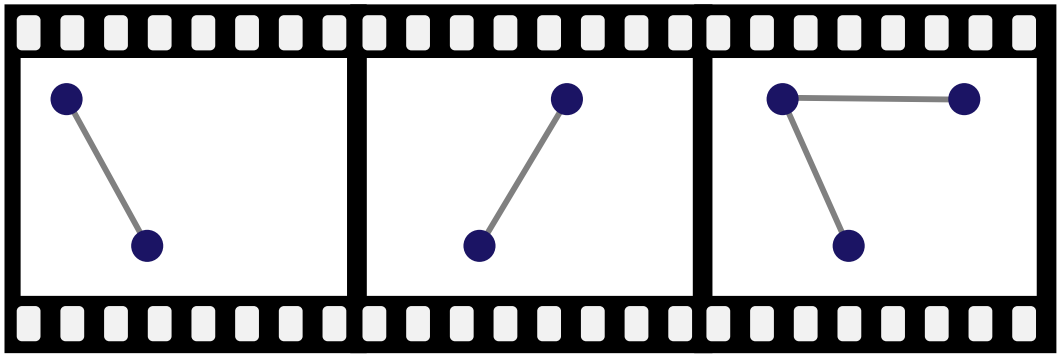


Figure 2.2: A representation of the way that animated approaches map the time dimension to actual 'real world' time. Each window of the film is shown in order, with the previous window being replaced by the next.

purpose leads to a reduced error rate and higher participant satisfaction, validating the high-level concept of this approach.

The principal drawback of animation is that all temporal information is not displayed simultaneously and that the user must interact with the system to show certain periods of time, relying heavily on memory for moments not displayed on the screen. Multiple experiments, discussed in Section 2.3, find that small multiples frequently outperform animation for a variety of task types [18, 102].

2.3.4 Time-to-Space Mapping

Instead of using animation, a graph can be drawn onto a timeline in a time-to-space mapping, as shown in Figure 2.3. Timeline-based approaches promise to provide a better overview of time as they show the complete sequence of graphs in a static image. Often there is only a little space available for drawing each graph, decreasing the readability of the representation. This visual scalability problem is one of the main challenges for the techniques in this category.

2.3.4.1 Node-Link

We have defined node-link diagrams in Chapter 2.1, and a simple example of one can been seen in Figure 2.1. A node-link diagram is some object or shape, representing a person, place, or other entity joined to another object or shape. Typically the nodes are drawn as circles representing entities, and edges are the lines that join the nodes, representing some connection between them.

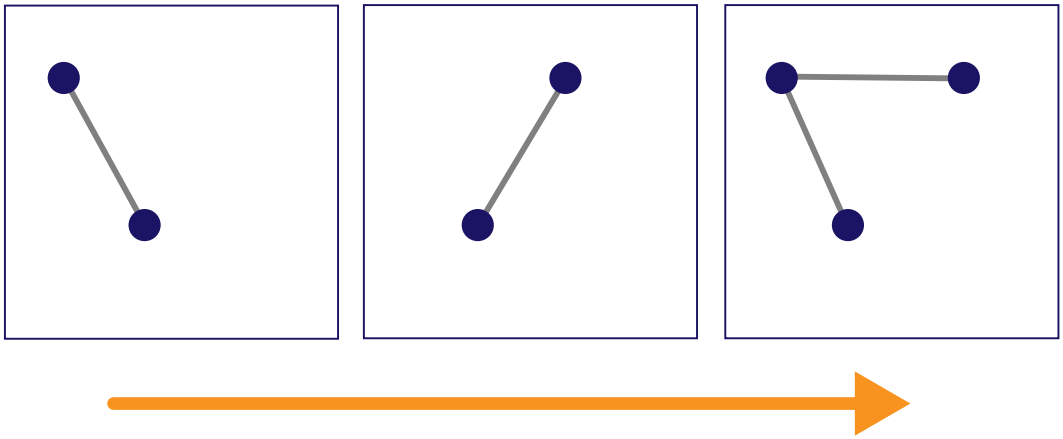


Figure 2.3: A representation of the way that time-to-space approaches map the time dimension to a dimension on the screen. The orange arrow represents the progression of time in the dataset, and in this method each window containing a diagram would be continuously visible, regardless of the passing of 'real world' time.

Node-link diagrams have multiple facets of readability and usability; including, but not limited to, the layout of the graph, edge encoding, node encoding, the scalability of the representation, and the stability of the graph. We are specifically interested in the practicalities of using node-link diagrams for dynamic network visualisation on large displays, however, due to the relative lack of work on this specific area we widen our lens to previous work on node-link diagrams for dynamic networks in any context.

Multiple works build on the abstract concept of parallel coordinate plots and MDS [63, 64, 66, 82, 240]. Bach et al. [28] also employ MDS to embed time-points in 2D space and show all data points laid out on the curve, rather than having them viewed separately as time slices, similar work by Collins et al. [82] also employs MDS. Van den Elzen et al. [310] also use MDS in a way not dissimilar to Bach et al. [28] via juxtaposed views for the comparison of network snapshots and network evolution. Gansner et al. [115] set out to visualise real-time text streaming data from Twitter. They use MDS to co-locate tweets on similar topics. Due to the constraints of the display area, the number of displayed messages are limited, limiting the application of the method to smaller datasets. They claim that approaches used to preserve the mental map can "introduce biases in the visual representation of the underlying data, which can persist over time" [115], and offer the opinion that their method of supporting the mental map via MDS and a Procrustes transformation is superior.

Alencar et al. [5] create similarity maps using a cumulative backwards strategy to preserve

the mental map. Burch et al. [63] also work to support the mental map by facilitating the display of all timeslices on the screen simultaneously. They use a radial layout, encoding graph information on narrow rings of a circle. However, this is challenging to read for non-expert users, and no studies support this method's utility. Burch [62] initially use a node-link layout but then transform this layout, via splatting, to a radial representation (similar to [313]). Further, Burch [61] then applies methods to transform these radial node-link diagrams into smooth density fields. The resulting overview representation is a starting point for further analysis and facilitates the rapid visual comparison of the sequence of graphs.

Dwyer et al. [94] and Groh et al. [128] both create three dimensional visualisations that use 'tubes' or 'worms' to encode graph elements. Brandes et al. [50] and Itoh et al. [157] also use 3D visualisations, though in these cases they stack multiple flattened timeslice representations, rather than continuously drawing time as with the 'tubes'. Federico et al. [103] present a tool that offers three views to preserve the users' mental map. The first view is a superimposition of all selected timeslices, the second a juxtaposition, and the third a two-and-a-half dimensional view. The use of a 'more than 2D' approach seems to be a popular one when trying to compare timeslices, despite common drawbacks with 3D visualisations [215].

Shi et al. [278] choose to focus only on the dynamic ego network, with network navigation achieved by the user switching the focus node. In order to generate the dynamic ego network, all static ego networks for that point are combined. They suggest that this method reduces visual complexity and makes pattern-finding tasks on the whole network easier due to all data being displayed in a single view.

Kumar et al. [182] introduce a hierarchical force-directed layout; claiming that their view allows the data to be presented in a way that abstracts away many small structural details. This case is more of a static method applied to dynamic data and expected to work, rather than a method developed with specifically visualising dynamic data in mind. Pohl et al. [238] explore the visualisation of dynamic hierarchical networks with their Xldn visualisation tool. The layout algorithm focuses on maintaining the users' mental map and requires knowledge of the complete network sequence to work.

Rufiange et al. [263] introduce DiffAni, a tool that treats graph visualisation as a series of three different types of tiles. This tool also allows the visualisation of several networks simultaneously, similar to previous work [102, 143]. Turker et al. [308] introduce a method that uses Hyperbolic Geometry to support focus context rendering. User-defined network metrics are also encoded into the temporal layout to improve the information density of the visualisation.

Node-link based representations are most suitable for small or sparse graphs due to the number of edge crossings in large or dense graphs. Equally, node-link representations are useless without a good layout (e.g. force-directed, Fruchterman-Rhinegold) and suffer from occlusion problems related to node overlap and edge crossings. Thus, they are not readable for dense graphs, and any layout manipulation requires further layout re-computation. However, node-link diagrams are superior for tasks involving path-finding or following, and with a good layout can be more readable than matrices. They also facilitate the quicker identification of between sub-group links and are easier for non-expert users to gain quick insights into the overall graph structure.

Our contributions primarily make use of node-link diagrams to draw temporal graph data. Our novel interface for the visual analysis of dynamic data (Chapter 3) always draws the graph as a node-link diagram as an initial step; though we do offer alternative views in a way similar to Henry et al. [144]. Our two laboratory studies, Chapters 5 and 6, also use node-link diagrams to display graph structure after data has been timesliced by our participants. While there is still no consensus on the best graph drawing algorithm for dynamic data, or even on what features define a good dynamic graph drawing algorithm, we mitigate this through several strategies that are detailed in Chapter 3.

2.3.4.2 Matrix

Adjacency matrices typically place the full distinct set of node identifiers on both the x- and y-axis, with each node pair associated with a square within the matrix. Where there is a link between nodes, the corresponding matrix square is then differentiated in some way, generally with colour, from the squares where there is no link between node pairs.

Yi et al. [341] introduce 'TimeMatrix', but rather than simple binary colouring for cells they instead create a TimeCell - a visual aggregate that displays temporal information associated with a node or edge as a composite glyph.

Corresponding interaction techniques allow for investigating a network at multiple granularity levels and layouts. In addition, overlays and filters allow for comparing temporal data and statistical information.

Brandes et al. [52] continue combining glyphs and matrix-based representations by using a matrix representation of Gestaltlines - a combination of Tufte's 'Sparklines' with glyphs based on Gestalt Theory. The result is a static, compact, and data-rich diagram that facilitates the exploration of evolving dyadic relations and persisting group structure.

Burch et al. [65] also encode data information within matrix cells. They use this pixel-based representation to generate a graph overview by applying weight aggregation or overplotting the single data points. They find that using a static diagram to display dynamic data allows for better preservation of the mental map, and allows for the better comparison of several weighted relations in specific time intervals.

Vehlow et al. [313] use a radial layered matrix to display dynamic directed weighted graphs in which vertices can also be hierarchically organised. This approach addresses the investigation of temporal changes rather than the graph structure at the particular point. This can be considered a matrix-based version of other work that applies radial layouts to node-link diagrams [61–63]. Yoghourdjian et al. [345] go a step further and introduce Graph Thumbnails; representations designed to fit into a small multiples structure but that cannot be classified as either node-link or matrix-based. The resulting time-step representations could be best characterised as a series of concentric circles, with each circle representing a group or sub-group within the graph.

Van Ham et al. [312] show how matrix-based visualisations can be used in combination with algorithmic metrics to help researchers find new patterns in the datasets; especially for very large sets. They utilise a hierarchy on the node-set to reduce the size of the adjacency matrix. Stein et al. [294] extend the work of van Ham et al. [312] by using matrix-based techniques as an addition to traditional graph visualisation. They use a static visualisation of the dynamic data and find that this visualisation is most effective on medium-sized networks where the whole matrix can be displayed simultaneously. The use of a large display, in this case, would be beneficial and all but remove network size constraints due to the available amount of screen 'real-estate'. The PONV method focuses more on displaying temporal patterns rather than detecting network clusters, which also means it is not suitable for use on sparse networks with low traffic.

Bach et al. [25] introduce the method of Small MultiPiles, in which adjacency matrices, each representing a single temporal snapshot, can be 'stacked' based upon similarity or a change in data. This method reduces visual complexity, and they claim that the action of flipping through the snapshots supports the pre-attentive perception of changes. The technique has also been extended to work in a computational biology setting [314].

Matrix-based representations can be 'reordered' through successive permutations of rows and columns to reveal interesting patterns in the graph structure. One of the main advantages of this representation is to avoid the occlusion and graph layout problems encountered using

node-link representations. Matrices are efficient when performing basic tasks like identifying the most connected node, a link between two nodes, or a common neighbour of two nodes. However, they perform poorly on more complex tasks such as finding a path between two nodes, even in small matrices. The lack of drawn nodes and edges in a matrix enables them to be readable for dense graphs and have comparatively faster navigation and manipulation. They are also superior for some graph tasks. On the other hand, a matrix-based representation is rarely intuitive and most, if not all, will need to be explained to a user before they can gain any insight from the visualisation. The space used can be a problem on smaller screens or with very large datasets and, due to the lack of edges, they perform very poorly for path following tasks; even on relatively small graphs.

We include matrix-based representations as a user-controlled option in our interface detailed in Chapter 3. By synchronising the matrix representation with node-link visualisations we can mitigate the issues surrounding poor performance for specific tasks, and we find that they work well alongside the node-link representations, explored in Chapter 4. However, we do not validate this experimentally.

2.3.5 Hybrid and other

While most dynamic graph visualisation techniques can be unambiguously classified as using node-link or matrix-based approaches, some combine both methods, and some use neither. The combination of the two representations can, however, follow different strategies in those hybrid approaches. Hybrid strategies are only those that use both representations closely connected and cannot easily be split into independent techniques.

Henry et al. [143] first introduce MatrixExplorer; a tool that allows both node-link and matrix representations to be created at the same time, and keeps them synchronised to that the user can work with both and switch from one to the other without disruption. Further work from Henry et al. [144] results in MatLink - a matrix-based visualisation with links between cells overlaid on the borders of the matrix. A user experiment finds that MatLink significantly outperforms ordinary matrix-based visualisations for path-related tasks, while still providing useful functionality for tasks that are not possible on traditional node-link representations.

Similarly to previous work [143, 144], Linhares et al. [194] introduce a system which can provide structural visualisations via node-link representation, and temporal views as matrix-style timelines. They ensure coordination between the two views for usability purposes. Cakmak et al. [67] also produce a comparable solution combining node-link, matrix, and activity

line representations to visualise temporal summaries of dynamic graphs.

Reitz et al. [249] reduce large and complex relations to small graphs resembling ego-centred networks; to convey when and how strong a relation evolved. Edges connecting actors are coloured using different matrix-style encoding depending on relations between actors. Visualisation results are promising, though this representation could have a high cognitive load for the user.

Farrugia et al. [102] also focus on visualisations for ego networks, this time placing the time dimension in the foreground by turning time into an element of shape. They also develop a system to enable the visualisation of multiple networks simultaneously, similar to MatrixExplorer [143], through the use of small multiples. In contrast to most dynamic network visualisation where the focus is to study network evolution over time, this work aims to support the analysis of actor rewiring. A tree ring layout is used to display the dynamic ego network, and from this, a small multiples interactive system is created.

Sallaberry et al. [267] contribute a new clustering algorithm which finds the ideal clustering for each time step and then links the clusters together. The resulting time-varying clusters are then used to create an overview, showing how clusters evolve, and another view consisting of a node-link diagram of a selected time-step. They place particular emphasis on the stability of the visualisation and the preservation of the mental map.

By using a hybrid technique, there is the potential for increased task support - tasks that would have been impossible for a node-link layout can now be carried out using a matrix-based representation, and vice-versa. We contribute to the area of hybrid visualisation via our system detailed in Chapter 3 as we simultaneously support synchronised matrices, node-link diagrams, and scatterplots with the user given control over the axis, colour, and size encodings. Table 2.2 describes how we apply the existing dynamic network visualisation research discussed in this chapter to the work presented in this thesis.

Table 2.2: How we apply the literature from Chapter 2.3 to the contributions in the thesis.

Beginning of Table 2.2			
Topic	References	Relationship to this thesis	Chapters this literature is used in
Gaps within the wider temporal graph visualisation literature	[3, 147, 170, 315]	We take recommendations for future work resulting in the always-on interaction provenance (to aid navigation reduce load on user memory) and our new method of exploring and interacting with the temporal dimension.	Chapter 3, 5, 6
Support for multiple views	[3, 170, 213]	Suggestions that multiple views for different data aspects could support more complex graph tasks led us to introduce the option for multiple representations within our visualisation system.	Chapter 3
The popularity of the time-to-space approach	[3, 37, 103, 133, 147, 170, 213, 263, 315]	Our understanding of the popularity of the time-to-space approach led us to conclude that our novel system should be situated within this area, but that we should offer an alternative treatment of time within this classification group.	Chapter 3, 5, 6

Continuation of Table 2.2

Topic	References	Relationship to this thesis	Chapters this literature is used in
Design choices made by systems that support time-to-time visualisations	[26, 53, 97, 99, 110, 122, 138, 264]	While we do not map time to time (i.e. animation) in our primary contribution of this thesis, we were able to apply some of the design choices of these interfaces to the animation interface that we tested to evaluate our new method of time exploration.	Chapter 5 and 6
Time-to-space, node-link visualisations, and multi-dimensional scaling	[28, 63, 64, 66, 82, 82, 240, 310]	Work using MDS does not traditionally treat time as though it is a simple timeline. We take the different treatment of time and build on this idea to develop our novel method of time exploration.	Chapter 3, 5, 6
Time-to-space, scalability of many timeslices	[5, 61–63]	The idea of constant representation of timeslices that are not necessarily close in time helped to inform our new method for navigation in time.	Chapter 3, 5, 6

Continuation of Table 2.2

Topic	References	Relationship to this thesis	Chapters this literature is used in
Time-to-space, matrices	[25, 52, 65, 294, 312–314, 341, 345]	We include a matrix-based representation within our novel visualisation system to ensure that we are providing support for tasks that are more suitable to be carried out with matrix-based representations. However, in our expert evaluation of the system, we found that our users generally found the matrices confusing or too dense to be readable, supporting previous work.	Chapter 3, 4

Continuation of Table 2.2

Topic	References	Relationship to this thesis	Chapters this literature is used in
Time-to-space, hybrid visualisations	[67, 102, 143, 144, 194, 249, 267]	Given that matrices are more suited to some tasks than node-link diagrams, and vice versa, we make a contribution to the hybrid area with our visualisation system as we provide support for matrices, node-link diagrams, and also scatterplots. We provide some coordination between views, supported by previous work, but observe that our expert users have problems with mapping matrices to node-link diagrams. From this, we conclude that the most usable hybrid system is one that maintains a close link between the representations it supports.	Chapter 3, 4

2.3.6 Interaction with dynamic network visualisations

There are many different ways for a user to interact with dynamic network visualisations [342], both in the physical sense (e.g. keyboard and mouse, touch screen, VR 'hand' interaction) and in terms of the operations that are passed to the system independently of the physical interaction style. As we cover visualisation and interaction with large displays in Section 2.8, we here discuss the operations that can be used to pass information into visualisation systems.

Shneiderman introduces the concept of Direct Manipulation (DM) [281], an interaction style where "users act on displayed objects of interest using physical, incremental, reversible actions whose effects are immediately visible on the screen." [281]. This type of manipula-

tion requires that users always see visual representations of the object that they can interact with, and that they receive immediate visual feedback during and after their interaction. This immediate visual feedback makes it easy for the user to validate that their interaction has had the desired result, and so is able to facilitate the rapid learning of the system and its operations. Direct manipulation is the basis for the modern GUI systems and fundamentally extends visualisation capabilities.

The use of DM facilitates several common interaction techniques. The most well known, **details-on-demand** [280], provides improved scalability by displaying information about data to the user only on demand, often moving views from the high-level aggregation of objects to a subset of the elements that make up the aggregation view. This style of interaction is best seen in the system introduced in Chapter 3, where we use an aggregated activity timeline to provide an entry point to the system and the visual analysis of the dataset. Other methods are the **direct walk** technique [39, 343], which allows the display of linkages between items where the exploration of one item may lead to another (e.g. hyperlinks on web pages), and the **manipulating views** method [297], where views can be rearranged and re positioned (e.g. sorting items in a table) or the representation type can be changed (e.g. histogram to scatterplot). We offer support for representation switching in our system detailed in Chapter 3, particularly as experiments have supported the idea that matrices are superior to node link diagrams for some graph tasks and graph types [250]. A further technique is **linking** [253], which is used to display the connection between multiple views of the same data space. In linking it is also usually the case that an update to one view updates all views. We also facilitate this style of interaction in the system presented in Chapter 3, though in a slightly different way to the classic imagining of linking – we allow updates in both a row-wise and column-wise manner, depending on the operation performed by the user.

Brushing [38] is an operation that allows the user to interactively select data items from a visualisation. Brushing was originally intended to be used to highlight brushed data items in different views within the same visualisation [38], though this has been extended and brushing is now more commonly used as a method to select items to perform any operation on them. At its highest level, a brush is an interface tool used to select subsets of data in a single view. When combined with linking and linked views the brushing operation can support the identification of common items and data features across multiple views and dimensions. Brushing is also a powerful combination when used alongside the details-on-demand operation, with users brushing directly on an overview (fulfilling the criteria of the DM) to select an area to explore

deeper and obtain a non-aggregated view. There are several different types of brushes – sweeping brush, composite brushes made up of multiple single-axis brushed, angular brushes, and smooth brushes [72, 137, 254]. We make use of multiple sweeping brushes in our system of Chapter 3 to allow users to select multiple areas to view details for simultaneously, allowing us to facilitate the easy comparison of several areas of the graph that are far apart in time. We validate this approach of time comparison in Chapters 5 and 6.

Zooming offers another way of moving from overview to detail, as laid out by details-on-demand. In the simplest form, zooming allows for the scale and translation of the viewpoint [80, 113]. There are three zoom types – geometric zoom, fisheye zoom, and semantic zoom. With a geometric zoom, the user specifies the scale of the magnification and the zoom items have their magnification changed by that scale. Fisheye zoom is similar to geometric zoom, but the information outside the zoom area is distorted, as opposed to being lost from view. For semantic zoom, objects stay the same size but move further apart as you zoom. Semantic zoom can also be used to progressively change the shape or context of information presentation, for example a user might zoom in on a circle which changes to a box, then a labelled box, then a rectangle with images and text. As the user zooms out this process is reversed, with only the presentation of the underlying information changing.

Dynamic graph visualisations rely on the use of at least one of the interaction methods detailed above, and due to the complex nature of the data and the associated interfaces they more commonly leverage nearly all of these; though not necessarily in the same way or for the same purposes. Evaluations of dynamic graph visualisation have classically focused on the readability of the drawn graph, with existing work evaluating methods of calculating the graph layout (for node-link diagrams), visually representing the graph (generally studies comparing node-link diagrams and matrices), and the best method for encoding time (largely time-to-time vs. time-to-space approaches). Little literature explores the best way of interacting with these representations, though the argument could be made that we have to know what can be classified as a good representation before trying to understand the best way to interact with it. However, given the very close relationship between representation type and interaction for these highly complex datasets, it would perhaps be more suitable for studies to examine not just each in isolation, but also combinations of representations, interactions, and time encodings. A summary of how we apply the literature relating to visualisation interaction to the work in this thesis is shown in Table 2.3.

Table 2.3: How we apply the literature in Chapter 2.3.6 to the contributions in this thesis.

Beginning of Table 2.3			
Topic	References	Relationship to this thesis	Chapters this literature is used in
Details-on-demand	[280]	Our visualisation system is built upon the foundation of details-on-demand by providing an aggregated activity timeline as an entry point and only showing lower-level data after specific user interaction and definition.	Chapters 3, 5, and 6
Manipulating views	[297]	We allow views in the form of columns to be manipulated by moving one top level item, and allow rows to be removed from the analysis by the user.	Chapters 3, 5, and 6
Linking	[253]	We provide support for linked views in our novel visualisation system via the row/column structure that we build on, and also via node selection and highlighting across all views generated during the analysis process.	Chapters 3, 5, and 6
Brushing	[38, 72, 137, 254]	Brushing is the operation that provides the primary entry point to our system, we use it to allow the selection of both time and graph elements of interest.	Chapters 3, 5, and 6

Continuation of Table 2.3			
Topic	References	Relationship to this thesis	Chapters this literature is used in
Zooming	[80, 113]	We use a geometric zoom within the system as a method of filtering, building further upon the details-on-demand foundation.	Chapters 3, 5, and 6

2.4 Methods for the evaluation of visualisation systems

In the evaluation of information visualisations the goal is often to understand how well different visualisation approaches support users in information tasks, or to understand how users conduct their information related tasks so that we can design visualisations to better support their workflows. For each of these goals we can study people while they use visualisation software or study them independently of software to better understand their processes. While a full enumeration of methods for visualisation evaluation is out of the scope of this thesis (work by McGrath [207] provides comprehensive detail, and Carpendale [69] explains how these relate to the information visualisation field) we provide a brief overview of the high-level concepts and categorisation of evaluation methods.

Study design tends to target some combination of the three primary factors: generalisability, precision, and real-world similarity [207]. However, no single methodology is able to support all three of these, favouring instead one or two at the expense of the others. As a result, methodologies are chosen with a particular goal in mind. Figure 2.4, adapted and simplified by Carpendale [69] from McGrath [207], shows the relationship between each method and the three primary factors.

In addition to each of the factors there are two main groups that evaluations fall into, quantitative and qualitative evaluation. Quantitative evaluations, generally laboratory experiments or studies, are studies which tend to have high precision and which some judgements can be made about the generalisability of the results. The development of these studies involves a rigorous process of hypothesis development, variable identification, and the application of some statistical measure to the results to allow the researcher to declare their confidence in the results

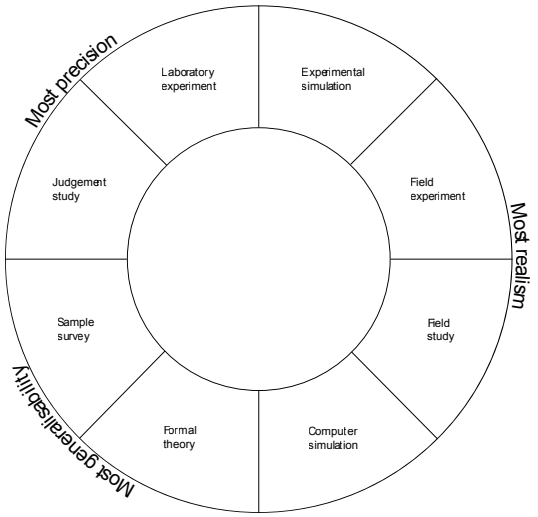


Figure 2.4: An illustration of the relationship between the three primary factors and their associated evaluation methods. This figure was adapted from Carpendale's [69] simplification of the classification for social scientists by McGrath [207].

of their study. Challenges for this type of experiment take the form of errors in relationship finding (e.g. Type I or Type II errors) and the validity of the measurements taken. This latter challenge will effect the extent to which the results of a study can be generalised. Qualitative studies use a more holistic approach that considers the interplay among factors that influence visualisations, their development, and their use [69, 231]. It is relatively easy to incorporate qualitative techniques in all types of studies, including studies of the design process, interviews, field studies, and observational studies that are used to inform that design and development of visualisation systems. The goal of a qualitative research method is to collect data to enable a rich description, rather than to use statistics to make inferences or draw conclusions [22, 192]. However, qualitative methods generally suffer from issues related to small sample sizes, the subjectivity of things reported by participants, and the associated challenges of analysing data collected. There are situations in which qualitative methods can be combined with quantitative methods to give a more human-focused context to statistical results, such as interviewing users before or after experimental studies, or asking the user to use a 'talk aloud' protocol during an experiment.

Within the graph visualisation field, and more specifically the dynamic graph visualisation literature, it is most common for new and novel visualisation approaches to be introduced alongside some form of qualitative evaluation – perhaps a small-scale field study, or a case

study using domain data but that is carried out by the visualisation creators. Quantitative experimental evaluations then usually take place at a later date, and often involve separating out the key novel features of a visualisation and testing them individually to avoid experimental confounds. For these experiments a wide range of tasks can be used, with tasks sometimes originating from initial design studies, task taxonomies, or other generally qualitative sources.

However, experiments can generally only test a very limited number of tasks due to time constraints and issues surrounding user fatigue and compliance. There are also challenges associated with developing a visualisation system capable of supporting the full set of possible tasks, as well as the lack of an interface with comparable features to test against. It is therefore important that studies do not take a scatter-gun approach to task selection but rather select tasks to be tested based upon tasks that real-world users may seek to carry out with all of the systems being tested. In this way we build on the qualitative user evaluations in Chapter 4 to select the limited set of tasks quantitatively tested in Chapters 5 and 6.

2.5 Experimental Evaluations

Despite the positives of qualitative evaluation and the rich situational understanding they can offer, we focus on quantitative studies here because we feel that the wider generalisability and precision of their results can be used to better inform the design and development process of features of our visualisation system. If a certain feature, for example animation, nearly always performs poorly in controlled experiments then we can use this knowledge as a foundation for our design and feature choices

2.5.1 Readability of Static Graphs

We focus primarily on experiments that involve the comparison of matrices or node-link diagrams to some other representation style, or work that compares the two directly. Via our novel system, introduced in Chapter 3, we support both of these representations, hence our focus.

Ghoniem et al. [118] produce a taxonomy of generic graph topography related tasks and then carry out an experiment to evaluate the readability of matrix-based representations and node-link diagrams. They find that the two methods are complementary; with node-link diagrams working well for small graphs and matrices being more suited to large or dense graphs. A second reported result is that path related tasks are difficult on both representations, and

interaction is required to help perform them. Based on these results, they recommend that matrices be used to aid greater familiarity and, consequently, improve a graph's readability.

Ren et al. [250] compare the readability of node-link diagrams and matrices in a large scale online study. They found that node-link visualisations were associated with higher accuracy and speed than matrix-based representations. However, the completion time gap narrowed throughout the experiment; with matrix performance steadily approaching that of the node-link as participants completed more tasks. This decreasing performance delta supports the idea that matrices are more challenging as an entry point for non-expert users than node-link diagrams; but that as users become more familiar with matrices, their performance equalises.

Okoe et al. [227] also carry out an experiment comparing the same but with large networks typical of those found in real-world situations. They find that node-link diagrams are better than adjacency matrices for questions about network topology, connectivity, and memorability tasks, while adjacency matrices outperform node-link representations for group tasks.

2.5.2 Readability of Dynamic Graphs

Experimental evaluations discuss 'large' node-link diagrams without quantifying what makes these large, and how this definition might differ between users and computers. What is considered complex by a user is very different from the graph considered complex by a supercomputer. Yoghourdjian et al. [344] examine existing dynamic graph visualisation experiments to characterise better the features of node-link diagrams that drive complexity. Saraiya et al. [271] evaluate and rank dynamic visualisation options. Tasks were similar to those chosen by Ghoniem et al. [118] as they were generic on graph topology. They found that overlaying data on graph vertices one time point at a time may lead to more accurate performance for tasks involving the analysis of a graph at a single time point, and also for tasks requiring comparisons between graph vertices for two distinct time points. Single views have an advantage over multiple views on tasks that require topological information. Also, the number of attributes displayed on nodes has a non-trivial influence on the accuracy of responses, whereas the number of visualisations affects the performance time.

Purchase et al. [241] introduce the first of a series of tests on the importance of the mental map in dynamic graphs. The results support the idea that the mental map factor is important in dynamic graph layout. However, they test only small graphs - the largest graph in this experiment had only 20 nodes and 30 edges. Purchase et al. [242] later extend this work, finding that "there is no benefit in a dynamic graph layout algorithm maintaining a mental map

between timeslices when mental map preservation is defined as minimising node movement". A further study by Archambault and Purchase [15] supports the conclusions from the previous mental map work [241, 242] with no significant difference found in terms of response time or error rate when preserving the mental map. In a separate experiment, Archambault et al. [16] find that mental map preservation can be beneficial for certain tasks on a dynamically evolving graph; with results showing that mental map preservation leads to faster completion time for tasks with significantly fewer errors.

Archambault et al. [20] experiment to examine how different maps help improve the readability of dynamic graphs and to find out under which interface they are most useful. The study provides evidence that difference maps can help answer questions relating to large-scale changes in a dynamic graph in terms of changes in the number of edges. They also found that participants strongly preferred difference maps.

Farrugia et al. [102] carry out two experiments, one in a lab and one using Mechanical Turk, to compare the efficacy of animated vs static displays for temporal graphs. Each visual representation has four tasks; two characterising overview level information presentation and two characterising micro-level analytical tasks. Their results show that static representations are generally more effective, particularly in terms of time performance, when compared to animated representations.

Federico and Miksch [104] evaluate two interaction techniques for node-link based dynamic graph visualisations: the adjustment of the layout stability and the highlighting of adjacent nodes and edges. They find that both techniques can increase accuracy, but that this is often at the cost of speed, and that adjacent edge highlighting outclasses the stability adjustment in most cases, except for where very complex tasks are involved.

While some visualisation features have been validated experimentally, and therefore it is good practice to support these features, many more have no empirical evaluation or do not have the statistical power to draw wide-ranging conclusions. Where possible we support experimentally validated features (see Table 2.4 for a full breakdown), particularly layout stability for mental map preservation, but were not able to support every feature for every task combination due to use-case constraints of our users.

Table 2.4: How we apply the literature in Chapter 2.5 to the contributions in this thesis.

Beginning of Table 2.4			
Topic	References	Relationship to this thesis	Chapters this literature is used in
Readability of static node-link diagrams vs matrices	[118, 227, 250]	The results of these works support the idea that node-link diagrams can be well complemented by matrices, but they also highlight that matrices are not generally suitable as replacements for node-link diagrams due to their complexity. They also find that matrices do not perform as well for path following tasks. Based on information from these studies we provide both node link diagrams and matrices in our system, but link the views and use node-link diagrams as an entry point for the ease of use for new users.	Chapter 3

Continuation of Table 2.4

Topic	References	Relationship to this thesis	Chapters this literature is used in
Impact of complexity on readability of dynamic graphs	[118, 271, 344]	There are advantages to overlaying wider information about the graph on graph verticies, and the display of single time points is more useful for tasks involving the graph structure. An increase in the number of visualisations (e.g. as in small multiples) has a negative impact on task performance time.	Chapter 3, Chapter 5, Chapter 6
Mental map preservation	[15, 16, 241, 242]	The findings about the importance of mental map perservation inform and guide our ethos for the layout of all node-link diagrams we use in the thesis. We consider the preservation of the mental map to be the most important feature of any representation that we generate.	Chapter 3, Chapter 5, Chapter 6
Animation vs small multiples	[102]	Small multiples outperforms animation for several dynamic graph tasks; we use this as justification to exclude animation from our novel visualisation system, and also to inform our hypotheses for the two experiments.	Chapter 3, Chapter 5, Chapter 6

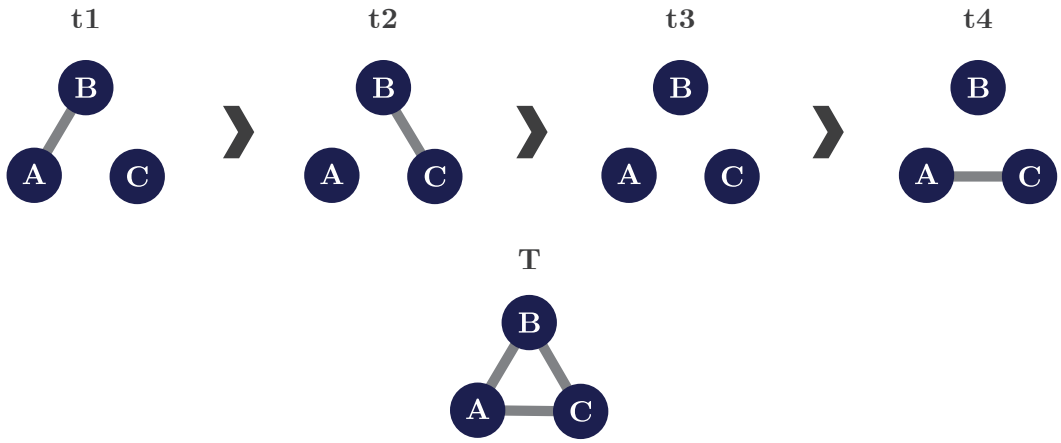


Figure 2.5: An example of how flattening events into a single timeslice loses within timeslice event order.

2.6 Event-Based Visualisation

In this thesis we discuss the concept of a 'timeslice', Figure 2.5 shows the construction of a timeslice from a series of events and how this action results in the loss of information about the order of events within a single timeslice. For this thesis and the broader dynamic graph visualisation community, a timeslice is a fixed period of time which maps to a dynamic graph. Imagine a continuous timeline T which begins at t_y and ends at t_{y+n}. Uniform timeslicing of the timeline T involves binning the entire timeline into equal intervals of time x such that each subsequent time point follows on from the previous time point. For example,

$$T = \{(t_y, t_{y+x}), (t_{y+x}, t_{y+2x}), (t_{y+2x}, t_{y+3x}), \cdots\} \tag{2.12}$$

Timeslicing is widely used in the dynamic graph visualisation community as it is a natural way to think of dynamic data, offers a quick way to reduce the complexity of one of the data dimensions, specifically time, and small multiple representations (see Chapter 2.3.4) are generally built on this concept.

However, when imposed on continuous data, timeslicing causes the loss of information about the order events occur within a timeslice. For example, given a set of 3 fully connected vertices within a timeslice $V = (A, B, C)$ and $E = (AB, AC, BC)$. At $t1$ an edge is created between A and B. At $t2$ an edge is created linking B and C. At $t3$ nothing happens, and at $t4$ there is an edge between A and C. When compressing all of these events into a timeslice T, the result is that the researcher or analyst would have no idea of the order in which edges were formed,

and would have no way of knowing that there was no event at $t3$. This is illustrated in Figure 2.5. If event order is key to understanding a graph, timeslicing means that within-timeslice events are entirely lost.

If we timeslice continuous data it is also challenging to define how many timeslices should be selected; too few timeslices can miss key temporal events and their order, and too many will slow computation and make analysis increasingly challenging. With the increasing availability and decreasing cost of more powerful hardware, we are now able to treat events as individuals, rather than binning them into periods of time. Shneiderman [282] extends the concept of Anscombe's Quartet [14] to apply it to event-based temporal data, clearly demonstrating that, for some continuous event-based data, display in tabular form is insufficient for finding insight.

Event-based analytics deals with many discrete events happening through time, usually in the form of scalar data. In these methods, the events are visualised individually and using aggregation. Aggregations can be used to reduce the visualisation to a specific number of categories [206]. Event-based methods can be useful for novice users who may need assistance to find a starting point for data exploration as visualisations of large datasets can be crowded and difficult to read [92].

Event-based analytics in the visualisation field have been primarily studied and designed for use in health informatics. Users – who are not necessarily visualisation experts – can use the data and the resulting visualisations to uncover patterns and insights into diagnostic treatment events and their subsequent outcomes.

The concept of event-based analytics has also gained traction in the graph mining field and the broader literature but is generally referred to as a temporal graph in other fields [150]. Our focus here is specifically on the visualisation of this data, and we do not examine applications specifically from the graph mining field.

2.6.1 Interfaces

Fisher et al. [107] introduce a tool for event exploration; though the main contribution is more regarding the query language and method than the visualisations, as the latter largely takes the form of standard frequency histograms. They identify the challenge of dealing with events where timing is not definite and the issues that can arise from the arbitrary ordering of event sequence data without proper consideration of the ordering aims.

Similar to other dynamic graph visualisation methods [63,66,82,240], Tominski et al. [303] introduce an event-based visualisation system which is based on the concept of a parallel co-

ordinates visualisation, with brushing on the axes as an interaction method. However, rather than have the parallel coordinates as a set of vertical lines separated by horizontal space, they instead treat each coordinate axis as an edge axis around a shape.

Monroe et al. [210] develop a stacking timeline interface to facilitate the analysis of interval events – events with both start and end time – and the absence of events. They also introduce a novel query method. Later, Monroe et al. find that their original interface [210] becomes so crowded as to be unusable with large datasets and as a result, they introduce a filter-based simplification method [209] to reduce visual clutter and complexity. Madkour et al. [200] also use vertically stacked events along a timeline to display event order, as do Chen et al. [76] to create an optimised summary of their temporal event data.

Ruddle et al. [261] introduce a matrix-style event sequence visualisation with the aim to scale-up event sequence visualisations to millions of records. Wongsuphasawat et al. [335] also use a matrix-style representation to support event sequence analysis, though this could also be considered a modified hybrid tree-matrix visualisation rather than a purely matrix inspired piece of work. Similarly, Krause et al. [180] use a hybrid mixture of treemaps, node-link diagrams, and histograms to facilitate the visual creation of cohorts for medical studies via a tool they name COQUITO.

Malik et al. [202] develop the CoCo tool to compare two cohorts. Their tool integrates statistics with visual analytics and an interface that guides users towards significant cohort differences. Similar to Monroe [209] this is a stacked timeline visualisation, though the work by Malik et al. differs in that it allows users to select event sequence combinations and visualise the associated outcome probability given a period of time.

Shiroi et al. [279] present a space-efficient technique for visualising temporal patterns that facilitates simultaneous visualisation of many event groups. This visualisation uses a circular axis as a clock to position events, with the barycentre of all timestamps associated with a given event-type group used to position groups of events within the circular axis. However, multiple groups of events can have very different temporal patterns but similar timestamp barycentres.

Itoh et al. [156] use stacked 3D layers for Spatio-temporal event visualisation based on data from different physical and social sensors. This is in contrast to previous work on dynamic graphs [157] where data is timesliced, and links are drawn in 3D. Similar to Shi et al. [278], Liu et al. [243] create a system for the egocentric visualisation of dynamic networks. In contrast to Shi et al., however, Liu et al. do not use any three-dimensional representation. Vrotsou et al. [316] introduce an interactive visual data mining approach and a visual inter-

face to facilitate user-centred exploration of data and identification of sequences significant to that user. Like [243, 278] this is an egocentric visualisation system, though this visualisation leverages a clock-style representation as in Shiroi et al. [279] also.

2.6.2 Studies

Ruddle et al. [261] also carry out an experiment to investigate the users' ability to judge event sequence similarity. They find that, in contrast with previous work [336], colour and shape are better for encoding event type than position. However, they also find that participant accuracy for simple sequences is less than 90% and conclude that this indicates that simple visualisation techniques are not necessarily the most effective option.

Zhang et al. [348] set out to evaluate different alignment approaches for temporal event-sequence visualisations. They find that dual-event alignment is more accurate when understanding intermediate events between two sentinel events. However, for understanding the duration between two sentinel events, they find that no sentinel event alignment is both more accurate and significantly faster than other approaches.

2.6.3 Graph Drawing

Simonetto et al. [291] introduce a method which draws the dynamic graph in its entirety, an unusual approach in a field where some aggregation of time is usually necessary from at least a scalability perspective, if not also a usability perspective. They build on the idea of 'worms' as discussed in a standard dynamic graph context [94, 128] but Simonetto et al. use these 'worms' for calculating node position over time; rather than using them as a 3d representation of the data itself. The resulting graph layout and visualisation is readable, but there are scalability constraints – the approach here is limited to relatively small graphs of under 1000 nodes and 10,000 edges.

2.6.4 Summary

Few existing approaches for the visualisation of event-based data use graph data; and of those that do, even fewer attempt to preserve the visualisation of relationships between individuals who are experiencing the events. Simonetto [290, 291] and Liu [197, 243] stand out as the few examples where this takes place. Due to the lack of importance placed on showing relationships between individuals the majority of existing representations are timeline based [76, 200, 202, 209, 210] or have some visual similarity to matrices [261, 335] or treemaps [180].

This thesis can be considered a method for visualising event-based dynamic graphs, but in contrast to previous work on event-based network visualisation [290], we do not attempt to draw the dynamic graph in its entirety; instead, we allow the user to select windows of time and create timeslices containing all events in this window. We also make use primarily of node-link diagrams due to their accessibility for non-expert users [119] and their superior performance for path and sub-group finding tasks [118]. Table 2.5 offers an overview of how the event-based graph drawing literature informed the research presented in this thesis.

Table 2.5: How we apply the literature in Chapter 2.6 to the contributions in this thesis.

Beginning of Table 2.5			
Topic	References	Relationship to this thesis	Chapters this literature is used in
Event-based graph drawing	[197, 243, 290, 291]	We use some of the underlying methods for drawing dynamic graphs to optimise the graph layout that we use in our novel system.	Chapter 3
Event-based visualisations	[76, 200, 202, 209, 210]	We include a timeline-style event-based visualisation in our novel system to facilitate the exploration, discovery, and understanding of event order within timeslices.	Chapter 3
Event-based time selection	[76, 200, 202, 209, 210]	Our new method of time selection – time slicing – can be considered an event-based focus of the more traditional timeline-type dynamic graph visualisations.	Chapter 3, Chapter 5, Chapter 6

2.7 Visualisation of Provenance

In this thesis, we deal with the challenging problem of visualising long in time graphs where the visuals will need some form of transformation by the user. This demanding task becomes even more complicated when the user needs to reproduce analysis steps that involve complex interaction stacking, or where the user wishes to reproduce their analysis on another part of the data while maintaining a view of their previous work. At the end stage of analysis, particularly in complex or lengthy analyses, the user will also typically want to verify the steps they have taken to ensure that their conclusions are correct. A user might also need to explain their analysis process to others so that those new to the analysis can understand what is being shown on the screen. We identified the requirement for interaction provenance as a critical function that our users would require and, as a result, this thesis contributes to the field of interaction provenance.

Provenance information is information about how a particular state has been attained. This information can take the form of interaction provenance, where the information is about the interactions that have taken place with an interface. It can also include data provenance, which includes information about the processes and source(s) that lead to the dataset creation and current representation. Although data provenance is significant, our focus is on the processing and sense-making parts of the data pipeline; hence in this thesis we focus only on the visualisation of interaction provenance.

Xu et al. [339] provide a comprehensive overview of provenance for visual analytics, opting to break down the existing literature into three broad categories: a) work that explains reasons of visualising provenance data, b) work that discusses what provenance data should be encoded and the best ways of encoding, and c) work that analyses provenance data. The survey has a comprehensive focus, but we concentrate only on the visualisation of provenance data generated from user interactions as other topics are out of the scope of this thesis. As a result, we discuss only a subset of work regarding provenance information identified by Xu et al.

Information provenance can be used to better understand a system and the user's analytic process. This data can be passed into systems that then recommend visualisations, and operations on visualisations, based on previous actions [35, 70, 71, 81, 100, 112, 123, 130, 174, 217, 318, 336].

Further usage of provenance data is to repeat or verify previous analysis sessions; an example of this is the undo/redo tool in most software. Callahan et al. [68] introduce VisTrails to

facilitate the usage and creation of provenance dataflows. Other systems also allow this [286], and some extend to the creation of data transformation scripts [167, 295] using the recorded provenance data. Provenance data is also a rich source for the creation of analysis session summaries [74, 124, 140, 205, 340]. These summaries can be restructured [205], used to gain a deeper understanding of collaboration sessions [340], or visualised for story generation [124]. Capturing provenance data is also practical for situations where the analyst needs to understand the user and their reasoning process. This occurs across a variety of fields [91] and can be used to uncover bias and strategies in data exploration [58, 79, 105].

When dealing with past user interaction, it can be useful to display these within the visualisation space. This approach is commonly used in cases where the user selects or draws elements, and then the system searches the data to find matching information [83, 216, 322, 327]. This user selection can be used for data cleaning [167] or the refinement of data queries in specific syntax [106, 151, 216, 288, 319, 332] or a more natural language expression [151, 277].

Rather than visually displaying interaction provenance, however, some systems encode each user interaction as a series of rules [117, 178, 217, 293, 328, 337] which can then be reused or separately analysed. In some cases, domain-specific languages are used in place of formal grammars [77, 166, 260].

A further, complex, approach for the encoding of user interaction is as a graph structure rather than a linear list of temporal events [68, 89, 93, 140]. Using this structure can facilitate the discovery of patterns in a graph structure that would not otherwise be found in a linear list, such as repeated groups of analysis steps, commonly revisited analysis methods, and cliques. In some cases [124] the user directly interacts with the provenance graph to generate a story from their analysis. Other examples of graph-based provenance information include [47, 205, 273]. Shrinivasan et al. [286,287] combine both history and concept graphs to represent user analysis. Users can click on a node in the concept graph and are returned to the corresponding historical analysis step via the coordinated views. SenseMap [222] displays both factual information, with web pages as nodes and visited links as edges, and a subjective "knowledge map" created by the user with information collected during online exploration as nodes and the edges are created by the user (not the visited link) to connect similar or relevant items.

User studies have shown that providing a view to support provenance visualisation can aid the process memory of the user [140, 244, 245]. Several systems have support for interaction histories [68, 93, 304] and report positive results related to insight discovery and exploration recall.

Due to the generally positive results reported for systems with integrated interaction provenance, and the anticipated complex analysis that will be undertaken with our tool, we choose to integrate the visualisation of interaction provenance into the system we introduce in Chapter 3, and detail how previous work on this topic informed our wider design choices in Table 2.6.

Table 2.6: How we apply the literature in Chapter 2.7 to the contributions in this thesis.

Beginning of Table 2.6			
Topic	References	Relationship to this thesis	Chapters this literature is used in
Repeating analysis sessions with provenance information	[68, 74, 124, 140, 205, 205, 286, 340, 340]	Multiple works explore the utility of using provenance information to allow analysis sessions to be repeated or reviewed. We build on this and automatically repeat previous analysis steps whenever a new timeslice is defined in our system, saving analysts a considerable about of time and effort.	Chapter 3, Chapter 4
Display of provenance information	[83, 106, 151, 216, 216, 288, 319, 322, 327, 332]	The display of provenance information can be used to help the user re-trace their steps, or help clear up confusion in cases where analysis has become complex. It can also be used to help a user explain their analysis steps to others. For this reason, and due to the collaborative nature of the wall display, we use the constant display of provenance information in our system.	Chapter 3, Chapter 4

Continuation of Table 2.6			
Topic	References	Relationship to this thesis	Chapters this literature is used in
Experiments related to provenance information	[68, 93, 140, 244, 245, 304]	Previous work finds that provenance visualisation can aid user memory and exploration recall. Informed by these findings, we include the visualisation of provenance information in our system.	Chapter 3, Chapter 4

2.8 Visualisation on Large Displays

Traditional displays have a low number of pixels, placing a fundamental upper bound on the amount of information that can be simultaneously visible on a screen. In the past, larger displays (usually composites of many smaller displays [30, 44, 129, 223, 298]) were, due to technology constraints, the easiest way to increase the available pixel count. This, in turn, increased the ability to display detail or more data items. Increased number of pixels and a physically larger size has been shown to increase performance [86, 301, 302].

Although modern display technology has reached pixel densities that make the argument about pixel counts largely irrelevant (a state-of-the-art 15" display can have as many pixels as an 88" display from just a few years ago), there are still reasons why large displays are desirable for visualisation. First, they enable larger numbers of people to work on the same data [158]. Second, people interacting with large displays seem to benefit from physical navigation [10, 32, 160], which might improve memory and performance [158]. Thirdly, the ability to change the distance to the display easily and naturally by stepping back and forth enables a natural zoom experience and visualisation techniques that would be hard or awkward with personal displays (e.g., [154, 160, 218]). Incidentally, a larger display will also reduce problems with the precision of touch interaction (e.g., the fat finger problem [289]) simply because the content will typically be larger.

Perhaps due to these reasons, research investigating and translating visualisation to large

displays is still very active (e.g., [176, 239]). From the information available in the literature and our own experience, we speculate that large displays are likely to offer the best environment for visualising complex and high-density information such as dynamic networks, particularly when scalability in the time dimension is required. Simultaneously, the specialised nature of this kind of analysis also means that the extra cost of procuring large displays is usually well justified, and many research environments already offer medium to very large displays.

2.8.1 Hardware of Large Displays

The term 'large display' covers numerous hardware configurations, each with their own set of advantages and disadvantages. Previously, CAVE systems surrounded users with at least four projection screens or displays [84], though some approaches used 5 or 6 displays [256]. Given the decreasing costs of LCD panels, it is now reasonable to use these to construct a CAVE system [101], these also have superior screen quality and resolution compared to projected systems.

Multi-monitor desktops are now commonplace and can be run by standard hardware at low cost. Some modern laptops are capable of simultaneously running four external displays with 4k resolution. However, these setups typically suffer from bezel issues [199, 300] – edges of the monitors breaking the continuity of the large display – and there are usability concerns with these setups for multi-tasking [30, 86, 152]. Interactions with multi-monitor desktops are traditionally indirect mouse-pointer style, which is generally considered to be more precise [338].

Tiled LCD panels can also be used in a vertical configuration as a wall display, flat as though a table [181], curved, or in any other positioning required by the users. Tiled LCD panels can have a larger physical size, and a total pixel count well over 100 million pixels is achievable [85,270]. Tiled LCD panels are generally more comfortable to use and cheaper than projectors, and they also take up less space. However, these also suffer from bezel issues [199, 300] affecting the seamless nature of the display and breaking any display of text. Despite this, the ultra-high resolution nature of tiled LCDs can facilitate the visualisation of vast amounts of data in one continuous display, and they can also allow multiple display content to be shown simultaneously on different parts of the display [258].

Projector arrays [1, 108, 116, 272, 299, 306, 331] present several difficulties: they have interaction challenges, lack geometry controls [75, 145, 146, 296], and suffer problems with colour rendering [320] and brightness [201]. However they lack the bezels which are problematic with tiled LCD displays so it is possible to seamlessly integrate multiple tiles [246, 247].

Stereoscopic displays [116, 136, 193, 269] show two sets of pixels for an image, making one set visible to the user's left eye and the other to the right eye. Typically the user is required to wear special glasses or viewing aids to see the 3D effects. We do not consider these for our use-case as network visualisation in 3D is challenging [215].

2.8.2 Visualisation Techniques and Experiments

Andrews et al. [9] define a set of design requirements for visualisation on large, high-resolution, displays; highlighting that it is not sufficient to merely upscale existing visualisation approaches for large displays and that, instead, we have to consider the importance of physical navigation in how the user will approach, perceive, and engage with visualisations on these displays.

Due to the typically larger number of pixels available on large, high-resolution, displays, there is a design choice to be made between showing a greater amount of data space – more overview – or a greater depth in scale – more detail. While there is always a trade-off between overview and detail, a display of high enough resolution can mitigate this by increasing both overview and detail to the point that is reasonable based upon the requirements of the task. However, if the screen displays a large amount of data, consideration must be given to the user's perceptual and cognitive abilities. Ware [325] suggest that a 4000x4000 display is the ultimate display because it is most efficient at matching screen pixels to the 'brain pixels' that interpret the signals sent by photoreceptors in our eyes.

Ball et al. [31] compare a 3840x3072 display to two smaller displays (1560x2048 and 1280x1024). They find that, for finely detailed data, the highest resolution display in combination with physical navigation significantly outperforms the smaller displays with pan and zoom navigation. They also find that users associate the large display with low-stress levels, engender a better sense of confidence when compared to smaller displays, and use more physical navigation for the high-resolution display and more virtual navigation for the lower-resolution displays. Czerwinski et al. [86] measure the productivity benefits of large displays against those provided by standard computer monitors. They observe significant benefits in the use of a prototype, larger, display, in addition to significant favourable user preference and satisfaction with its use over a small display, confirming previous results [31].

Later, Ball et al. [32] build on their findings regarding high-resolution displays and physical navigation. They identify a relationship between display size, user performance time, amount of physical navigation, and virtual navigation. For the spatial visualisation tasks, larger displays

led to more physical navigation, reducing virtual navigation, which offered improved user performance. They conclude that physical navigation is an efficient and valuable interaction technique that reduces dependency on less-efficient virtual navigation and is preferred by users – in situations where either physical or virtual zoom-in navigation could be used to complete the task entirely, participants chose physical navigation 100% of the time. These findings are confirmed by the further work of Jansen et al. [161] who find that overview improves recall, and the combination of overview and user locomotion outperforms all other tested combinations of the stationary, locomotion, peep-hole, and full overview factors.

Bezerianos et al. [43] investigate the perception of visual variables on tiled high-resolution wall-sized displays. Perception accuracy is impacted most when viewers are close to the wall, though the impact differs for each variable of angle, area, and length. A second study finds that a static viewpoint that is far from the screen is both more accurate and more efficient than allowing the user to move freely. For tasks close to a wall display, they recommend either placing important information directly in front of the user or providing the user with an estimation of the distortion effect.

Ruddle et al. [262] set out to experimentally identify situations in which large, high-resolution displays are beneficial. They found that display resolution does not affect the speed of finding targets in densely distributed data, but for sparse target finding a 54 million pixel display was 30% faster than the 2 million or 12 million pixel displays.

Large, high-resolution, displays are also able to facilitate collaboration between users. Jakobsen et al. [158] find that multi-touch wall displays can support different collaboration styles and that their participants could switch fluidly between parallel and joint work.

There seems to be a relative lack of research in the intersection of graphs and large displays. A significant contribution to this area is GION [203], a skeletal animation technique for interacting with large graphs on wall displays. The technique is based on a physical simulation, to enhance the users' ability to interact with the graph visualisation for exploratory analysis. After evaluation, they find that GION produces layouts with less stress and fewer edge crossings. They also find that users prefer GION and that task completion with GION required significantly less mouse movement than the other tested interfaces. Kister et al. [176] combine spatially-aware mobile devices and large wall-sized displays to facilitate the visualisation of graph data using both node-link and matrix based techniques. However, they do only use and design for static network data.

In this thesis, we opt to use a high-resolution vertical display due to the reported user ben-

efits, and the extra available screen real-estate compared to standard desktop-sized displays. This is also an understudied area for dynamic graph visualisation. We chose to use an 84" Microsoft Surface Hub, and a Dell 84" Interactive Touch Monitor. Both displays have a maximum resolution of 4K (3840x2160). See Table 2.7 for an overview of how we apply findings from previous work to the work presented in this thesis.

Table 2.7: How we apply the literature in Chapter 2.8 to the contributions in this thesis.

Beginning of Table 2.7			
Topic	References	Relationship to this thesis	Chapters this literature is used in
Benefits of large displays	[9,31,32,86, 161]	Previous work finds that the increased number of pixels, and increased overall size, available with large displays is able to increase the performance of tasks. They are also able to facilitate collaborative work, and physical navigation is able to improve memory and performance. For these reasons we use large displays in the development of our visualisation system.	Chapter 3, Chapter 4, Chapter 5

Continuation of Table 2.7			
Topic	References	Relationship to this thesis	Chapters this literature is used in
Visualisation on large displays	[43,158,176, 203,262]	Work shows large high resolution displays, in combination with physical navigation, outperform smaller displays which relying on pan and zoom navigation. Users also report improved productivity, and more confidence in their results when using large displays. They also find overview improves recall, and that large displays can support many different collaboration styles. We choose to design, develop, and test our system on a large touch-screen display.	Chapter 3, Chapter 4, Chapter 5

2.8.3 Interaction Techniques for Large High-Resolution Displays

Large, high-resolution, displays offer a wealth of input types. Input is direct when the user's motor actions take place roughly in the same location as the output (e.g., in touch UIs). Input is indirect when there is a spatial separation between the input device (where the user's motor actions occur) and where the visual feedback is provided (e.g., using a mouse to control an on-screen cursor).

For large display interaction, both mouse and touch input have known drawbacks: direct touch requires users to move physically to reach all areas of a large display, although alternative techniques have been proposed for interacting with areas physically out of reach [42,165,173]. In contrast, mouse input allows interaction from a distance but may confine users to a specific location. Also, it can be challenging to keep track of mouse cursors [153], hard to distinguish between multiple cursors, and distracting with the presence of multiple cursors [131].

A large body of work has examined how different interaction types impact on user performance and behaviour for tabletop displays [4, 54, 98, 109, 131, 135, 155, 175, 211, 212, 219, 257, 265, 292, 317, 334], however as we do not use tabletop displays, nor do we develop novel interaction techniques, a discussion of these is out of scope for the thesis.

While the argument could be made that a tabletop is simply a wall display moved to a lower height and laid flat, we find several key differences that invalidate this suggestion. Firstly, when a user is using a wall display they have access to the entire screen by physically moving in only one axis along the front of the screen. When the interaction surface is changed to a tabletop, however, the user must either lean uncomfortably over the length of the tabletop display to reach areas close to other sides; or they must physically move around the table. The former has the potential to greatly exacerbate the distortion effect found by Bezerianos and Isenberg [43], due to the effect of the users eyes becoming closer to the tabletop display but at an oblique angle relative to other elements on the display. The latter mechanism would mean that content on the tabletop display would become sideways or upside down as the user travels around the table, leading to difficulties reading, understanding, or mapping previous content positions. This type of movement would also likely be disruptive to any analysis tasks; while Ball et al. [32] find that physical movement in the context of wall displays has a positive effect on user concentration and understanding there is no work proving or disproving this for tabletop displays. Rotating elements on the screen, as opposed to the entire screen, may mitigate this slightly but the rotation would likely move user targets in at least one dimension; potentially causing frustration and confusion. These key differences in visual field, targeting, and physical interaction mechanism mean that we do not consider tabletop displays to be relevant to the thesis and so discussion of such is omitted.

Knudsen et al. [177] use the results of workshops with target users to identify required support for data interaction and visualisation on large displays. They call attention to the requirement to support sequences of visualisations with backtracking and fluid exploration of alternatives, as well as using the distance between the user and the display to change the visualisation.

Regardless of the type of input, the input device itself demands some level of attention, and McLaughlin et al. [208] identify that a user's task performance can be affected by the match between the input device and the action performed on the interface. They find that indirect input suits tasks involving precise movement or repetitive movement, whereas direct input is better suited to pointing tasks and ballistic movements.

Riehmann et al. [251] produce a set of interactions for working with scatterplot matrices on wall-sized displays. To address the challenges of long swipes on blunt surfaces and uncomfortable postures to reach far regions of the screen, they use fling-based gestures to centralise the user interaction.

Specifically for wall displays, some work has been carried out to evaluate the feasibility of mid-air gestures as an interaction technique [321] and describe the type of mid-air gestures that might be preferred by users [333]. Further work has identified the feasibility of these techniques for collaborative situations [163, 164, 266], and the user perception of territory in the same [329]. Ntoa et al. [224] use mid-air gestures for navigation and orientation in a 3d map overview of a data centre and support other basic visualisation tools with the same gesture set. Zhai et al. [347] combine mid-air gestures with a wall-sized touch screen display and provide sets of interactions for both near mode, where the user can physically touch the screen, and far mode, where the user steps away from the screen. As highlighted by Bezerianos et al. [43], the perception of variables on a large screen display changes depending on the position of the viewer in space. Variable perception is most accurate from a far, static, viewing position. Providing gestures that facilitate interaction through multiple view angles and positions could be one way to address this problem. Positional perspective distortion can also be corrected through perspective compensation [220], though this is generally not appropriate for multiple users. Nancel et al. [221] use a very-high-resolution wall-sized display to develop and empirically evaluate techniques for mid-air pan-and-zoom interactions. They find that linear gestures and a high level of gesture guidance significantly improve performance. Rateau et al. [248] also combine mid-air gestures with tactile interactions to support continuous drag-and-drop over a large area on a wall display, addressing the issue of dragging items over a large space, and ending or initiating a dragging action at a point out of reach.

A specific subset of mid-air interactions combine wall-sized displays with mobile devices controlled by the user. Langner et al. [185] use spatially-aware mobile devices in front of wall-sized displays for data exploration and navigation. With this method, the user can navigate data sets by walking and moving a mobile device within the interaction space. However, collaborative situations in these systems can be challenging to design and build for due to difficulties differentiating between individual users. Kister et al. [176] combine mobile devices and wall-sized displays specifically for graph exploration and manipulation tasks and report positive results. This is one of the few works that exists in the intersection between graph visualisation and large displays. Supporting these results, Paay et al. [228] find that swiping on

a mobile device in control of a wall-sized display is the most accurate and efficient method, closely followed by swinging the mobile device. One suggestion is that the ability to keep the pointer steady on the large display, unaffected by concurrent gestures or body movements used to complete the interaction, is essential for designing effective cross-device interaction with large displays.

Others have introduced and evaluated [165] ray-pointing techniques, for example with laser pointers, for interaction with large and distant displays. Badam et al. [29] propose a hybrid technique of proxemics — the distance, orientation, and movement of each user – and explicit mid-air gestures. They find that, for collaboration and navigation, users prefer implicit interaction through proxemics. For more direct actions, such as cancelling an action, users preferred mid-air gestures. Liu et al. [195, 196] introduce a set of collaborative gestures that combine input from multiple users to manipulate content, facilitate data exchange, and support communication. An experimental evaluation finds that collaborative gestures reduce physical fatigue and facilitate collaboration compared with traditional multi-touch gestures.

Jakobsen et al. [159] compare mouse vs direct touch interaction for collaborative situations. They find that mouse input was quicker and more accurate, but that the single point of control led to conflicts within some participant groups and that non-controlling participants would move closer to the screen, blocking the view of the user in control of the mouse. In contrast, they found that dragging objects using touch was slower, required extensive physical movement, and participants had to navigate around each other. However, for the touch condition only an optical display was tested, rather than capacitive or other touch systems, which may have impacted these results – not all touch screens are equal. Despite this, participants generally consider that touch interaction is superior to mouse input for collaborative situations. Sambrooks et al. [268] go further and compare mouse, touch, and gestural interactions. They find that touch and mouse interactions are quicker and more accurate than gestural, though this may be related to deficiencies of the implementation. While touch mostly performed comparably with mouse input, there were situations where performance suffered with smaller targets due to occlusion and the impreciseness of a finger compared to a mouse cursor.

As this thesis is primarily focused on data visualisation, we were careful not to introduce noise into the research via a novel or generally new interaction technique that might have had a secondary effect on our users, participants, and interface. For this reason, we primarily aim to use direct touch interaction with our large-screen vertical displays; while there are issues with touch interaction they are generally related to very large wall-sized displays, as opposed to the

relatively small – in large display terms – 84" screens that we had available.

To understand the potential impact of touch vs mouse interaction we also carry out two studies, Chapters 5 and 6, one using direct touch input and the other using indirect mouse interaction, though we do not statistically compare between these.

2.9 Discussion

Despite a large body of work examining the best method for visualising dynamic network data, there is no consensus on the hardware, interaction, layout, representations, or even the most suitable dimension for mapping time. Task taxonomies are now relatively complete for dynamic network data visualisation [3, 12, 13, 172, 285], yet no dynamic graph visualisation method is suitable for all identified tasks. However, this is reflective more of the diversity and complexity of the identified tasks than a lack of work in the area. We conjecture that no single interface could do a good job of supporting all 127 tasks identified by Kerracher [172], for example.

For time-to-space representations, the vast majority of research chooses timeslicing as a mitigation strategy for challenges surrounding the scalability of the time dimension. However, constant timeslicing - for example binning the whole graph into set periods of 5 hours regardless of event density - does not take the underlying dataset into account, meaning that events are not necessarily equally distributed between timeslices. A result of this is that some time windows will have many events (and therefore be cluttered) while others are relatively, or entirely, devoid of events.

Previous work has also identified the importance of providing statistical information in conjunction with visualisations of dynamic network data [232]. However, often there are no statistical measures provided in current methods, or the user is required to go hunting for them via menus and filters [341]. Having the user break away from their analysis task to search for more information within the representation can lead to increased frustration [230], less efficient analysis [230], and interruptions are associated with a more significant number of mistakes [134].

In this thesis, we deal with the challenging issue of visualising long in time dynamic network data. Due to the diversity of this data, and the considerable number of possible analysis tasks [13, 172], we will need to support some state retrieval of the visualisation at different, user-initiated, transformation points. Others have shown that providing a view to support provenance visualisation can aid the process memory of the user [140, 244, 245]. Several sys-

tems have support for interaction histories [68, 93, 304] and report positive results related to insight discovery and exploration recall.

In visualisation terms, rather than in feature terms, the consensus is generally that time-to-space mappings are superior for most dynamic graph tasks [18, 102]. Within the time-to-space mappings, however, there are also variations regarding the exact visualisation of the graph. These can be broken down to node-link diagrams, adjacency matrices, or some hybrid of the two. Generally, node-link diagrams work well for small or sparse graphs [119] and matrices are more suited to large or dense graphs [119]. Further work finds that node-link based representations are quicker and more accurate than matrices for non-expert users [250] and that node-link diagrams outperform matrices for all tasks except those requiring group identification [227].

Within the time-to-space mapping group, however, there is not much variation in the treatment or display of time – the majority of previous work relies on continuous uniform timeslicing despite the issues with the loss of within timeslice events [291], the general problems of computing a good graph layout with layout stability across the whole time period [290], and the further problems of comparing distant points in time.

A large body of work has also described the impact of the 'mental map' - layout stability - when applied to node-link diagrams that use dynamic data. Some results find that preserving the mental map only in terms of minimising node movement has no impact [15, 242], while others contradict this for certain dynamic graph tasks [16], reporting that mental map preservation leads to faster completion time with significantly fewer errors. However, even in cases where mental map preservation makes little performance difference, the user may feel that it helps [16]. No one has tested how different methods of exploring or controlling time might impact user performance for some graph tasks; instead, previous work focuses mainly on graph layout or the intricacies of specific representations.

Further, there is little work for the visualisation of dynamic graph data – or even static graph data – on large-screen displays, despite reports of improved recall and performance associated with physical navigation [31, 32, 86]. Large displays can also facilitate collaborative working [158] and offer the ability to change the distance to the display easily and naturally by stepping back and forth, enabling a natural zoom experience that would be hard or awkward with personal displays [161].

We choose to use large, vertically-mounted, touch-screen displays for our visualisation system due to the reported user benefits and the extra available screen real-estate compared to

a standard personal display. Surprisingly, there is a general lack of research regarding dynamic graph visualisation on these display types. We use an 84" touch screen vertically-mounted display with a maximum resolution of 4K (3840x2160). Effectively using the space provided by these devices gives additional space to compare distant points in the time series simultaneously and supports multiple visualisations of the dataset in the session, facilitating collaboration and novel dynamic graph visualisation methods. We also introduce a novel treatment of time and timeslicing. Rather than setting an arbitrary timeslice size or limit we instead hand control of timeslicing to the user, and also allow them to only select regions of interest (rather than the whole graph) to reduce visual complexity and ensure that their attention is only consumed by regions relevant to the analysis. This choice also facilitates the comparison of points that are distant in time in a way that other visualisation systems do not currently do.

While many experiments have described the differences between time-to-time and time-to-space mappings, and the visualisation types within these broad categories, none have tested non-uniform user-defined timeslices, and so we have no understanding of the options for time control and treatment outside of continuous time (e.g. animation) or uniform continuous timeslicing (e.g. small multiples). We first carry out an experiment using our large touch screen vertical display to describe the differences between these three methods of time interaction. Later, to verify the generalisability of our method, we carry out a further experiment with mouse pointer input on personal displays.

We expected that our system users would carry out complex and sophisticated analysis and that they would like to retrace their steps or visually understand how they reached a certain analysis point. For this reason, we also support interaction provenance and the continuous display of statistical network measures. Few existing visualisation systems for dynamic network data support both of these features simultaneously.

We describe these contributions in the following four chapters.

Chapter 3 introduces our novel visualisation system with support for the continuous display of statistical measures, constant visualisation of interaction provenance, an online stable graph layout to support the mental map, and our new method of time treatment.

Chapter 4 describes the two evaluation sessions of our system by our target users, and we highlight insights that they would have not typically had the ability to uncover given their original reliance on statistical summaries of data.

Chapter 5 details the experiment that we carried out to quantify the difference between the visualisation of continuous-time (i.e. animation), uniform continuous timeslicing (i.e.

small multiples), and non-uniform user-defined timeslicing on a large touch screen vertically mounted display.

Chapter 6 details the remote experiment that we carried out to verify the generalisability of our treatment of time on a standard display with mouse pointer input.

Chapter 3

Dynamic Network Plaid

Contents

Long in time dynamic graphs that span long periods of time (months) with respect to their temporal resolution (seconds) are particularly difficult to analyse. If a sizeable temporal distance separates two or more interesting views of the data at different points in the analysis session, they become difficult to compare. Time-to-space mappings (multiple snapshots of the graph on a timeline) do not have enough screen space to show relevant time points simultaneously for comparison. Time-to-time mappings (animations) require the user to remember two frames of the animation and interactively navigate through all states of the animation between them. These two main methods for dynamic graph analysis are significantly limited when analysing such datasets.

When analysing dynamic network data, not only would we like to compare distant points in time, but we would like to compare the dataset as it is being visualised now to some previous state in the session. We also might want to repeat some analysis, automatically or otherwise, on a different part of the data. Provenance information helps the analyst understand how a particular insight was obtained. Effective visualisation of this information can help compare two or more interesting views of the data at different points in the session.

Our goal is to visualise both long in time dynamic graphs and provenance information simultaneously in an intuitive manner. In order to accomplish this task, we use large displays. We designed for and deployed to a physically large display because the additional physical space enables physical navigability and the display of large amounts of information. Effectively using the space provided by these devices gives additional space to simultaneously compare distant points in the time series and multiple visualisations of the dataset in the session, supporting collaboration and new dynamic graph visualisation methods.

We consider these goals in designing a visualisation technique that we apply first to a group of collaborative researchers in public health. The specific datasets consist of Instagram comments and images of self-harming behaviour. The dataset spans more than one month and has a temporal resolution down to the second. Our users are interested in understanding how communication between actors occurs and if one actor in the network causes other actors to post (network contagion).

In this chapter, we present Dynamic Network Plaid (DNP), a system and technique that addresses the challenges of analysing long in time dynamic networks along with provenance information. Our technique is the first to present dynamic graphs of this type together with provenance in an effective way. The technique is validated through an experiment with public health researchers on a dataset of self-harming behaviour. The findings from these sessions are

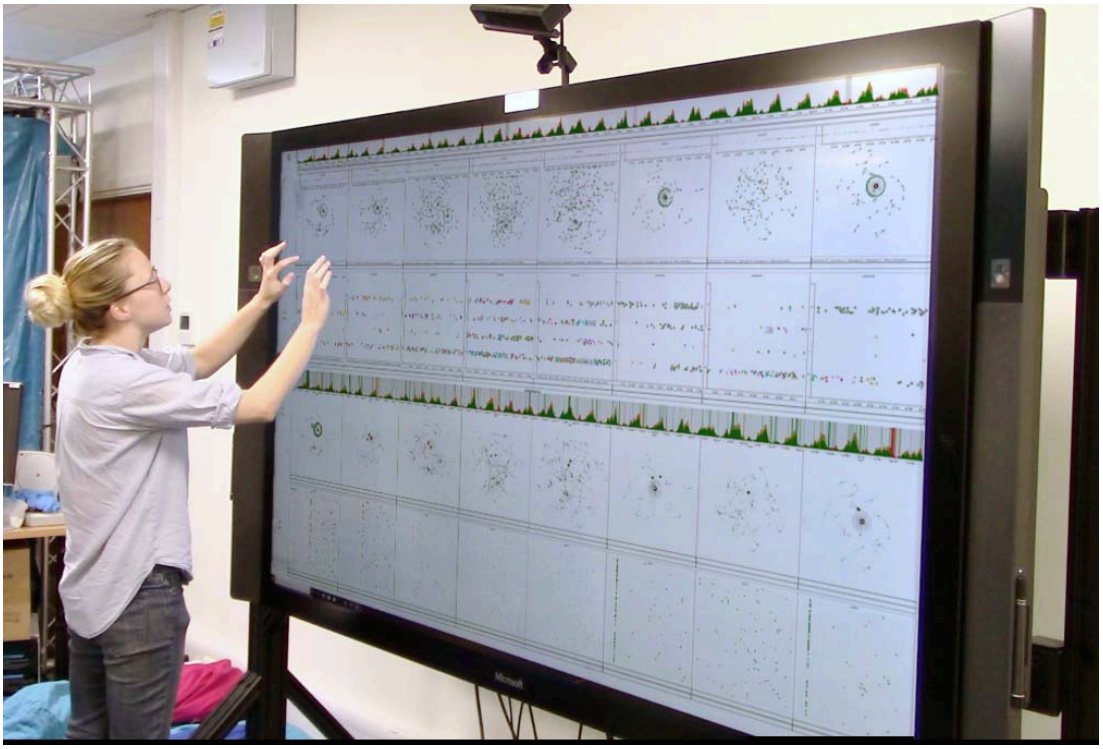


Figure 3.1: Dynamic Network Plaid (DNP) running on a Microsoft Surface Hub 84".

discussed in Chapter 4.

3.1 Example Dataset

Dynamic Network Plaid was designed with the analysis of long in time dynamic networks, networks that span a long time interval when compared to their temporal resolution, in mind. With this goal in mind, we choose an application in public health to enable researchers in this domain to collaborate on such datasets. In this section, we describe a specific *self-harm* dataset for four main reasons: a) it provides an excellent real-world example of the challenging data that the tool can help analyse; b) this is the dataset that we used to develop, test, and validate the tool; c) it is useful to illustrate the design and features of the data concretely, and; d) it is a socially relevant dataset.

The data was collected and rated by Brown et al. [59] for mental health research purposes. The dataset describes interactions between people on the photo and video-sharing social net-

working service *Instagram*[1]. The interactions are centred on non-suicidal self-injury behaviour (e.g., self-harm in the form of cutting one's body), which are shared on the platform in the form of pictures of the harm, and can be commented on or "liked" by other members of the social network. This data is important since it can help characterise self-harm behaviour and determine whether it is socially contagious.

The data was collected from Instagram between 31 March 2016 and 30 April 2016 and tracks interactions centred around images tagged with a self-harm related hashtag set, in the German language. The dataset captures interactions between actors representing Instagram accounts that either posted self-harm images (tagged with specific hashtags–1,154 accounts did this) or commented/liked these images (12,541 accounts did this). Each interaction between nodes is recorded as an edge between a source and a target node. Edges have unique identifiers and are associated to their occurrence time (with second precision), do not have a duration (i.e., are considered atomic, only measured at the time of posting), and are tagged with wound grade. Wound grade is a measure of self-harm severity between 0 and 3 assigned by an expert in the area. The analysed dataset does not contain the original images. Images coded by public health experts are encoded in the dataset as self-referential edges of the posting node. The self-referential edge is assigned a wound grade of 3 if the image shows severe self-harm and a wound grade of 1 if light self-harm. A wound grade of zero means that the image did not show self-harm but still had relevant tagging. Comments and "likes" of an image are shown as links of wound grade zero, where the origin of the link is the commenter account, and the destination is the account that posted the commented image. The dataset has 41,073 edges (2,826 of them are posted images), occurring at 40,487 distinct time points (an average of 1,370 interactions per day).

[1] `https://www.instagram.com`

Table 3.1: An example of the structure of the edge list of the Instagram NSSI data. The column explaining the row event is included here for clarity and readability purposes, and would not normally be present in an edge list.

Beginning of Table 3.1					
ID	**Source**	**Target**	**Event Time**	**Wound Grade**	**Explanation of row event**
1	person A	person B	1262304060	0	Person A has liked or commented on an image posted by Person B
2	person A	person C	1262305207	0	Person A has liked or commented on an image posted by Person C
3	person C	person C	1262305933	1	Person C has posted a photo with a wound grade of 1
4	person A	person A	1262307479	2	Person A has posted a photo with a wound grade of 2
5	person D	person B	1262315020	0	Person D has liked or commented on an image posted by Person B
6	person D	person D	1262316943	3	Person D has posted a photo with a wound grade of 3
7	person E	person B	1262317573	0	Person E has liked or commented on an image posted by Person B

Continuation of Table 3.1

ID	Source	Target	Event Time	Wound Grade	Explanation of row event
8	person E	person D	1262319857	0	Person E has liked or commented on an image posted by Person D
9	person B	person D	1262321865	0	Person B has liked or commented on an image posted by Person D

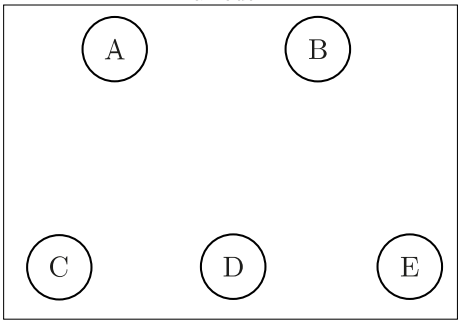
This dataset presents multiple and varied challenges (CH) for data analysis which are also representative of other large dynamic network datasets (i.e., network data of long time span and relatively large numbers of nodes and interactions). Practitioners typically start their analysis by calculating global numerical statistics. This is a necessary step but provides little access to unexpected or interesting occurrences in this data, since there might be much variance in which actors are dominant and how they behave throughout the time span (CH1). Research practitioners use popular analysis tools (from this study [59] and our own experience) and often try to visualise their data, but creating useful visualisations of the data (node-link diagrams are popular) can be fiddly and very time consuming (CH2). When they visualise this kind of data, they often have to produce intermediate data outputs and sometimes draw the data with a force-directed algorithm. If the results are unsatisfactory, they adjust parameters and repeat the process, which results in delayed feedback because of non-interactive workflows (CH3). Even when they produce visualisations, these tend to be global views, but these become tangled and uninformative due to a large number of nodes and interactions (CH4). The natural solution is to create timeslices, but this means, especially if the slicing has to be done programmatically, that significant patterns in network events can easily be missed if they are distributed throughout the data set at different resolutions (CH5– [290]). Aggregate node-link diagrams across the full timespan of the dataset flatten time so that it becomes difficult to notice more subtle interactions where the timing is vital (CH6–e.g., a ping-pong-style interchange of comments).

Also, several common challenges of information visualisation apply, such as finding appropriate interaction (CH7– [342]), effectively representing the connection between multiple representations (CH8– [280]), maintaining awareness of what processing the data being vi-

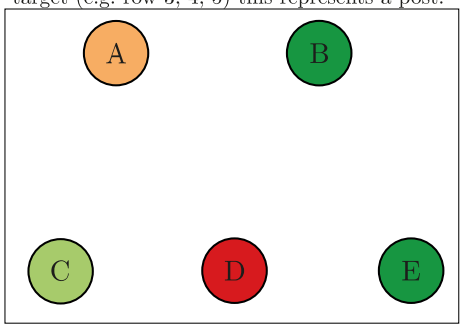
id	source	target	event time	wound grade
1	person A	person B	1262304060	0
2	person A	person C	1262305207	0
3	person C	person C	1262305933	1
4	person A	person A	1262307479	2
5	person D	person B	1262315020	0
6	person D	person D	1262316943	3
7	person E	person B	1262317573	0
8	person E	person D	1262319857	0
9	person B	person D	1262321865	0

wound grade	node colour
0	
1	
2	
3	

Step 1: Each source and target person becomes a node.



Step 2: Nodes are given colours according to the wound grade of their most recent post. Where a row of the edge list has the same source and target (e.g. row 3, 4, 5) this represents a post.



Step 3: Edges are drawn between nodes using the edge list. Each row in the edge list becomes an edge. Arrow heads are used to indication which node is the source and which node is the target. Edges are coloured here in the same colour as the corresponding row in the table for clarity.

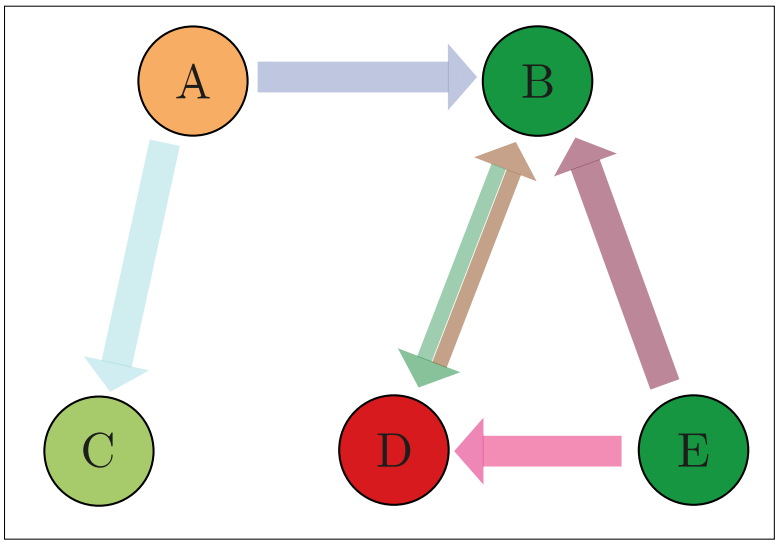


Figure 3.2: These steps explain how an edge list of interactions (see Table 3.1 for explanations of the rows of the edge list) is transformed into a node-link diagram. As we have a source and a target the edges are directional, and the arrows on the edges shown in Step 3 indicate which node is the source and which is the target.

sualised has undergone (provenance, CH9, [68, 88]), and helping users navigate a potentially complex interface (CH10– [324]).

3.2 Functional Requirements

Using our knowledge of the challenges that users of this data frequently encounter, we translate them to functional requirements to aid the design and development of the Plaid.

CH1 Aggregate network statistics do not provide a clear entry-point to the data.

- FR1 Provide a clearly accessible entry point to the data in a visual format.
- FR2 Show a general overview of the dataset across all time, before the user has made any analysis decisions.
- FR3 Facilitate different views for the entry point to the data.

CH2 The visualisation of this type of data can be time-consuming, complex, and requires an amount of expert knowledge and judgement.

- FR4 Remove the mental cost of visualisation from the user.
- FR5 Offer multiple representations of the data that do not require the user to have any input into layout or representation type.
- FR6 Ensure that representations offered by the system are not visually misleading.
- FR7 Representations included with the system should be usable with all data sets
- FR8 System representations need to be relatively easy to interpret for non-expert users.

CH3 Refining node-link diagrams can delay the workflow without immediate visual feedback.

- FR9 Create node-link diagrams without any input from the user in terms of layout.
- FR10 Provide immediate visual feedback during the gravity iteration of the node-link diagram layout algorithm.
- FR11 Node-link diagram layout should take less than 10 seconds per representation to avoid workflow delays.

CH4 It is difficult to find good layouts for node-link diagrams when drawing large temporal network data.

FR12 The system needs to maintain the mental map of the node-link diagram, and so any layout should prioritise this.

FR13 As our expert users are primarily interested in interaction clusters, not 'lone wolves', the system should prioritise clusters in the layout process.

CH5 Timeslicing can miss significant network events if they are not equally distributed through the dataset.

FR14 The system should not programmatically or equally timeslice the data, instead the user should be able to timeslice the data in way they want.

FR15 Users should be given views that enable them to discover significant network events via visual analysis.

CH6 Aggregate node-link diagrams will lose the order of events within a dataset or timeslice.

FR16 The system needs to include representations that can show events that are happening within user-defined timeslices.

FR17 Alternative representations should be used in conjunction with the flattened node-link diagrams to ensure that information is not lost.

CH7 It is challenging to find the appropriate interaction for a visualisation.

FR18 The system should provide standard and easily understood methods of interaction.

FR19 The system also needs to offer a method for interacting with time, as well as with the representations and the interface as a whole.

CH8 Effectively representing the connection between multiple representations can be complex and can lead to interface overcrowding or user confusion.

FR20 The system needs to be able to show where user-selected elements exist across all representations.

FR21 Cross-representation highlighting should be as simple as possible and easily removed when it is no longer required by the user.

FR22 Users should be able to save occurrences of cross-representation highlighting for later reference.

CH9 Maintaining awareness of what processing the data being visualised has undergone can be difficult.

FR23 The system should show interaction provenance information at all times, where possible.

FR24 It should also support users in repeating their analysis steps as much as possible.

CH10 Users often need help to navigate a complex interface.

FR25 The system should use standard and common interaction and navigation techniques as much as possible, to reduce learning overheads.

FR26 The interface should not provide large areas of 'dead space' in which the users can get lost.

FR27 The visual interaction provenance and new representations should be cohesively combined, preventing user disorientation.

3.3 Design Goals and Principles

This thesis's overarching goal is to facilitate the exploratory analysis of these notoriously difficult to analyse and visualise dynamic networks. As we have argued in the previous section, this kind of data presents specific challenges to visualisation due to their structure and the potentially long periods of time during which networks change.

To address the challenges and functional requirements enumerated in the previous section, we made two a priori decisions and selected a set of design principles. The first a priori decision (D1) was to design for large displays. Displays that are physically large and have sufficient resolution offer the opportunity to display large amounts of data and simultaneously support physical navigation in space for individuals and groups of people [31, 32]. Although physical size and number of pixels are not completely independent, one key consideration is that dense resolutions do not necessarily support the visibility of visual elements if the screen is medium or small. We believe large displays can be valuable for dynamic network data by partially addressing CH4 (FR12, FR13).

The second a priori decision (D2) was not to try to show the full data in a single view and instead apply a small multiple approach [37, 311] (addresses CH4 [FR12, FR13], CH5 [FR14, FR15], CH6 [FR16, FR17], and CH8 [FR20-FR22]) and allow the timeslices to be created

interactively (addresses CH1 [FR1-FR3], CH3 [FR9-FR11], CH5 [FR14, FR15], and CH6 [FR16, FR17]). This decision is based on our own experience, observation and workflow of analysis of dynamic network data, and fits with the large display paradigm [346]. We consider the following design principles:

DP1.–*Provide Stable Global Spatial Mappings* Although it is good to give analysts flexibility, we believe that in an exploration of complex data it is essential to keep mappings stable to avoid analysts getting confused about the representations (CH4 [FR12, FR13], CH8 [FR20-FR22], and CH10 [FR25-FR27]).

DP2.–*Prioritise Provenance* We aim at visualising interaction provenance information whenever possible (CH2 [FR4, FR5], CH8 [FR22], CH9 [FR23, FR24], and CH10 [FR26, FR27]). In exploration it is often important to know how a particular insight has been obtained [88, 244], and this is particularly relevant if the work is collaborative [96].

DP3.–*Enable Consistent Cross-Representation Highlighting* Dynamic network analysis can involve recognising how the same node or edge connects through time. Since we use an interactive small multiples approach (D2), we need to track elements or groups throughout multiple representations whenever possible (CH8 [FR20-FR22], CH10 [FR25-FR26]). Note that DP3 and DP2 mutually reinforce each other, since provenance information allows the establishment of cross-representation links across different stages of the processing of sections of data.

DP4.–*Familiarity* Whenever possible, to reduce learning overheads, we aim to use workflows and interface elements that are familiar to non-expert users (CH2 [FR6-FR8], CH7 [FR18, FR19]).

3.3.1 Methodological Approach

Our design process was principled, iterative, and evidence-based. Our starting point was identifying the specific problems of the analysis of this kind of data, which arose from our previous collaboration with public health researchers. In particular, a group of researchers were interested in looking at a self-harm dataset (described in Section 3.1). We got acquainted with the researchers' current approaches and created the prototype through short iterative cycles of

design, implementation, and critique. We presented and studied the prototype with the target scientists in an open-ended observation, from which we obtained important insights and change suggestions. After another redesign and re-implementation period, including some new features, we carried out another observation session and made a final set of changes. The observation/evaluation sessions are described in Chapter 4 and use interviews and qualitative coding of the researchers' interaction with the system.

3.4 The Dynamic Network Plaid

The general shape of the Dynamic Network Plaid is structured across columns and rows of small multiple representations, interleaved by thinner rows (this structure resembles that of a plaid or tartan fabric, hence the name). The key distinction and guiding principle for the interface is that the vertical space dimension represents data transformations by the users (e.g., filtering, change of representation, timeslicing) (FR23, FR24, FR26, FR27), whereas horizontal space represents dataset time (left to right) (FR2, FR8, FR18, FR19). This mapping is consistent for the whole interface and at any point of time, which enforces DP1 to allow analysts to navigate and interpret the space in a simple way, without having to keep track themselves of which representations correspond to which parts of the data (FR26, FR27). Because user operations on the data always generate new rows underneath, this also helps to navigate provenance information (DP2 [FR1-FR3, FR9-FR17, FR20-FR22]). An example view of the interface after some timeslicing, filtering and representation operations appears in Figures 3.11, 3.3, and 3.4. We further describe the different elements vertically, starting with the narrow rows (timelines), continuing with the different types of chequered rows (vignette rows), and then describing the filtering, selection and brushing mechanisms, to finish with a description of various interface features.

3.4.1 Timelines

Timelines are area line graphs that aggregate the number of edges per unit time, and that run continuously from the left screen border to the right screen border, always representing the full data time span (see Figure 3.3.A). When the interface starts, the first and only visualisation element on the screen is the *main timeline* (Figure 3.5. Top). The main timeline provides an appropriate overview of the full time span. The different colours of the stream represent the different values of the main attribute of the link. In our case, there are four colours which

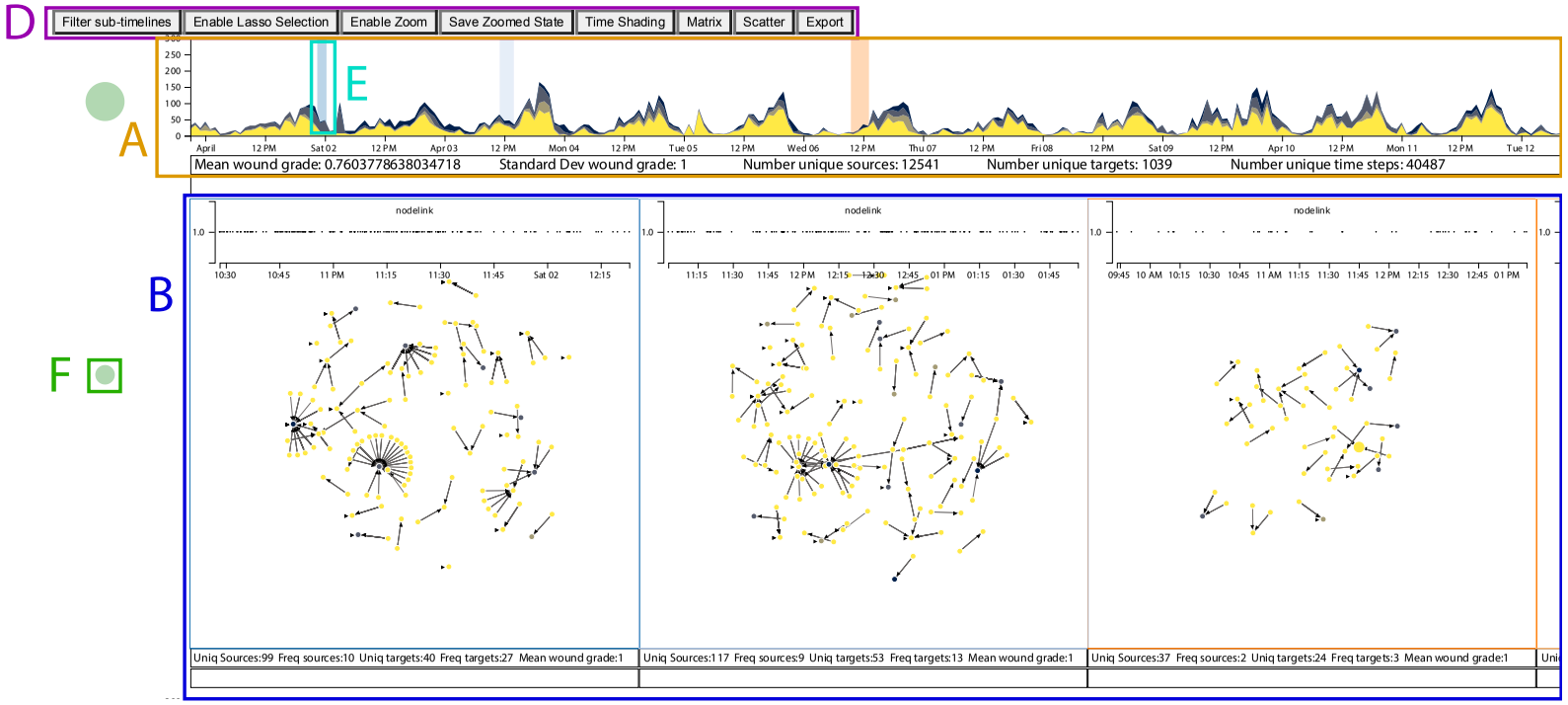


Figure 3.3: This is the top part of the DNP interface, we have split the labelled parts of the top and bottom of the interface for readability purposes. Element A is the primary activity timeline that is used as the gateway to the analysis and an entry point to the data. In this case the user has defined eight timeslices – the coloured rectangles on the main timeline, one labelled element E – and these trigger the creation of the vignettes showing node link diagrams, labelled element B. Element D is the modal menu, which is used to activate certain modes (e.g. zooming or lasso selection) and to trigger the creation of alternative representations. Element F allows the user to collapse unwanted rows and hide them from view.

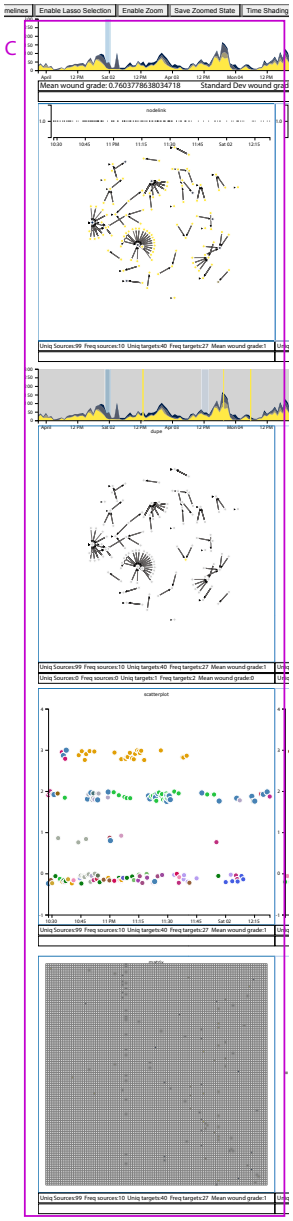


Figure 3.4: A zoomed-in illustration of the column-wise provenance encoding of the DNP. In this case the interface has undergone several interactions. First, the user has defined a timeslice on the primary activity timeline (the blue box), which has triggered the creation of a node-link representation of the defined time period (row 1). Next, the user has used the modal menu (see Figure 3.3, element D) to select a matrix representation (row 4). After this, the user has used the modal menu again to choose a scatterplot representation (row 3). Then the user has activated the lasso selection, again using the modal menu, and clicked and dragged in one of their vignettes to select a group of nodes. Using this selection the user has then opted to save it as a new row (row 2). As we choose to keep the most recently created row immediately under the top set of node-link representations (DP2, DP3) the oldest operation is always at the bottom of the provenance column.

correspond with the four wound severity grades of interactions (from 0 to 3 they go from yellow to dark blue also decreasing in brightness). This scale draws from the cividis colour map [225], which is optimised for the general population, inclusive of viewers with non-typical colour vision. The main timeline provides an appropriate overview of activity across the full timespan. In our experience, this provides the desired entry point to analysts, who usually want to then zoom into details at specific periods of time (usually with unusually large amounts of activity) [280], fulfilling FR1-FR3, and FR15.

Two primary operations are possible on the main timeslice. First, we can filter for a specific event type (e.g., only events of wound grade 3). This creates another timeline (Figure 3.5.Bottom), meeting FR1 and FR3. The second and most important operation is to select timeslices. When the analyst "brushes" the timeline horizontally (i.e., performs a tap-and-drag motion), this creates a timeslice (FR14, FR15, FR19, FR25). The slice is represented in the timeline as a section with a background colour (Figure 3.3.E). This slice can also be dragged to a different location in the timeline, or its size changed by dragging its edges. We allow up to eight slices in our current configuration, which can contain and overlap each other. Each timeslice is assigned a different colour.

Notably, the timeslices are always the same for all interface elements and timelines. This means that elements in the same column always correspond to the same period of time (DP1, FR23, FR24, FR26, FR27) and that changing the span covered by a slice will propagate the changes vertically to all the rows (for that column) (FR24). Underneath each timeslice, two narrow rows contain selected statistics of all nodes represented in the timeline (top) and only the selected nodes (bottom) (FR2, FR5, FR8). The selection mechanism is described below in Section 3.4.3. Example statistics supported are average links per node, average wound grade, and the number of unique sources and targets.

3.4.2 Vignette Rows

Creating timeslices automatically creates a first vignette row. The default vignette row shows square-framed, force-directed node-link diagrams for each of the timeslices (Figure 3.3.B, Figure 3.6 shows two vignettes in more detail), fulfilling FR4-FR13. We call each of these square small multiples a *vignette*. Each vignette corresponds to a timeslice, and the timeslice representation and the vignette are visually linked by the colour of the background of the slice and of the frame, respectively, meeting FR20 and FR21. Vignettes appear in the order in which the corresponding timeslices are created. This allows analysts to create a sequence of vignettes that

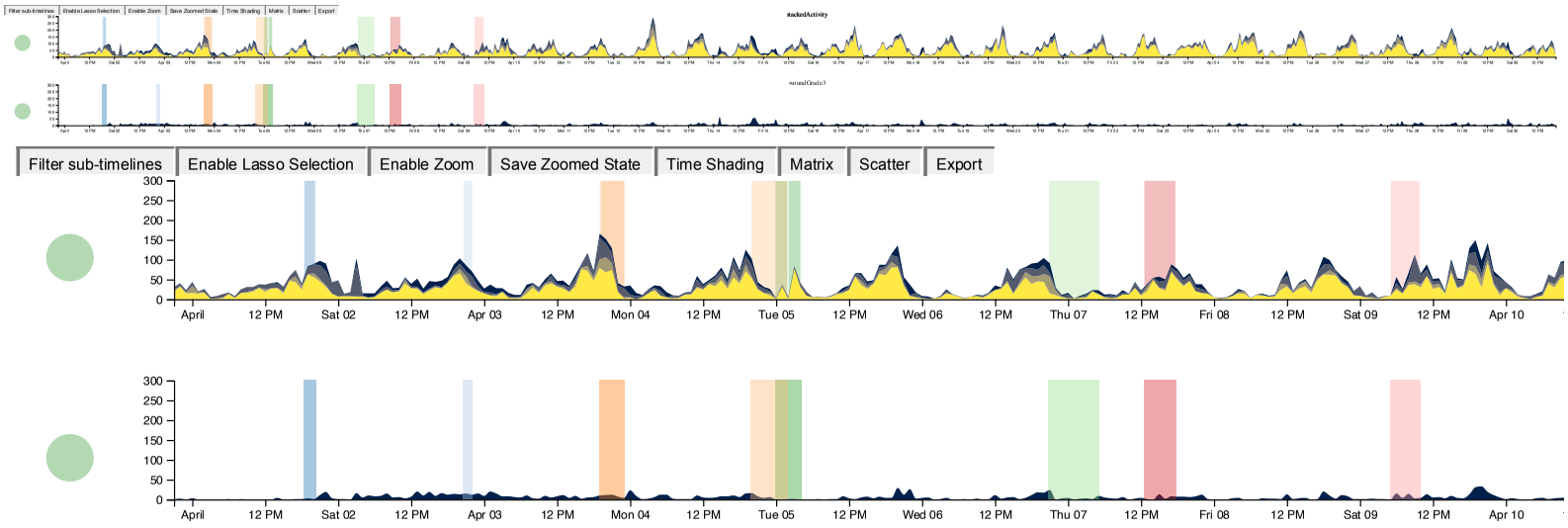


Figure 3.5: An example of the way the interface displays the main timeline (top) and a derived filtered timeline (bottom). The top figure displays the full timelines, and the bottom is a zoomed in version to better display the timeslices and how they cascade over the main and all derived filtered timelines. The derived filtered timeline in these figures is displaying only the activity where the wound grade is level 3 – i.e. only the times at which images of wound grade 3 were posted. Timelines in DNP always cover the full horizontal space and represent the full timespan of the data as in the top image. The coloured rectangles on both timelines are user-defined timeslices, we duplicate timeslice positioning across both timelines for ease-of-use where the user is focusing on the derived filtered timeline for analysis purposes.

are unordered in time, which could be considered a transgression of our DP1. However, the price to pay for enforcing the data-based temporal ordering of vignettes would be that analysts can only look at two vignettes contiguously if there was no intermediate timeslice between them and that vignettes would not be usable as a type of lens on the dynamic data by dragging the timeslice around the timeline while observing changes in the dynamic graph (FR14, FR15). Our analysts often performed this operation, and we found it quite compelling (an example of leveraging DP3).

All vignettes have a *subtimeline* (Figure 3.6. top) which shows how events by the different nodes are distributed within the timespan of the vignette (i.e., its timeslice) (FR16, FR17). This timeline highlights interaction by offsetting any nodes that post upwards from the horizontal line and any nodes that like/comment down from the horizontal line (FR20, FR21).

The two thin rows at the bottom of each vignette show the globally selected statistics, as applied only to the timeslice (top) and the same statistics but applied only to selected nodes (see Section 3.4.3) (FR2, FR5, FR8).

Most of the area of each vignette is occupied by the main representation. In the default vignette row, this is a force-directed diagram as shown in Figure 3.6, where each node is coloured according to the grade of its latest interaction within the timeslice. However, three additional types of vignette rows offer different visual representations: time-shaded node-link diagrams, matrices and scatterplots (FR4-FR8).

Time-Shaded Node-Link Diagrams These are node-link diagrams drawn with a force-directed algorithm where the node's colour (grey to black) encodes the time of the last interaction in the timeslice (see Figure 3.7). This representation was meant to help when subtle time-based interaction takes place (CH6, FR16, FR17, FR19).

Matrices Matrices are useful for visualising direct connections [6] and dense components such as community structures [118]. In an adjacency matrix, the rows and columns are nodes and a square of the matrix is filled if an edge exists between the two nodes (Figure 3.8). The colour indicates the wound grade of the target node. Nodes can be ordered by name, wound grade, and community[2]. By including this we further address FR4-FR8.

[2]Community membership is calculated globally over the entire dataset using Infomap [259] the first time the dataset is loaded into the system. From human-centred [187] and metric [184] perspectives, Infomap has been shown to be effective.

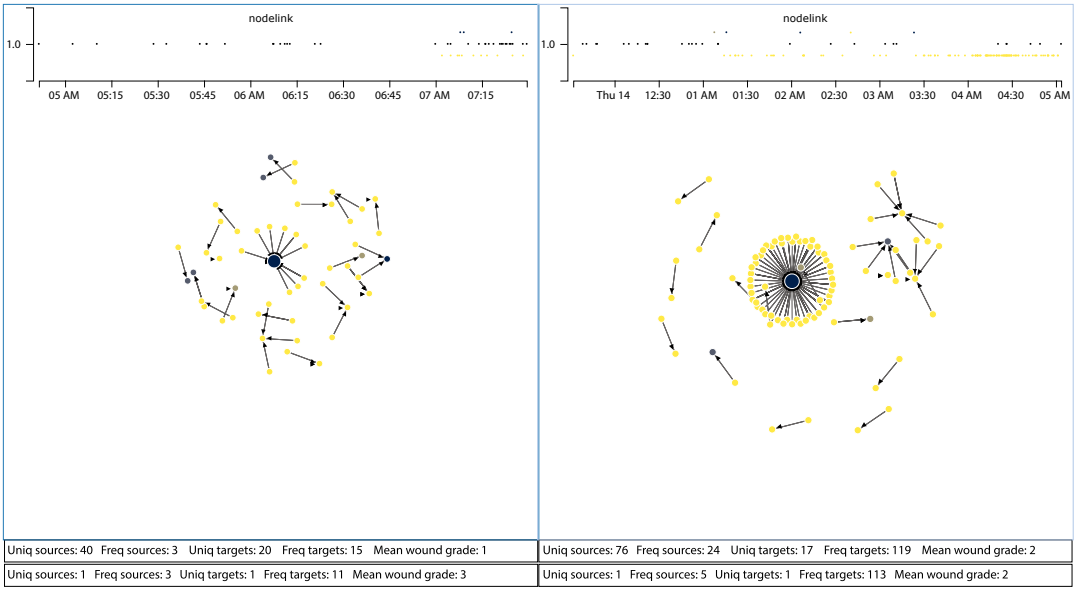


Figure 3.6: Two node-link diagram vignettes with a node of wound grade 3 selected in the left-hand vignette. In the right-hand vignette we can see that the selected node also has a wound grade of 3, meaning that the selected user posted images in both of the defined timeslices. Arrows are used on edges to indicate the source and target of each event. The within-vignette timelines can be seen at the top of each vignette, where every point corresponds to an event. Due to the node selection operation we see within timeslice highlighting by offset in the vignette timelines – the selected node posts 3 images of wound grade 3 within the time period covered first timeslice (corresponding to the left-hand vignette), shown by the three raised points towards the right-hand side of the left vignette (07 AM onwards). The colour of these points maps to wound grade 3. Further, we see multiple points below the central line of the vignette timeline. Using the offset we know that these are source nodes, i.e. people that have interacted with images posted by the selected person. Given that the first offset source node is prior to the first post of the target node within the selected time, we know that the selected node also posted before our selected time window. We also observe a clear, but limited, pattern of behaviour for the selected node – they receive some interactions (likes or comments), then post, then receive further interactions, and then post again; all in a relatively short period of time. We also see a similar pattern in the vignette timeline on the right-hand side vignette, though in this timeslice the selected node also posts an image of wound grade 0 (i.e. no self-injury present). The statistical measures appended to the bottom of each vignette also give information about the timeslice as a whole (the top box), and only the selected node (the bottom box).

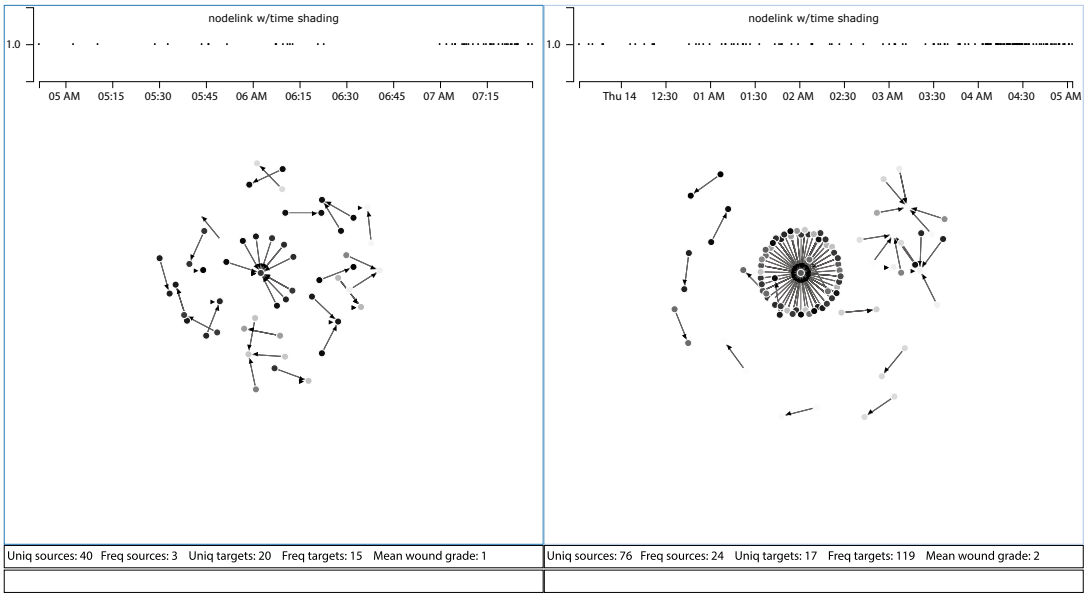


Figure 3.7: The two vignettes from Figure 3.6 with the time-shaded representation option selected. In this representation node colour is derived from the time of the most recent event that a node partook in within a timeslice. Events that are closest to the start of the timeline are coloured white, events at the end of the time period are coloured black, and intermediate events use a continuous grey scale. In the right-hand vignette there is also an edge with seemingly no nodes displayed (if the reader scans vertically down from the central cluster), and we can see that there is one event at the very start of the vignette timeline. Therefore we know that this event is the one corresponding to this edge as both nodes are coloured white.

Scatterplots This representation does not necessarily consider network structure but provides an important perspective of the multivariate data. The scatterplot view allows custom mapping of the vertical and horizontal dimensions, as well as the size of the dots (see Figure 3.9). Each dot represents a link, and available attributes to map are time, wound grade, and community. Community is an offline calculation of groups of nodes that are tightly linked together that can be used in network research [184] (FR4-FR8).

For provenance information visualisation (DP2) reasons, vignette rows do not *change* representations (FR23-FR27). Instead, new rows appear below the current one with the new representation if one of the corresponding buttons in the menu is pressed (see Figure 3.3.D), allowing the user to trace how they arrived at a particular view (FR23, FR27). The interface grows vertically; if the page height is exceeded the analyst is able to scroll up and down the page. To avoid clutter, there is also a mechanism to "minimise" vignette rows by pressing a round button on the left end of the row (see Figure 3.3.F) (FR18). Minimising a vignette row

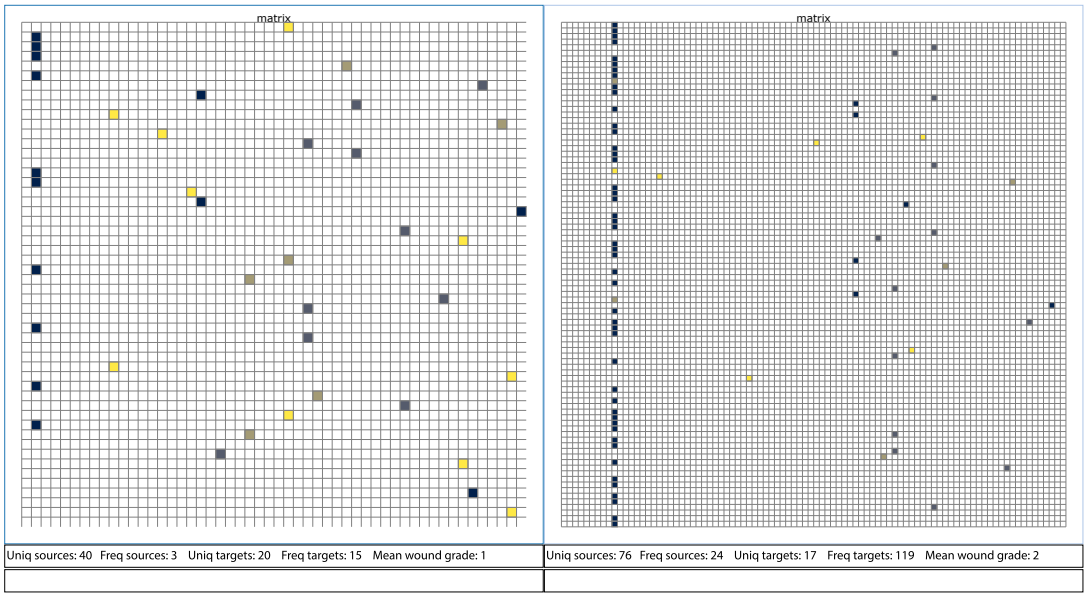


Figure 3.8: The two vignettes from Figure 3.6 with the adjacency matrix representation option selected. Each person in the timeslice appears once on the x-axis and once on the y-axis, with a coloured square representing an interaction between the two corresponding people (a like or a comment). Squares for interactions are coloured by the wound grade of the *target*, i.e. the wound grade of the image that is receiving the interaction, and the default ordering for the matrix is by node/person ID. Users can reorder the matrix by wound grade or by global community membership using the drop down box on the top left of the left-hand vignette. From the left-hand vignette we can see that one image of wound grade 3 receives a lot of interactions (likes/comments) within the selected time period, represented in the matrix as a column with many coloured squares. We can also generally observe that images with a wound grade of 1-3 (i.e. images showing any amount of self-injury) receive more interactions than images showing no injury with a wound grade of zero. From the right-hand vignette we see that one person in the selected time period receives the majority of interaction, and that this interaction is nearly always with their image(s) of wound grade three, as shown by the column with many coloured rows.

makes it appear as a thin horizontal line in the plaid. Rows can also be deleted if no longer needed (FR18). Any hide or delete operations do not cascade down the timelines; hence, these operations are not part of our partial interaction provenance model.

3.4.3 Zooming, Selection and Filtering

Vignettes can be zoomed in by using a reverse pinch movement on the vignette of interest, which offers the ability to closely examine the network structure in a part of the graph (CH7, CH8, FR18, FR20, FR25). A zoom operation creates a new row of vignettes that preserves provenance information (DP2, FR22-FR24). The zooming operation can be considered a type of filtering dependent on network structure (i.e., only nodes in a particular region).

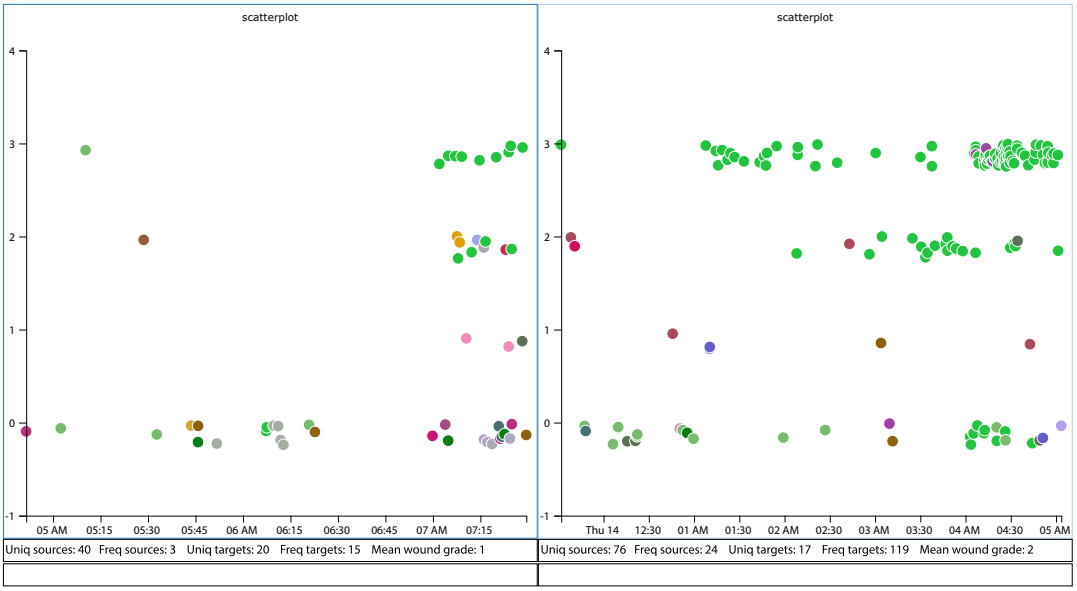


Figure 3.9: The two vignettes from Figure 3.6 with the scatterplot representation option selected. Rather than encoding individual people as points on the scatterplot, we instead encode edges. Edges are either a like/comment or a post (a self-referential edge, i.e. an event with the same source and target). Points are coloured by global community membership by default, but the user can change the items encoded on the x-axis, y-axis, the point colour, and the point size by using the drop-down menus at the top of the vignettes. In the right-hand vignette we see that clusters of interactions take place between people who belong to the same global community, with very little interactions outside of their community. This could indicate that an action from one community member is likely to trigger an action from further members.

Sometimes it is essential to be able to track the same nodes throughout the timeline. However, it can be challenging to represent every actor in a distinctive and recognisable way when thousands of them can be active at potentially any point in the timeline. We address this through selection and linking (DP3, FR15, FR18-FR22). Analysts can select a single node on any given vignette by tapping on it. The node will be highlighted in all of the vignettes where it is present by doubling the size of the selected node (dot in the scatterplots and cell in the matrices). In the timeline view, the node is represented as a vertical line (see Figure 3.11, row 2).

Groups of nodes can be selected with a lasso tool (see Figure 3.10), which highlights all the nodes pertaining to that group in all vignettes except matrix and scatterplot vignettes (the exception is because highlighting many elements in matrix cells makes the matrix hard to read – in matrices with many cells, the squares are too small to show colour or size changes clearly). If the selected set is of particular interest to the analyst, it is possible to press the *selection filter* button, which creates a new row with the selection (FR20-FR22).

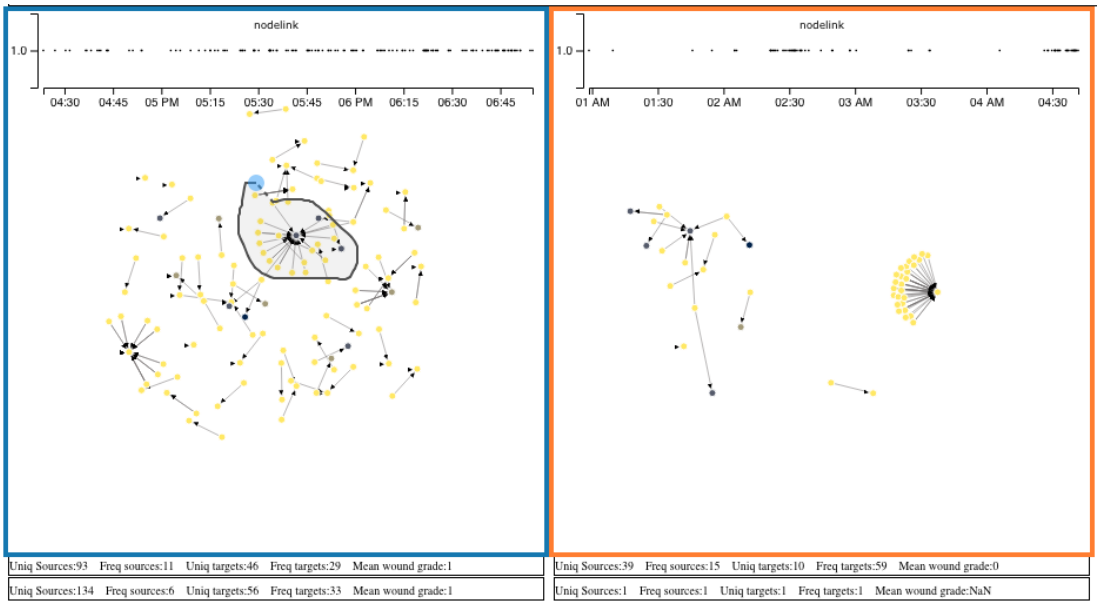


Figure 3.10: The two vignettes from Figure 3.6 showing how the lasso tool is used to select a group of nodes. Users clicked and dragged to define the lasso area. A previous selection had taken place prior to this so the second row of statistical measures has been populated.

Selecting a node, or group of nodes, also causes a change in each vignette activity timeline. Dots where the selected node is the source raise above the centre line and are coloured according to the wound grade of the interaction. Dots where the selected node is the target move below the line. In some cases, this can show turn-taking behaviour and ensures that network evolution events are not lost through the flattening of the timeslice into a node-link diagram (FR16-FR19).

3.4.4 Stability of the Drawing

A stable drawing of a dynamic network supports the mental map of the user in a way that helps them retrieve particular nodes and paths in the dynamic graph as it evolves [17, 18]. Thus, supporting a stable representation across vignettes is a critical factor for effective visualisation.

As our dynamic graph is long in time compared to its temporal resolution, drawing the entire dataset beforehand is prohibitive. Thus, we compute a stable drawing to support the mental map on the fly as vignettes are added to the display. When a new vignette v is added, we compute what nodes are present in the specified time interval. For any node in v already existing in a vignette that has been previously created, it is pinned to the coordinates that have been already calculated. If the node has not previously appeared in a vignette, its position

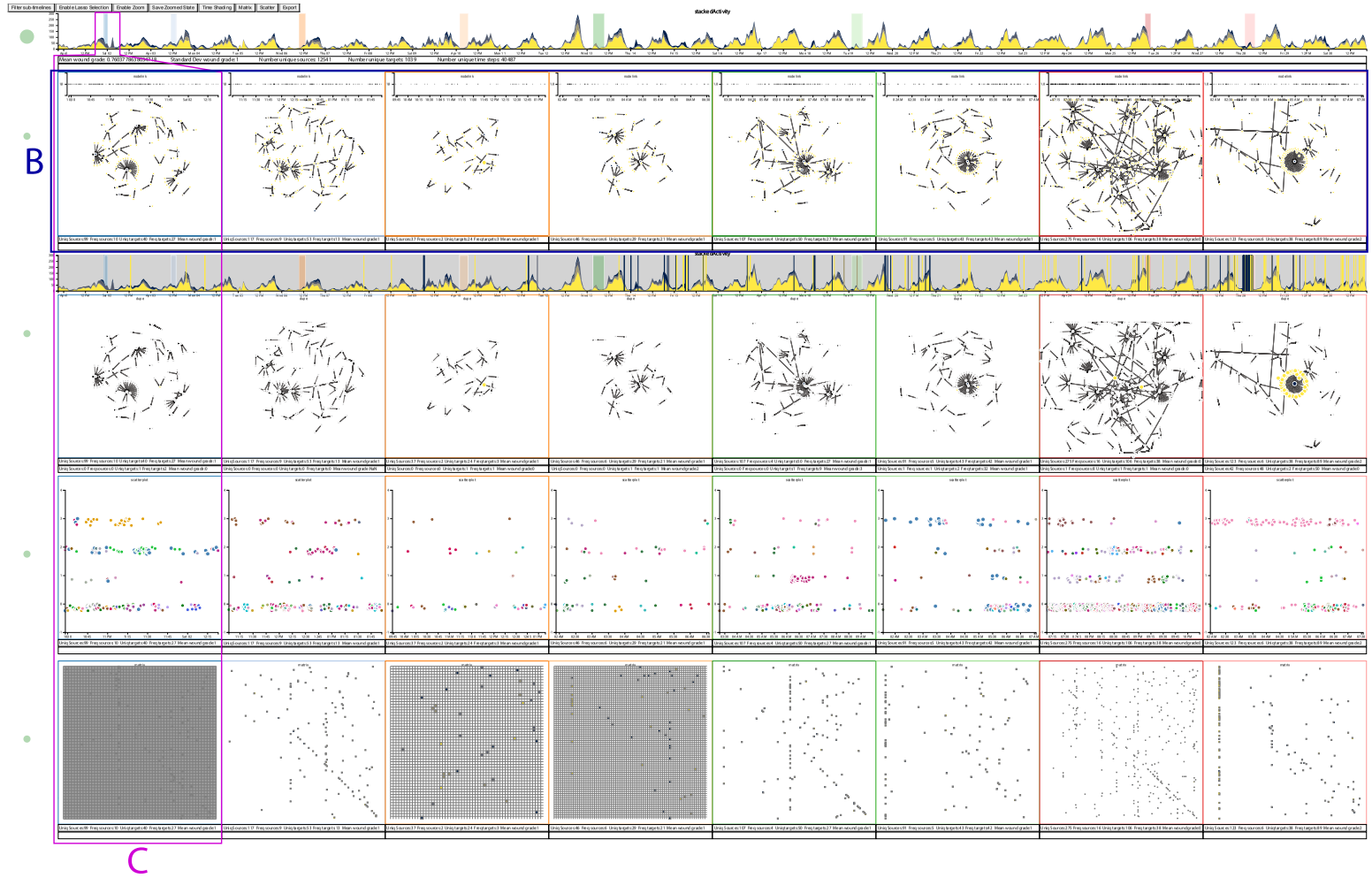


Figure 3.11: This is an example of the DNP interface after the user has carried out several analysis steps. In this case the user has defined eight timeslices – the coloured rectangles on the main timeline – and these trigger the creation of the vignettes showing node link diagrams, labelled element B. The modal menu has been used to define two alternative representations of scatterplot (row 3 in the diagram) and matrix (row 4 in the diagram). The user has used the lasso selection tool to select a group of nodes in vignette 8 of element B. This selection operation displays lines on the main timeline, showing where the selected users also appear; with lines coloured by the wound grade of the occurrence at the corresponding time. The second row is a clone operation of this group selection, ensuring that the elements resulting from this operation persist, either for use in later analysis or as a general reference for the user. Element C displays how interactions cascade in a column-wise manner, with a timeslice mapping to a vignette, and all other displays and representations for this timeslice being placed within the column.

is computed using a force-directed algorithm taking into account the pinned nodes and other non-pinned nodes, meeting FR9-FR13.

3.4.5 Other Facilities

DNP also allows for data export in a format that is compatible with R. After the user has defined timeslices, they can export only the data that appears in these timeslices for statistical analysis.

3.4.6 Design and Feature Summary

The summary of the features of the DNP with how they fulfil requirements and address our identified challenges is shown in Table 3.2.

Table 3.2: Summary of the features of the DNP and how they fulfil our design principles, the challenges associated with the analysis of the type of data that we design for, and the enumerated functional requirements.

Beginning of Table 3.2			
DNP feature	**Design principles**	**Associated challenges**	**Fulfilled functional requirements**
Mapping of vertical space to data transformation by users	DP1	CH9, CH10	FR23, FR24, FR26, FR27
Mapping of horizontal space to dataset time	DP1	CH1, CH2, CH7	FR2, FR8, FR18, FR19
Consistent mapping across the whole interface	DP1	CH10	FR26, FR27
Display of provenance information	DP1, DP2	CH1, CH3, CH4, CH5, CH6, CH8	FR1 - FR3, FR9 - FR17, FR20 - FR22
Timelines as an entry point to the data	DP3, DP4	CH1, CH5	FR1 - FR3, FR15
Ability for the user to filter timelines by data attribute	DP1, DP4	CH1	FR1, FR3
Allowing users to define their own time selections	DP4	CH5, CH7, CH10	FR14, FR15, FR19, FR25
Columns always correspond to timeslices	DP1	CH9, CH10	FR23, FR24, FR26, FR27

Continuation of Table 3.2

DNP feature	Design principles	Associated challenges	Fulfilled functional requirements
Timeslice modification cascades dataset change down the associated column	DP1, DP2, DP3	CH9	FR24
Display of numerical statistics for the dataset within the representations	DP4	CH1, CH2	FR2, FR5, FR8
Timeslice creation automatically leads to node-link diagram creation	DP1	CH2, CH3, CH4	FR4 - FR13
Node-link diagrams have a stable drawing to facilitate analysis	DP1	CH3, CH4	FR9-FR13
Using colour to link vignettes to timeslices	DP3, DP4	CH8	FR20, FR21
Sub-timelines within vignettes to show within-timeslice event order	DP1, DP3	CH6	FR16, FR17
Sub-timelines respond to user selection operations to provide more information	DP3	CH6, CH7, CH8	FR16, FR19 - FR21
Users can select alternative representations of the data	DP1, DP4	CH2, CH6, CH7	FR4 - FR8, FR16, FR17, FR19
Vignette rows do not change representations	DP1, DP2	CH9, CH10	FR23 - FR27
Users can hide and discard un-needed rows	DP4	CH7, CH8	FR18, FR21

Continuation of Table 3.2			
DNP feature	**Design principles**	**Associated challenges**	**Fulfilled functional requirements**
Vignette zooming as a filtering mechanism	DP2, DP3, DP4	CH7, CH8, CH9, CH10	FR18, FR20, FR22 - FR25
Selection and linking across multiple representations	DP3	CH5, CH7, CH8	FR15, FR18 - FR22
Users can persist selection operations to a new row	DP2, DP4	CH8	FR20 - FR22

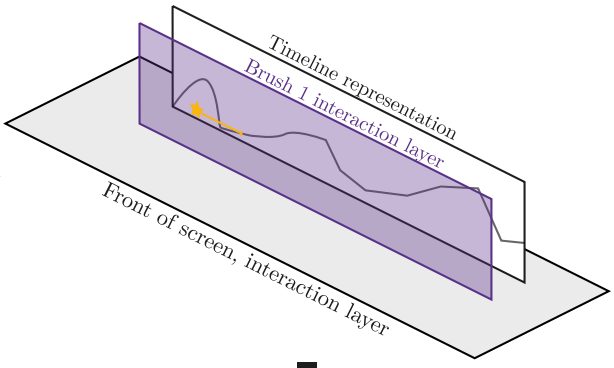
3.5 Implementation

DNP was developed using a customised version of the D3.js library [48]. As we wanted to facilitate the creation of multiple views via brushing, it was necessary to modify the library's underlying code. While it is certainly possible to handle multiple brushes in native D3, the solution is neither clean nor elegant and is also particularly esoteric. To follow general good practice conventions of programming, we instead decided to modify the library.

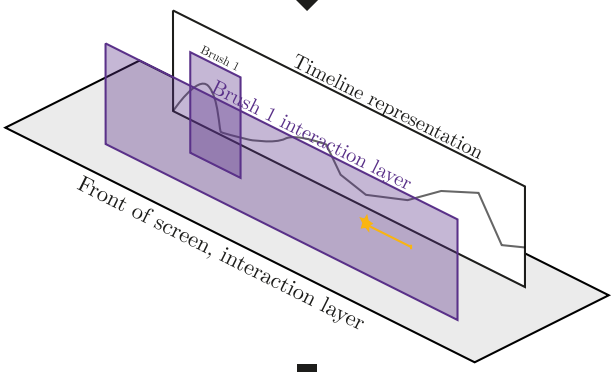
In the typical case for creating a brush in D3, i.e. when the user only needs to define a single brush, a function layer is bound to the Document Object Model (DOM) of the browser. It is this function layer that listens for, triggers, and handles brush creation events. This layer also makes it possible to move and resize a single brush. An example of the DOM structure, in this case, is shown in Figure 3.12. However, it is not natively possible to overlay multiple brush function layers, and even if D3 baked-in support for this, it would not be advisable. Adding a new brush function layer on top of an existing layer would essentially 'freeze' all underlying brushes, as illustrated in Figure 3.13.

We needed our system to facilitate complete analysis cascades and analysis reproduction, and a vital component of these features was defining another brush or moving an existing brush after multiple interaction steps. This new or modified brush would then 'catch' all existing elements and interaction steps and automatically regenerate the full analysis tree, including all views, for the new or modified timeslice. For these reasons, we did not feel that freezing or

Step 1: The user clicks and drags on the brush interaction layer.



Step 2: The brush is created. The user now wants to explore a different part of the timeline, so they click and drag at another position on the brush interaction layer.



Step 3: The original brush is moved to the new position defined by the user. The previous brush position is lost. There is no way to have more than one brush to explore this timeline.

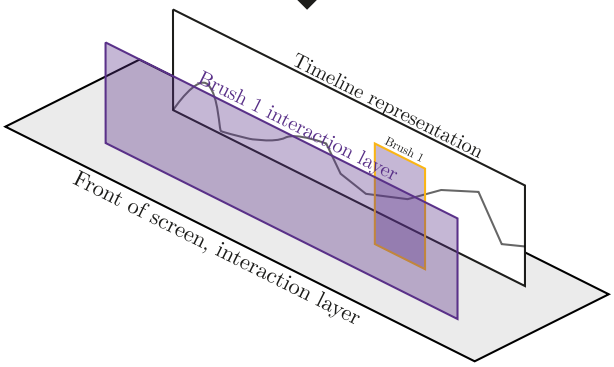


Figure 3.12: This is the normal layout and brush operation process in D3 where only a single brush is required. Notice how the visual representation of the brush selection (labelled brush 1) is sandwiched between the timeline (the visual representation of the temporal data) and the actual interaction layer for brush definition. The brush interaction layers shown here are invisible to the user within the system, and are only assigned a colour in this figure for readability purposes.

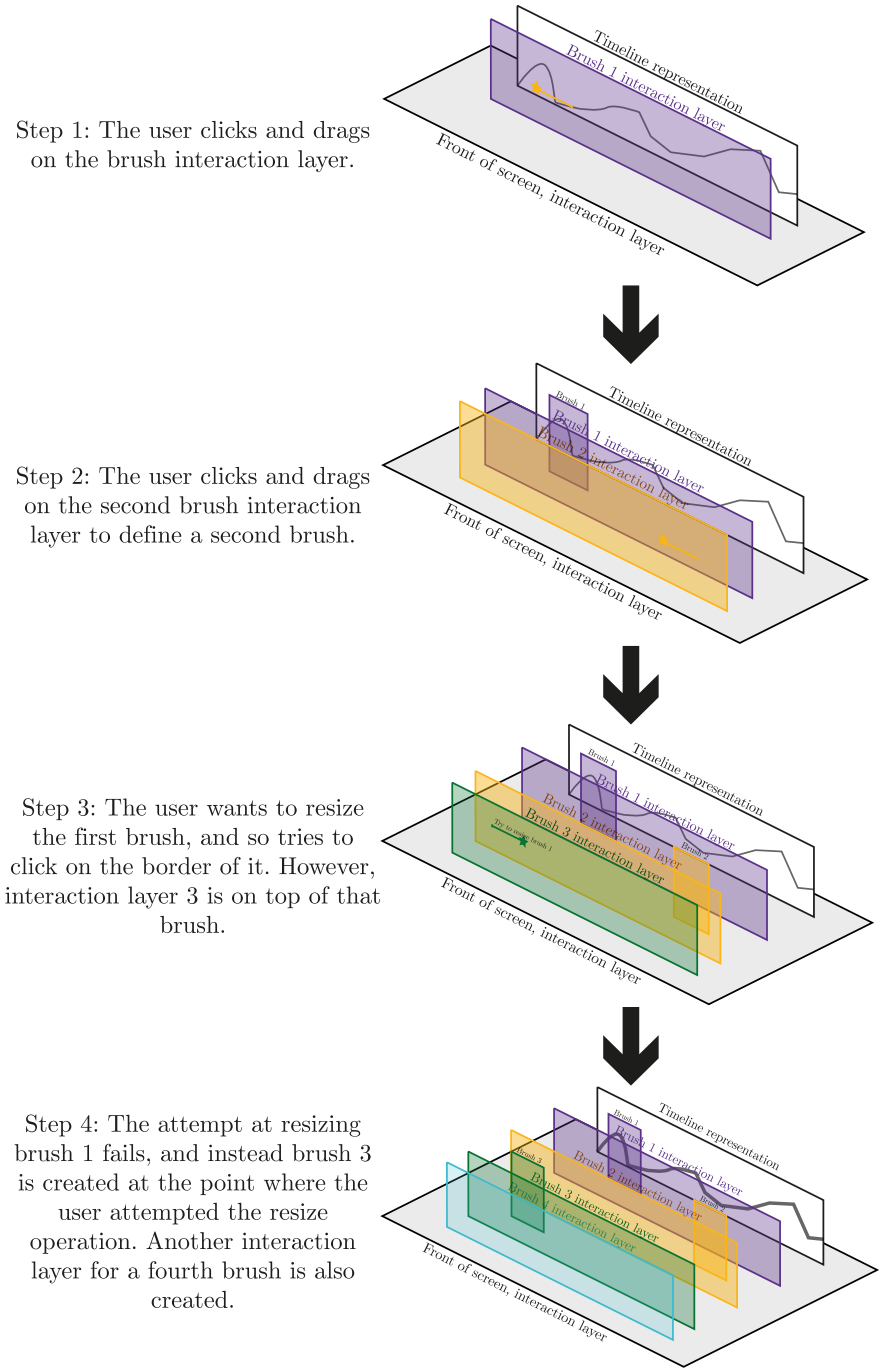


Figure 3.13: This is the layout and brush layering that takes place when attempting to use multiple brushes with native D3. Similar to Figure 3.12, notice how each visual representation of the brush selection (brushes 1, 2, and 3) are sandwiched between their brush interaction layers. From Parts 3 and 4, notice how this layering would prevent the user from resizing an existing brush. The brush interaction layers shown here are invisible to the user within the system, and are only assigned a colour in this figure for readability purposes.

blocking underlying brushes, and therefore preventing the user from making changes to their analysis would be a good design choice.

To mitigate these problems, we modified the D3 library so that when defining the brush function in system code, the developer passes the maximum number of allowed brushes as an argument. Our modified version of D3 then creates a single listener layer for the first brush. After the user defines the first brush, assuming the maximum allowed number of brushes is greater than one, a second listener layer is created for the second brush. However, crucially, the first brush object is raised *above* the second brush listener layer in the DOM tree, rather than being trapped below it as in Figure 3.13. This method, illustrated in Figure 3.14, ensures that previously defined brushes can be edited without interfering with the creation of new brushes.

Related to our use of a touch screen, we also add functionality to the brushing to prevent errors associated with users defining too small brushes. Again, D3 does not natively handle the accidental or incorrect creation of a brush (i.e. where the width of the created brush is 0px), though this is because it is not an issue in cases with only support for one brush as the user can simply click and drag again and create the brush from scratch. As our method (shown in Figure 3.14) creates a new full-length listener layer after the user defines a brush, we needed to prevent this from happening in the error case. Instead, we check that the defined brush has a width of over 0px and also that the defined brush captures a non-zero amount of underlying data. If these conditions are not met, then the new listener layer is not created, and therefore the number of brushes present does not increment.

Due to the request of our expert practitioners to include statistical measures for the network (their motivations and the context for this request are detailed in Chapter 4), we also needed to modify how D3 binds data to elements. We needed to pass the entire dataset to the library that calculated statistical measures but did not want to add to the overhead associated with loading the dataset into browser memory each time this needed to be calculated, as statistics were recalculated on the creation of any new row, new column, or any selection. Due to the asynchronous nature of D3, it is not possible to simply load data and assign it to a variable, and the accepted practice is to load the data and carry out all operations involving it within the data loading function. However, this method leads to large overheads in time, memory, and processing power in cases like ours where we wish to repeat functions in an analysis cascade.

Therefore in our modified version of D3, we added a function to bind the entire dataset to a special DOM element. This special element could then be selected and treated as a dataset containing variable like other programming languages (e.g. Python, Java). Through this, we

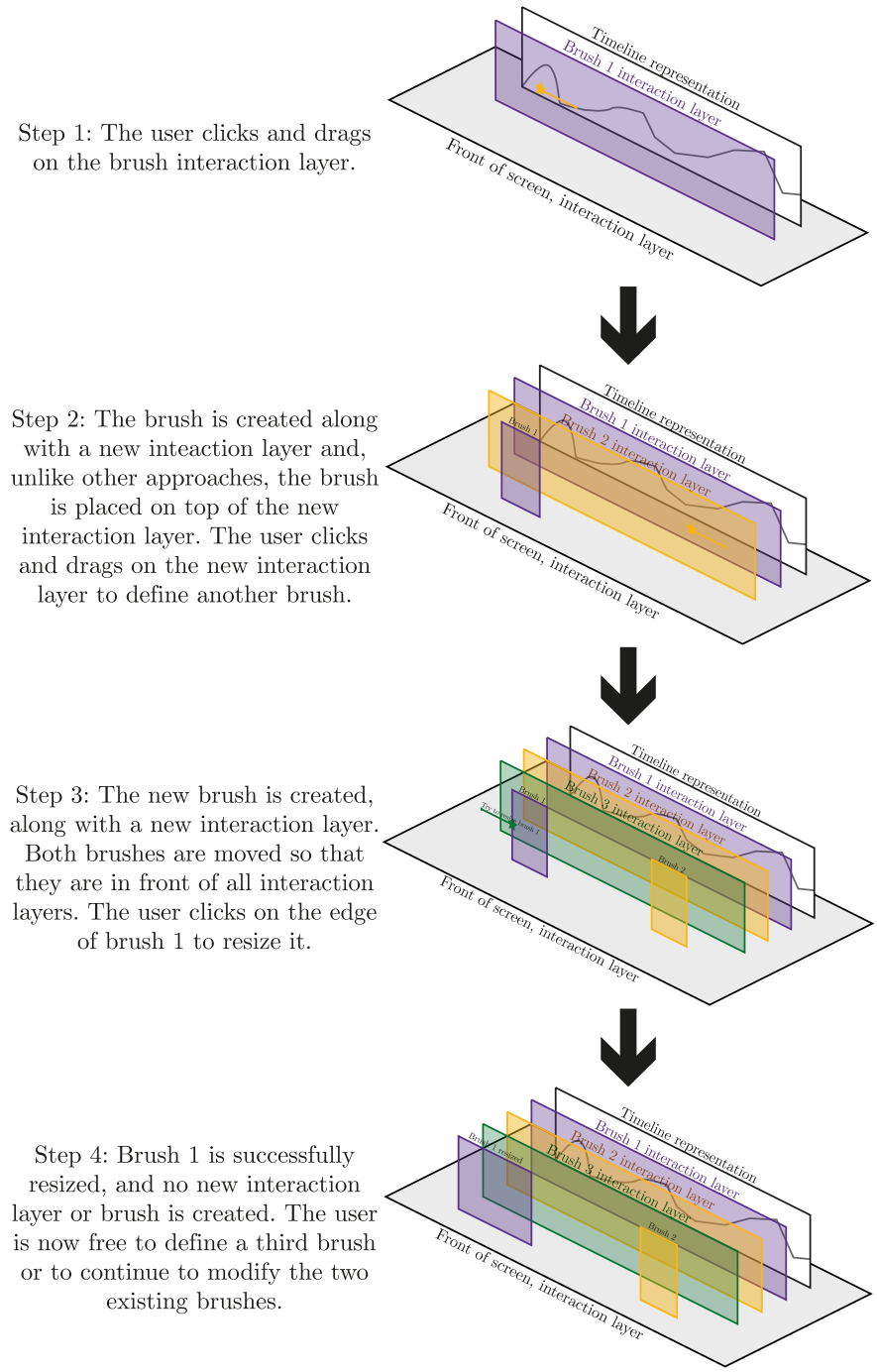


Figure 3.14: This is the layout and brush layering that takes place when using our modified version of D3. Notice how, unlike Figures 3.12 and 3.13, each brush representation is placed on the same layer, above all of the individual brush interaction layers. By adding handles to the outlines of these individual brushes, we allow the user to target and modify previously created brush selections while also facilitating the creation of new brushes where appropriate. The brush interaction layers shown here are invisible to the user within the system, and are only assigned a colour in this figure for readability purposes.

also allowed subsets of the dataset to be bound to their associated DOM elements; for example, vignettes would have the whole raw dataset for the user-selected time period attached to them. Our new data binding method meant that the system did not need to reload the raw data on the creation of a new representation type.

Related to this, we also modified the selection function to reduce the reliance on method chaining. The traditional D3 method of selecting a deeply nested element, for example nodes within a vignette, would be as follows:

```
// select the data of all nodes with colour associated with wound grade 3 from the first
// vignette of the main timeline
let nodes = d3.select("g.mainTimeline").select("g.vignette1")
    .selectAll("g.gravityContainer).selectAll("g.nodes")
    .filter(function (d) { return d.fill === wound_grade(3); })._groups[0][0]['childNodes']
// the above returns an array of SVG Objects, so to access the data the user needs to iterate over the array to access the inner data element
let node_data = []
nodes.forEach((d) => {
    node_data.push(d.__data__)
})
```

Listing 3.1: A classical chained method selection of nested elements using D3

Using our modified, shortened selection, the above code becomes:

```
// select all nodes with colour associated with wound grade 3 from the first
// vignette of the main timeline
let node_data = d3.select(timeline=0,vignette=1,object="nodes",filter={"fill": wound_grade(3)}, return="dataset")
```

Listing 3.2: Selection of deeply nested elements with our modifications to the D3 library to reduce method chaining.

We believe that this modified type of selection is both more readable and cleaner than the traditional method chaining, especially in complex interfaces where the developer might need to chain 20 or 30 methods to reach the deepest level of the DOM.

For the line duplication functionality, to save the results of single or group selections (see Chapter 3.4.3), we also introduced a method for cloning via shallow copy. Shallow copy cloning ensures that no interaction layers are copied into the saved line, making the saved line an immutable representation, supporting DP2 and DP4.

As well as our modified version of D3, we also use the Underscore.js library[3], which

[3]https://underscorejs.org/

provides powerful array and dictionary operations that are missing from default JavaScript. To calculate the displayed statistical measures, we use the JStat library [4].

DNP is built primarily as a series of nested SVG elements to support both analysis cascades and the weaved layer structure of the system, as shown in Figure 3.11. DNP has been designed as a browser-based tool; thus, it runs through a local server into Chrome, Safari, or Firefox. Touch screen input from the large display is the native input that the system expects to encounter. It is possible to run DNP on any machine, although at the moment, the screen size is set at a width of 3840px. DNP provides a method for translating mouse and keyboard input events into expected touch screen events. The DNP is functional up to two million edges on a machine with average performance (16GB RAM, 2.5GHz i5 processor). There is an evident deterioration in performance past two million edges though there is the potential for this to be addressed through a modification to the rendering engine. Increasing RAM by a factor of 8 to 128GB ensures smooth operation up to 20 million edges; however, the bottleneck with this level of computing power ceases to be on the computer, and instead, the user becomes the primary bottleneck; primarily due to the visual complexity associated with a considerable number of edges.

3.6 Discussion

Throughout our design work, we have gained a new respect for supporting provenance. Effective support of interaction provenance information not only allows analysts to trace their analysis backwards, but it has advantages for finding new routes of exploration and direct new insights. We believe that new designs and programming models for integrated provenance in visualisation (e.g., [68, 93, 244]) have great potential for dynamic network analysis. It is nonetheless important to highlight that our system does not offer comprehensive coverage of interaction provenance. For example, changes in a timeslice's span or position are not recoverable or visible in the current version. It could be difficult to provide comprehensive coverage without over-complicating the interface; this is a fertile area for future research.

Naturally, there are still many challenges to tackle in the analysis of this kind of data. One of the most challenging elements for us to address has been the performance issues to achieve responsive visualisations with dense links between the large numbers of small multiples. Several multi-objective optimisation trade-offs are particularly intricate. For example,

[4] http://jstat.github.io/

when laying out the force-directed diagrams, it is necessary to have some pre-computation of final node positions and layout decay constants.

Despite these challenges and limitations, however, we believe that Dynamic Network Plaid, presented in this chapter, can facilitate the visualisation and exploration of long in time dynamic network data. Visualisations of this type of data are generally complicated and of limited use for many complex graph tasks, in part due to display restrictions of small screens. We designed DNP specifically for a large display to ameliorate this restriction through the extended physical navigability and usable visual area of large displays. The extra available screen space also allowed us to provide (partial) interaction provenance information of the visualisations, which is useful in reducing the mental load of a user when many representations are visible.

The DNP's design specifically considered the extended visible area made available by the large display and the interaction techniques required for such set-up. Many approaches attempt to 'scale-up' existing work for large screens, but this does not provide the user with the tools to explore a dataset of this type in depth. When designing a visualisation, it is essential to consider the display method; mobile devices differ substantially from wall displays. As part of the design, we have also provided a novel method of combining certain types of information provenance and consistent spatial mappings.

Some features of DNP could be adapted for small screens. Activity distribution timelines that use height to encode node ownership, therefore ensuring that ABAB interactions and network evolution events are not lost through time-flattening, may be useful for some more dense node-link diagrams with several actors of high in-degree. Allowing users to interactively define their own timeslices, rather than having the system carry this out, ensures that the user is able to fine-tune the visualisation as and when they require. Follow on work by Wang et al. [323] introduce a computational method for the non-uniform timeslicing of dynamic graphs for visualisation. They achieve this by timeslicing not by time, but by number and complexity of events – more 'event-slicing' than 'timeslicing'. This can be thought of as an automatic version of our novel timeslicing method and has the potential to be useful for graphs that are so large as to be unmanageable by users alone.

To verify the utility of the Plaid as a whole, and better understand the tasks that interactive timeslicing is useful for, we carry out two empirical evaluation sessions with our target users on data that is of interest to them. These sessions are detailed in the following chapter – Chapter 4. For further validation of interactive timeslicing as a method for time navigation, we also conduct two formal studies where we experimentally compare this to both small multiples

and interactive animation. The first of these studies – Chapter 5 – uses a large touch-screen display, and we then extend this in Chapter 6 to a personal display with mouse-pointer input; allowing us to understand the generalisability of interactive timeslicing across multiple use-case situations.

Chapter 4

Expert User Evaluation

Contents

This chapter describes the evaluations in which we validated and tested our designs of the interface introduced in Chapter 3. For the Plaid, this took the form of two qualitative evaluation sessions with three public health researchers who had a specific interest in the Instagram NSSI dataset [59] that the Plaid was designed around. We found that these users knew that they needed to look at their data, but they could not articulate their analysis needs in a clear-cut way without having the initial interface as a jumping-off point.

The first qualitative session allowed us to iterate over and improve the features packaged in the Plaid based on feedback from the researchers and our observations. While the first session was with an unfinished version of the Plaid, the researchers could still uncover new insights into the data with it. They uncovered further insights in the second evaluation session with the final version of the interface. The second session used the most recent version of DNP described in Chapter 3.4. The researchers found the integration of graph statistics particularly helpful during this session as they were not present in the early version of the Plaid. We gained ethical approval for both of these sessions.

In addition to facilitating data insight, we also noted that the target users uncovered new patterns of tasks and exploration methods that they would not have had access to outside the Plaid. These tasks were not ones that the researchers would typically have chosen to carry out, primarily due to the inaccessibility of the tasks when using non-visualisation-based approaches, and one of the strengths of the Plaid is that it allows new task types that would not be possible in non-visualised data.

4.1 Method

These evaluation sessions took the form of structured observations of a group of expert practitioners. The participants were three public health researchers. When dealing with networks, their research area typically employs statistical measures on static graphs computed by aggregating long periods of time into a single network. Interactive visualisation is not yet widespread in this field, and dynamic network actor models were only introduced to their research area within the 18 months before the sessions. One of the participants could not attend the second evaluation session; therefore, we tested only two people in session 2.

The observations took approximately 90 and 110 minutes, respectively. They were both structured in three parts. First, the participants provided written consent. Second, they underwent a guided explanation of the interface's features and organisation, which was displayed in a Microsoft Surface 84" device with 3840x2140 resolution and touch input. After this, they

began to use the Plaid to explore their dataset. Participants could work independently or collaboratively. We recorded video of all their interactions with the interface and collected the answers to the interview and questionnaires.

We coded the video for occurrences of the following types of events: a) participants being confused about or making a mistake with a feature of the interface; b) participants encountering data insights through the interface, and; c) participants expecting a feature or asking for a feature in the interface. We analysed the answers to the questionnaires and interview in a similar manner. The following two sections provide a summary of the most important findings of each session. Notice that the findings below might relate to features that are not included in the final interface as described in Chapter 3.4, or that were created as a consequence of our observations. We make this explicit when appropriate.

We choose an expert evaluation for the Plaid system as it is a novel combination of visualisation techniques and time exploration, and therefore the Plaid offers a unique representation of the data and corresponding analysis.

For a complex system with multiple novel features, as shown in the Plaid, it would be unrealistic at an early stage to break down each feature and individually quantitatively test them (e.g. via formal laboratory experiments). It is also complicated to accurately assess insight discovery through quantitative experiments, particularly for datasets where we, as the visualisation designers, are not the expert users for the data in question.

But where a more realistic task is used in the evaluation – i.e. where a task matches the application domain – participants need a clear understanding of the problem that the visualisation tool is attempting to solve and also, one might argue, an understanding of the data itself [95, 179, 284, 305].

For these reasons, we choose to work with a small number of expert users for this evaluation and later carry out a more formal evaluation of our novel method of time navigation (detailed in Chapters 5 and 6. Our choice of evaluation methodology in this chapter aligns with the suggestion from Tory and Moller [305] that experts should evaluate early prototypes (formative evaluation) and that end-users should evaluate a refined version (summative evaluation).

4.2 Session 1

The first session was carried out with an early version of the DNP. The analysts started exploring the interface individually but ended collaborating in the exploration. This interface version did not have network statistics, data export, group node tracking, full-timeline tracking of

nodes, zooming, or vignette activity timelines. Additionally, most of the interaction happened via contextual menus activated on top of the vignettes. Our main findings for this session are related to how participants go from general overviews to detecting unexpected patterns in the data, exploring implications of the data, and being able to follow specific actors.

4.2.1 Activity Timelines are a Good Entry-Point for Exploration

On six occasions, we observed that participants used the primary timelines to focus on areas of increased activity in the data. This addresses the first challenge (CH1) identified in Chapter 3: practitioners typically start their analysis by calculating global numerical statistics, but this typically provides little access to unexpected or exciting occurrences in the data since there might be much variance in which actors are dominant and how they behave throughout the time span. Encoding wound grade as colour on timelines allowed participants to identify the wound grade responsible for the significantly increased activity level. These areas of increased activity would be challenging to identify without visual aids. Participants found that the time periods surrounding these activity features were helpful for initial data exploration. One analyst asked, “Why do I see a peak, and is there really a peak, at the 15-day mark?". Participant use of the activity features as entry-points to analysis supports the design decision to use timelines as entry points.

However, we should be cautious about relying on activity timelines as the sole entry point to the data. While undoubtedly a helpful initial exploratory step, there is the danger that the high points of the activity timeline will continue to direct and consume the attention of the user, occasionally to the detriment of other key areas of the graph that might aggregate to lower overall activity periods on the timeline. Some mechanism to either bypass, filter, flatten or otherwise manipulate the activity timeline would be a valuable addition to future interfaces that aim to use an activity timeline as an entry point.

On two occasions, we observed that our participants actively discounted the area of highest activity in the data as they felt that their focus should be on low activity areas when selecting time periods for comparison.

4.2.2 Network Structure Becomes Important in Comparison

After finding high activity times, the next step was to examine the network structure of those peaks and the surrounding times. For example, an analyst was interested in the underlying structure of the graph’s most significant activity peak (Figure 4.1). However, a single look at

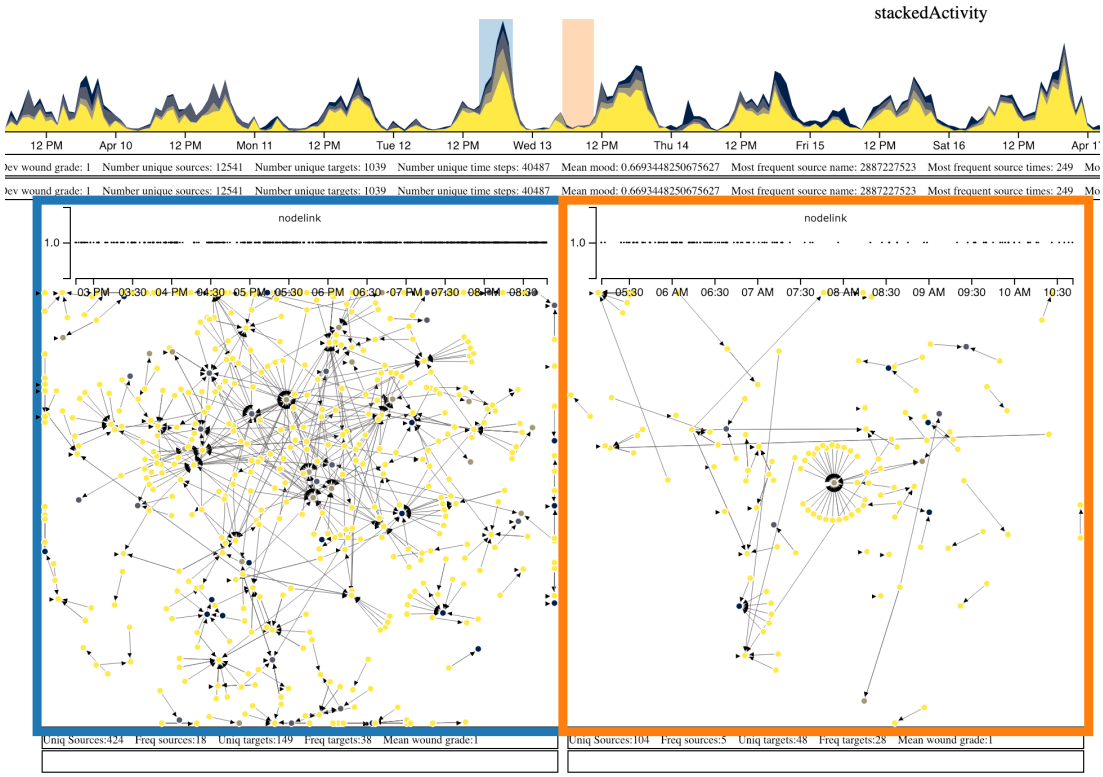


Figure 4.1: The large peak on the timeline with vignettes showing the difference in the network structure at the peak and in an nearby valley.

the structure of the peak does not offer enough information about abnormality, and therefore we see that comparison of the structure of the network at multiple time points is key to the completion of this task type. The researcher created several timeslices on the peak as well as other days to compare the network structures at different times. Through the vignettes, he found that, surprisingly, the network was very decentralised at the peak, the product of many posts with many interactions, rather than one or two specific very popular posts (Figure 4.1 left). A new timeslice of the valley occurring immediately after the peak of interest showed that the valley had one actor with very high in-degree, which led to the majority of interactions for that vignette (Figure 4.1 right).

> Within vignette two, there are lots of people responding to a single image; within vignette one, there are not people focusing on one image, there are more images, and the reactions are more spread out between them.

Through this, we find evidence that the Plaid can address several other challenges relating

to the practitioners existing workflow. By providing node-link diagrams on demand, we reduce the overhead typically associated with generating (CH2), refining (CH3), and drawing (CH4, CH5, CH6) these representations. We further address CH5 and CH6 with our novel method of non-uniform time exploration, as we provide views that do not flatten the entire time-space – and therefore preserve within-timeslice event order (CH6) – and also do not programmatically equally timeslice the data (CH5).

4.2.3 Selection and Linking are Key

The analysts then wanted to know whether the high in-degree actor of the valley above was also a contributor to the peak. They could quickly check this by selecting the node on the valley vignette, which revealed that this node did not appear in the peak vignette. This feature usage corroborates how, through features supporting DP1 and DP3, the system could offer swift answers to the analysts through interactive analysis. Our decision to maintain selections and facilitate linkage across multiple views, and view types, was a key factor for this insight. It is easy to imagine the challenges that one might encounter when trying to generate the same kind of question-answers through a textual-programming approach (consider challenges CH1, CH5, and CH8).

4.2.4 DP1 and DP3 Together Facilitate Navigation and Repetition

We observed that analysts quickly familiarised themselves with the interface's main consistent conventions (DP1) and oriented themselves very quickly in the interface (CH8), addressing further challenges relating to the interaction type (CH7) and the steps involved in the preceding analysis (CH9 and CH10). They selected an additional three slices to compare their structures (centralised vs decentralised) and discovered that the biggest group in the third vignette had the same central actor as in the second. However, the level of wound grade had increased. The central actor also had a much higher in-degree in the third vignette than in the second vignette. The analysts then questioned whether users were increasing the grade of the image they posted to receive a greater level of feedback. Here, we observed that our choice to design for large displays (DP1) and to enable consistent cross representation highlighting (DP3) allowed the analysts to navigate the interface quickly and repeat previous analysis actions.

4.2.5 Alternative Representations are Important

Building upon interactions that led to findings outlined in 4.2.3 and aided by the visual work history (DP2), participants continued examining the most significant activity peak, systematically comparing it with the activity distribution over each of the previously defined timeslices. They created a scatterplot vignette row with time encoded on the horizontal axis, wound grade on the vertical axis, and points coloured by global community membership. These alternate views led to participants hypothesising that: “the time of day is relevant to actor activity".

The previous paragraphs show specific examples of how different features resulted in interesting questions and new hypotheses–a landmark of good exploratory analysis. The examples above were a result of synergies between features implementing different design principles. Examples of other hypotheses and findings range from precise questions regarding a specific actor (“how central is this particular actor to the dynamic of the entire network and the generation of peaks") to more generic hypotheses about whether there were contagion effects for certain kinds of actors, whether actors escalate in wound grade because they receive reaction or because they do not, characterisation of “actor popularity” and “time of day” effects or even speculations about ‘what is the best moment to intervene?’

The analysts explicitly stated that “we would not have found this actor behaviour [4.2.4] over time with statistics due to the way we bin data before processing.” Based upon feedback from participants in this session, we were also able to identify missing features (e.g., multi-node selection in vignettes, vignette timelines) and cumbersome interactions (e.g., application of operations and representation changes using contextual menus). These resulted in changes incorporated in the next version of the DNP.

The inclusion of alternate representations in the Plaid reduces the overhead that would otherwise be associated with manually generating these visuals (CH2, CH3, CH4).

4.3 Changes to the Plaid after this session

After the first evaluation session of the Plaid, we were able to identify features of our interface that were difficult to use or that did not make sense in the interaction context that we provided with the large touch-screen display. As well as our observations, the expert practitioners who took part in the first session also provided feedback regarding new features that they felt would better support their existing analysis process and that would further address the associated challenges identified in Chapter 3.

We proceeded to make several changes to the Plaid from our observations and the practitioner feedback, primarily by adding new and extending existing features. The exception to this was the contextual menu that allowed the user to select different representation types. The trigger event for this contextual menu was a right-click operation adapted for touch screens, meaning the user had to tap with two fingers simultaneously. We observed, on multiple occasions, that the study participants had difficulties executing a right-click event within the Plaid window – a browser, as the Plaid is web-based – using the touch screen. Due to the use of the browser, it was too easy for users to accidentally activate the underlying built-in Chrome contextual menu rather than the Plaid contextual menu. This interaction error led to confusion among users where the interface performed in an unexpected manner.

In terms of additions and extensions to the first version of the Plaid, we received several direct and indirect feature requests from our participants. Due to the expert practitioners familiarity with network statistics and how strongly their mental model of networks tie in with these statistics, we added panels to the system that display both high-level overview measures of the data (e.g. network profiles of the whole timeline, whole vignette) and more detailed statistics (statistical attributes of selected nodes). Our expert practitioners also requested the ability to export timeslice data from the Plaid in an R-compatible format.

We also made changes based on findings and observations from the first session. Finding 4.2.4 – change in actor behaviour over time – led us to extend the single node tracking functionality to operate over groups of nodes, allowing users to more easily differentiate between the node makeup of regular clusters across their selected timeslices.

Participants expressed a wish to know precisely where specific nodes acted as sources/targets across the whole timeline and also to know what the wound grade was at these occurrences. Further informed by findings 4.2.4 and 4.2.3 we also added the ability to apply single and group node tracking to the entire timeline. This tracking allows the user to select a node or group of nodes with 'flags' applied to the timeline where the selected nodes participate in events. We colour these flags to reflect the wound grade of the corresponding event.

We also observed our users using the inbuilt browser ability to zoom so that they could examine the network structure of single vignettes in more detail. They used this functionality most often where the vignette was quite crowded with many elements or where clusters were dense enough to impact readability at 1x zoom. Participants also expressed that it would be helpful to have some way to filter vignettes by nodes. To address this, we introduced a geometric zoom function, and to facilitate node filtering, we also allow the creation of a new row

of vignettes, which only shows the nodes in the target zoom coordinates.

Finally, we were aware that version one of the Plaid did not wholly allow us to address CH6 – the loss of within-timeslice event order information. Our desire to sufficiently address CH6 led to the introduction of vignette activity timelines, small timelines within each vignette with 1px dots representing that an event occurred at the relevant time. This representation tied in with the selection and linking as selecting a node within the vignette would raise it either above or below the other event points within the activity timeline to show exact times within a timeslice where the node events occurred. While we did not see our expert practitioners looking for or expressing a need for this type of representation in the first session, we still felt that it was a significant challenge for us to address in the evolution of the Plaid.

4.4 Session 2

This session used the most recent version of DNP described in Chapter 3.4. This time the analysts worked collaboratively from the beginning. Only two analysts attended this session. We observed many of the patterns and uses of Session 1; however, we focus on features that were not present in the previous implementation of DNP.

4.4.1 Value of Per-Vignette Timelines

We observed that participants queried a single actor's activity distribution within their defined timeslices (i.e. not over the whole graph) on eight occasions. When an image was posted, the point in time may explain why some networks appeared denser than others due to an artefact intrinsic to timeslicing (CH5 and CH6). If a post is made at the beginning of a timeslice, it is likely to have received more likes/comments than one posted at the end of a timeslice. The per-vignette timeslices allowed the analysts to recognise this problem and to attempt to determine whether their observations were due to the artefact and whether it interacted with the "popularity effect" [121] described previously. In combination with the node selection function (DP3), analysts could use the vignette activity timelines to examine ABAB interactions which would typically be very difficult to observe with existing visualisations (CH4, CH5, CH6, CH7). Vignette activity distribution timelines also allow participants insight into within-timeslice actor behaviour, ensuring that network evolution events are not lost through time flattening (CH5, CH6, and CH8).

4.4.2 Global Tracking of Supporting Behaviour

We observed on two occasions that participants queried the global behaviour of supporters of high in-degree nodes. Participants identified several timeslices in which two distinct clusters of nodes existed with both primary target nodes having high in-degree and a wound grade always greater than one. The two clusters did not intersect at any point. The introduced group selection feature (which reinforces DP3) and the ability to filter a group creating a new vignette row creation (DP2) allowed participants to highlight all source nodes from cluster one from which they could create a new vignette row to compare to the previous cluster selection (CH5, CH6, CH8, CH9). The analysts concluded that the actors in the clusters were very different, with the first group primarily posting images of no harm and engaged in like/comment activity on lower grade images and the second posting high-wound-grade images of self-injury and only occasional like/comments on other high-wound grade images.

4.4.3 Tracking Changes in Actor Behaviour

Following from 4.4.2 (two different types of supporter clusters), the analysts were interested in whether supporter activity influenced the long-term future activity of the high in-degree nodes (CH5 and CH6). The availability of all their previous selections as rows above (DP2, CH8, CH9), accessible in a familiar way (DP4, CH10) was particularly useful since they did not have to reproduce intermediate states. Just by clicking on one of the earlier row vignettes they were able to quickly determine that the activity of one of these central actors became more sparse in time (by looking at the highlighted vertical in the global timelines—DP3) and lower in wound grade (going from 3 to 1). In contrast, the central actor of the second cluster became more active (the analysts could compare the two relevant rows vertically by hiding everything else in between).

4.5 Discussion

Our limited but informative evaluation shows that DNP is able to address many of the challenges inherent in visualising long in time dynamic network data, an overview is given in Table 4.1. Analysts were often able to carry out complex graph tasks [172] that triggered new and unexpected insights. Although the researchers were initially sceptical about the ability of DNP to provide value, they now clearly recognise how the interactive workflows afforded by DNP have opened new perspectives on the data. Naturally, this does not invalidate their

expertise or statistical validation and research procedures, but the analysts see compelling reasons to use interactive visual graphics in the exploratory phases. In fact, they have repeatedly requested us to work in the direction of integrating their existing procedures (mostly based in text-based programming languages) with the Plaid. The *slice export* feature is a first step in this direction.

Table 4.1: A summary of the challenges that we used to inform our design of the Plaid and how the findings from the two evaluation sessions demonstrate that our system is able to address these challenges.

Beginning of Table 4.1		
Challenge reference	**Description**	**Associated finding reference**
CH1	Aggregate network statistics do not provide a clear entry-point to the data.	4.2.1, 4.2.3
CH2	The visualisation of this type of data can be time-consuming and complex.	4.2.2, 4.2.5
CH3	Refining node-link diagrams can delay the workflow.	4.2.2, 4.2.5
CH4	It is difficult to find good layouts for node-link diagrams of this type of data.	4.2.2, 4.2.5, 4.4.1
CH5	Timeslicing can miss significant network events if they are not equally distributed through the dataset.	4.2.2, 4.2.3, 4.4.1, 4.4.2, 4.4.3
CH6	Aggregate node-link diagrams lost event-order within a dataset or timeslice.	4.2.2, 4.4.1, 4.4.2, 4.4.3
CH7	It is challenging to find the appropriate interaction for a visualisation.	4.2.4, 4.4.1
CH8	Effectively representing the connection between multiple representations.	4.2.3, 4.2.4, 4.4.2, 4.4.3
CH9	Maintaining awareness of what processing the data being visualised has undergone.	4.2.4, 4.4.2, 4.4.3
CH10	Helping users navigate a complex interface.	4.2.4, 4.4.3

Although this kind of evaluation might not generalise and is by no means definitive, it shows that the challenges that we have considered and the design principles that we have

chosen to address offer promise to improve the workflows of many scientists working with networks that evolve over long periods. Although all the design principles are straightforward and have been advocated many times in the communities of InfoVis and Visual Analytics, we believe that this prioritisation and, perhaps more importantly, the specific synergies that we found between the top three design principles (e.g., the unexpected advantages of prioritising interaction provenance when the small multiples are densely linked through interaction) can be a successful way forward. Our experience with our group of researchers also makes us suspect that it is difficult for most researchers in these areas to see the potential value of visualisation a priori. Any single operation carried out with our system is reproducible without too much effort by a competent programmer; yet, the agility of the interface, how some patterns jump at the analysts without having to switch to a "programming mode", and the value to work collaboratively in the exploratory part of the analysis are only now evident to the researchers, who want to use the tool in their daily practice. In other words, it is the multiplicative effect of features designed with this dataset in mind that make the difference, not any single one of them in isolation.

In this chapter, we have shown how we validated the design and implementation of the system presented in Chapter 3 via two experiments with public health researchers. Results of the first experiment were used to inform the final design of DNP. The second evaluation shows that users can use DNP to find insights in complex data that they would not usually look for, and that would be almost impossible to detect using traditional statistical methods. Findings from the second experiment illustrate how DNP can facilitate complex graph tasks [172]. Both validation sessions show the clear benefit of many linked sequence views and the ability to simultaneously view multiple representations. These sessions also validate our approach for addressing temporal information loss through temporal aggregation and ABAB events within user-defined timeslices.

To validate the principal component of the Plaid – the method of time selection – we also feel that it is necessary to scientifically evaluate the performance of the interactive timeslicing selection method when compared to small multiples and interactive animation. We carry out two formal experiments to validate this time selection method, and these are presented in Chapters 5 and 6. These experiments also allow us to make more accurate judgements about the generalisability of the Plaid on both large touch-screen displays and personal displays with indirect mouse input.

Chapter 5

Evaluating Time Navigation for Long in Time Dynamic Graphs

Contents

In the previous chapters, we presented Dynamic Network Plaid, a system for the visualisation and analysis of long in time dynamic graphs, and a corresponding empirical evaluation of the Plaid. The principal contributing features of the Plaid are interactive timeslicing, the consistent visualisation of interaction provenance data, and the visualisation of statistical measures per selection. To validate the design choices that we made with the Plaid, we first carried out two user evaluations.

To build on groundwork knowledge gained from the design and testing of the Plaid, we needed to understand how interactive timeslicing performed when experimentally evaluated against other popular methods of time interaction. The focus of this chapter is a formal study allowing us to validate the interactive timeslicing approach, which is the backbone of the Plaid. We target only the interactive timeslicing feature, rather than provenance or statistical display, because it is essential first to understand how these features perform in isolation before we assess their effects in combination.

This study compares our interactive timeslicing method with two approaches, small multiples and interactive animation, that are both popular in literature [3, 27, 37, 147, 170, 213, 315] and that have been experimentally compared in the past [18, 21, 102]. Interactive animation provides the viewer with an interactive animated representation where the viewer can control which moment in time is displayed. Small multiples involve splitting the time domain into a series of timeslices and representing them separately. Multiple studies have shown that the small multiples approach is faster than interactive animation with no significant differences in error rate for several task types [18, 21, 102].

The above approaches and experiments all assume uniform slicing at a given level of granularity. However, what uniform duration of timeslice should be chosen? If the timeslices are too coarse, the representation collapses too many events onto the same timeslice, hiding the subtlety and the actual order of events within each timeslice. If the timeslices are too fine, there are too many points in time to navigate in the data and analysts will have a hard time remembering timeslices that are off-screen when identifying patterns. Interactive timeslicing addresses this issue by allowing the analyst to interactively select the width and location of timeslices with the possibility of representing several of these timeslices at once (see Chapters 3 and 4). Similar methods have been used in other areas [237], but the effectiveness for interactive timeslicing for dynamic graphs is still unknown.

In this chapter, we present the results of a study, made up of a series of experiments, to test an interactive timeslicing approach against interactive animation and small multiples

for navigating time in dynamic graphs. We tested these approaches on a touch-based 84" display with 4K resolution that we consider representative of advanced visualisation set-ups in current professional collaborative settings. Currently, there is no evidence that an interactive timeslicing approach will be better; the additional complexity of interaction might be too costly, in terms of time, which could negate all benefits.

We conducted three experiments [1] where participants interactively navigated time to find: a) changes in graph structure at points in time (specific timeslices), b) changes in graph structure across an interval of time (a series of consecutive timeslices), and c) changes in attributes at points in time (specific timeslices). For each experiment, we had two conditions: near, where the moments in time of interest were close to each other in time, and far, where the moments were further apart in time. We gained ethical approval for the experiments detailed in this chapter.

5.1 Experiment Interfaces

For this study, we consider three dynamic graph visualisation techniques: small multiples, interactive animation, and interactive timeslicing. Animation and small multiples were chosen because they are widely considered the dominant alternatives in dynamic graph visualisation [37], are in common use, and have been revisited in recent data visualisation experiments for small displays [56]. Interactive timeslicing represents a novel alternative that is promising but has not yet been compared with the other two approaches [188].

We use node-link diagrams for all graph representations as they are popular in media and have been well explored in literature. All approaches were implemented to work on a large touch-enabled display (see Apparatus).

5.1.1 Interactive Animation

Interactive animations present the dynamic graph as an interactive film, with the user being given control of playing the dynamic data via a slider that can travel both forwards and backwards in time. Nodes and edges fade in and out of the drawing area as they are inserted or removed from the network.

For this study, our animation interface has four components. The component labelled (A) (see Figure 5.1, interactive animation) was the main timeline which showed the number of

[1] All experimental material is available on the OSF at `https://osf.io/bdpnr/?view_only=8d2e29693b714b7d8bb4abd407ad8e56`.

edges within the dataset aggregated at the hour level. The component labelled (B) was an interactive time window selection. Users would touch and drag to select a time window from the longer time series, with component (B.1) also selecting the 6 hours immediately preceding the user-selected time window. An alternative solution would have been to animate directly on the long timeline. However, this could potentially introduce a confound into our experiment as the animation would need to consider a much larger amount of data (the entire long-in-time dynamic graph) rather than a shorter animation around a given time window. Therefore, we decided to allow the user to select the animation window on the longer timeline around the target area.

Component (C) is the flattened graph representation of the time window selected by component (B.1). This static graph allows the participant to enter an answer without navigating through the network. Component (D) is the interactive animation window — when the time range was first selected, component (D) showed a flattened, static, representation of the graph within the selected time range. The participant pressed and dragged on the area (D.1) to control the rate and position of the animation. A purple line within (D.1) follows the participant's finger to show the current temporal graph position relative to the timeline.

5.1.2 Small Multiples

Small multiples use many interrelated graphs to represent time. It is analogous to a comic book representation for time where the evolution of the data can be tracked by reading across the frames from left to right. We use the term small multiples as it is most similar to those in other experiments [18, 21, 56, 102, 255]. For temporal networks, it is common to uniformly timeslice data by flattening each timeslice into its own window, with these graphs then drawn next to each other in a timeline pattern. Small multiples can be used for most forms of dynamic data; in dynamic graph visualisation, it can be used both with matrices and node-link diagrams.

For the small multiples representation in these experiments, our study dataset (Chapter 5.3.3) was divided into blocks of 6 hours, giving a total of 120 timeslices which were translated into vignettes. Along the top of the screen, labelled (A), is the timeline. Each block of colour defines one 6 hour time range, and each block of colour also corresponds to the border colour of the relevant vignette shown in (B). The vignette area was designed to only show four small multiple representations simultaneously due to screen size and readability constraints. The remaining vignettes could be accessed by moving the scroll bar at the bottom of the screen, (C), touching and dragging on the vignette area, (D), or by moving the light grey

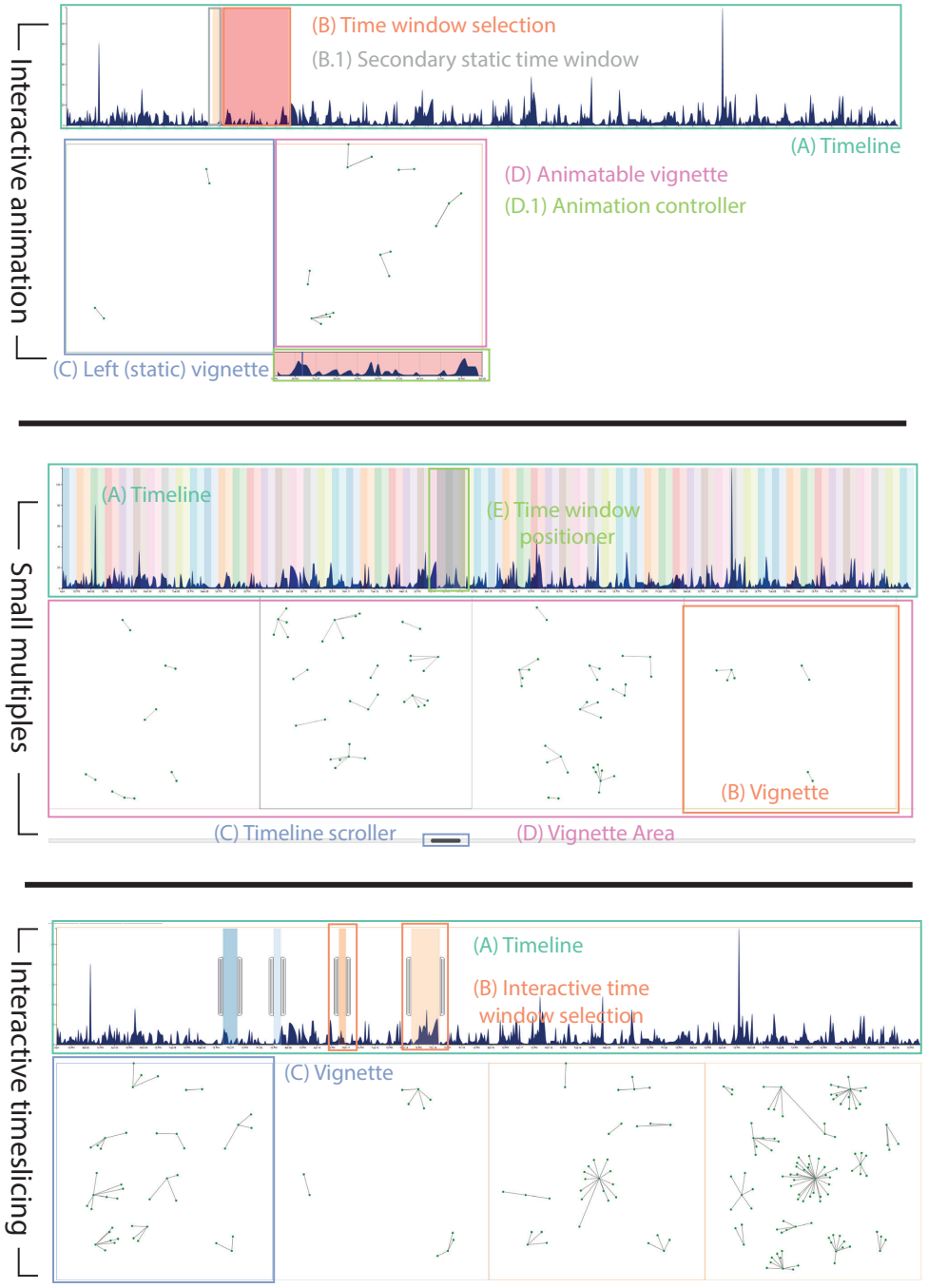


Figure 5.1: The interfaces used in this study. Interactive animation is at the top, small multiples in the middle, and interactive timeslicing at the bottom.

time-positioner on the timeline, (E). The grey time-positioner, (E), helped participants map the current position of the vignette area to its temporal position within the entire time series without performing colour comparisons of the vignette borders.

5.1.3 Interactive Timeslicing

This interface was based on a previous interactive timeslicing design [188] where users could define custom time ranges of any desired size or order and have the individual representations positioned based upon the order of user interactions.

Component (A) is the timeline that is common to all of our experimental interfaces. Participants touched and dragged at any position on the timeline to create a new time window selection of any size (B). A time window selection created a new vignette (C) containing a flattened, static, representation of the graph. A time window selection could be any multiple of 6 hours, with the edges of the selection snapping to the same timeslice intervals as all conditions — this was to avoid small off-by-one errors that may seriously hinder the ability of a participant to complete any given task successfully.

The maximum number of interactive timeslices that could appear on the screen at any time was four. This limitation was introduced to avoid giving interactive timeslicing an artificial advantage over small multiples. Interactive selection of timeslices in this way allows the participant to make independent time windows at very distant points in time for the network. However, there is an interaction cost associated with creating the time windows that is higher than both animation and small multiples: the location in time needs to be determined first, and then the participant needs to draw a much more precise window both in terms of its width and position.

5.2 Experiment and Research Questions

Our general goal is to provide empirical evidence that can support the design of better interfaces for time navigation when exploring dynamic networks. Thus, we designed three experiments, each testing a different task. For each experiment we seek to answer the following questions:

Q1 Which interface will have the lowest completion times for the selected tasks?

Q2 Which interface will be the most accurate for completing the selected tasks?

Q3 How does distance in time between data elements of interest affect the performance with the different interfaces?

5.3 Experimental Approach

We designed a series of three experiments through the following iterative process. First, we looked at existing relevant taxonomies of tasks [2, 11, 172] and at findings from previous observations of analysts working with dynamic networks [188]. Then we filtered out tasks that do not involve navigation between at least two given time points. We excluded tasks that only involve single time points, since are we primarily interested in challenges specific to dynamic, rather than static, networks. From the remaining tasks we selected three tasks with the following criteria: a) we preferred low-level tasks that might be components of larger tasks, and b) we preferred tasks that were very different from each other and where the focus was on different elements of the data (e.g., the structure of the network or the variation of attributes) and how they varied over time.

The chosen tasks can be broadly summarised as:

E1 Detecting graph structure changes at discrete points in time.

E2 Detecting graph structure changes over a time interval.

E3 Detecting single attribute change at discrete points in time.

We run three experiments with the same participants in the same session. We prioritise comparing results across interfaces on the same task and factor; to reduce noise from order effects on our comparisons of interest, we run the three experiments in the same order for all participants (i.e., we do not randomise experiment order). Experiments took place in early March 2020 and took approximately 85 minutes to complete.

5.3.1 Participants

Twenty-four unpaid volunteers (8 identified as females, and 16 as males, the age range of 18 to 32) from our department participated in the study. The number of participants was decided in advance using our prior experience from the four pilots and to ensure proper counterbalancing. We used a simple questionnaire to screen participants without typical colour vision and those who did not have basic computer and mathematics experience. We did not require prior knowledge of networks.

Before beginning the first experiment, participants filled a short demographic questionnaire indicating their familiarity with networks and network graph visualisation using a 1–5 Likert scale. Four participants gave their familiarity as 1 (no knowledge), six as 2, nine as 3, three as 4, and two as 5 (high knowledge).

5.3.2 Apparatus

The interfaces were built in JavaScript, primarily leveraging the D3.js library [48] with some modifications. Conditions ran in a Chromium browser window wrapped in a QT front end interface, with a Python Flask back-end. Implementations of the interfaces are identical across the three experiments; visual indicators for the task and methods to indicate the answer vary with the task, and we describe these within each experiment. All approaches were built for an 84" wall display with 4k resolution (3840 by 2160 pixels) mounted at 90cm from the floor.

5.3.3 Experimental Dataset and Graph Layout

The dataset used in these experiments is a filtered version of a previously collected dataset [59] which describes interactions between anonymised actors on Instagram who liked or commented on non-suicidal self-injury posts. Edges have unique identifiers, have a specific occurrence time (down to second precision), and do not have a duration (i.e., are considered atomic, only measured at the time of posting). A duration of one hour is assigned to each edge at its posting time to ensure it is visible and lies entirely within the six-hour time window. Additional attributes of the dataset were removed for anonymity; hence the dataset contains only nodes, edges, and the time that the nodes and edges appeared in the dataset.

The dataset is representative of typical dynamic networks but was substantially filtered to reduce its size to make our tasks feasible. We removed single interactions between actors (cases where two actors only interacted with each other once), self-interactions, actors that received less than 250 interactions, and actors who only interacted with removed actors. This filtering allowed a stable graph layout and made task duration appropriate to the available time. The final dataset consists of 776 distinct nodes and 8182 edges over 30 days. It had an approximate average density of 11 events per hour or 66 events per 6-hour timeslice. As this dataset is based on real data, the event distribution was non-uniform, ergo some time periods had a higher event density, while others had a lower event density. This variability over long time periods allowed for very different graphs to be presented at the time points and intervals in our tasks, which increases the generalisability of trials.

Our dataset is long in time with a fine-grained temporal resolution (hour resolution over a month of data), meaning that there are hundreds of timeslices. Typical timeslice-based approaches [51], in particular linking strategies, generally do not scale to hundreds of timeslices as nodes move unnecessarily with the many inter-timeslice edges being artificially inserted [290]. As we wanted to control for graph layout across conditions, we could not draw the graph on-demand as the user navigated through time. Thus, we needed to draw the full dynamic graph beforehand. Event-based dynamic network drawing techniques could provide a solution [290, 291] but they only scale to about 5000 events. Therefore, we computed a fixed layout for nodes and edges that uses pinning strategies so that nodes retain position whenever they appear, allowing the drawing stability to support the mental map across conditions for all experiments consistently. The final graph layout was generated first with the Visone aggregation strategy [53] (all events collapsed down to a single timeslice) and then cleaned slightly in Gephi [34], using its implementation of Fruchterman and Reingold's algorithm [111]. Although more scalable algorithms are available [132], our graphs are relatively small in terms of the number of nodes and edges. This allows us to use standard force-directed approaches instead of multilevel ones. The resulting layout still contained some overlapping nodes and edges; we manually adjusted overlapping items.

5.3.4 Experimental Design and Procedure

All three experiments share an identical structure in terms of factors, conditions, and repetitions. The main manipulation of interest is *interface*, with the three interfaces `Anim`, `Mult`, and `Int TS` as levels—see Chapter 5.1. One factor is temporal distance (near and far), which manipulates the separation of the target timeslices or the length of the time interval involved in the task. In the near condition, timeslices had a separation of 18 hours (three timeslices) while in the far condition timeslices had a separation of four days (16 timeslices).

The true impact of the temporal distance factor differs between interfaces. For the small multiples interface, for example, the far condition means that timeslices of interest cannot be on the screen simultaneously, and instead that users must scroll between them. The impact is similar to interactive animation, with the far condition increasing the distance that a participant needs to navigate between target timeslices. For interactive timeslicing, the result of the different temporal distances is expressed as an increased number of timeslices to be selected (as in Task 2 for the far factor), but for some tasks involving the selection of individual timeslices (Tasks 1 and 3) there was minimal difference within the interface.

	First trial	Second trial	Third trial
Group 1	Interactive timeslicing	Small multiples	Interactive animation
Group 2	Interactive timeslicing	Interactive animation	Small multiples
Group 3	Small multiples	Interactive timeslicing	Interactive animation
Group 4	Small multiples	Interactive animation	Interactive timeslicing
Group 5	Interactive animation	Interactive timeslicing	Small multiples
Group 6	Interactive animation	Small multiples	Interactive timeslicing

Table 5.1: Participants were assigned to groups so that each group had an equal number of participants. Participants would complete trails in the order given by the Latin square for each of experiments one, two, and three, always in that order.

We performed a within-participant design with participants completing three repetitions for each cell, for a total of 18 trials, and two additional easy tasks on each interface for training that were discarded before data analysis. Participants carried out trials (including training) in three blocks of 8, with 15-second breaks between each block. Possible ordering effects were counterbalanced using Latin squares for each participant. See Table 5.1 for the full Latin square design. A participant always did the interfaces in the same order for each experiment that they completed. Trials in the near condition always took place before trials in the far condition. We did not counterbalance temporal distance because we are not interested in the quantitative comparison of performance between near and far conditions.

Before the real trials of each interface, condition participants received a tutorial on the specific interface and carried out one trial example. Upon completing each experiment, participants ranked each condition on a scale of 1 (best) to 3 (worst), according to their preferred interface for completing the type of tasks tested. The experimenter also collected qualitative notes during the experiment. Participants had the opportunity to make other comments and share their thoughts about the tasks, interfaces, and hardware.

5.3.4.1 Task Completion

Across all interfaces, tasks, and factors, the interaction techniques do not change. However, the method of answer entry does change between tasks for all interfaces. The changes in answer entry between tasks is detailed in Table 5.2.

Task	Answer selection method
1. Graph structure changes at points in time	Direct selection of edges in the given timeslice
2. Graph structure changes over a time interval	Selection of a single timeslice via the timeline
3. Attribute changes at points in time	Selection of a single timeslice via the timeline

Table 5.2: A description of the way that answer entry changes depending on the task type. The answer selection method changes uniformly across all interfaces for a given task.

5.3.5 Statistical Methodology

The statistical methodology was decided during the experimental design process and recorded before data collection. All three experiments employed the same methodology.

Correctness and completion time were measured and analysed separately for all three experiments. Completion time was measured as the number of seconds (s) to complete each task for all three experiments. Measurements for correctness varied for each experiment, and these measures are detailed in their respective sections. However, for all three experiments, correctness was measured on a $[0,1]$ interval (with 1 corresponding to 100% correctness).

As we were interested in determining the performance of the three interfaces under near and far conditions, we chose to divide the data of each experiment by the near and far factor before beginning the analysis. The completion times and correctness of the three repetitions in each cell were averaged per participant. Completion time was log-transformed (log2) before analysis and compared through pairwise, two-tailed t-tests. We were uncertain if the distribution of correctness data would follow a normal distribution for our measurements; thus, we applied a Shapiro-Wilk test with $\alpha = 0.05$ to each condition of near and far for each experiment separately. For all three experiments, the correctness data did not usually follow a normal distribution. Therefore, two-tailed, pairwise Wilcoxon signed-rank tests were used for correctness in all three experiments. For each experiment, six pairwise comparisons (three for near and three for far) for time and six pairwise comparisons for correctness were performed. Holm–Bonferroni [149] corrections were used to determine significant results. We use Holm-Bonferroni to counteract the problem of multiple comparisons and to control for the family-wise error rate. The formula to calculate the correction is:

$$\frac{\alpha}{n-i+1} \tag{5.1}$$

where α is the significance level, n is the number of tests, and i is the rank number of pair (by degree significance).

The results are presented in the next section. In all result figures, blue corresponds to an-

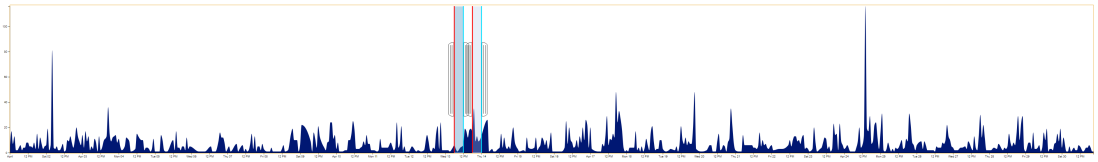


Figure 5.2: An example of the expression of Experiment 1 on the main timeline common to all interfaces. The red and blue line pairs (one each of red and blue makes a single pair) show participants where they need to start timeslicing and where they need to stop. This figure shows the timeslice selection as completed using the interactive timeslicing interface.

imation (Anim), red to small multiples (Mult), and orange for interactive timeslicing (Int TS). The mean is indicated using a circle and the median using a square. Error bars represent bootstrapped 95% confidence intervals computed using 250,000 repetitions. We report p-values to three decimal places in all figures [40, 126, 127, 326]. Solid lines indicate statistical pairwise significance after Holm-Bonferroni correction, and dashed lines indicate no statistical pairwise significance after correction.

5.4 Task 1: Graph Structure Changes at Points in Time

This experiment tests completion time and accuracy for tasks where the graph structure had to be compared at two separate time points. We suspected that interactive timeslicing would perform the best for this task (**Q**1 and **Q**2). Also, we anticipated that interactive timeslicing would perform much better than animation and small multiples for the far condition. However, we were less sure of the performance of the remaining two interfaces on this condition of the experiment as they have not been tested and were not designed for exploring distant points in time (**Q**3).

Procedure. The task prompt provided to participants for experiment one was 'each pair of red (start) and blue (stop) lines signify a timeslice. In the first timeslice, click on all edges that disappear at least once in all other timeslices.' The target timeslices of six hours each began with a red line and finished with a blue line. Participants completed the task by selecting edges in the first window that did not appear in the other window.

Figure 5.2 shows an example of how the task was expressed on an interface and how interactive timeslicing interface would be used to select the given time ranges. Figure 5.3 shows how a participant would use the lasso tool in the small multiples interface to select edges in the left vignette that are not present in the right vignette.

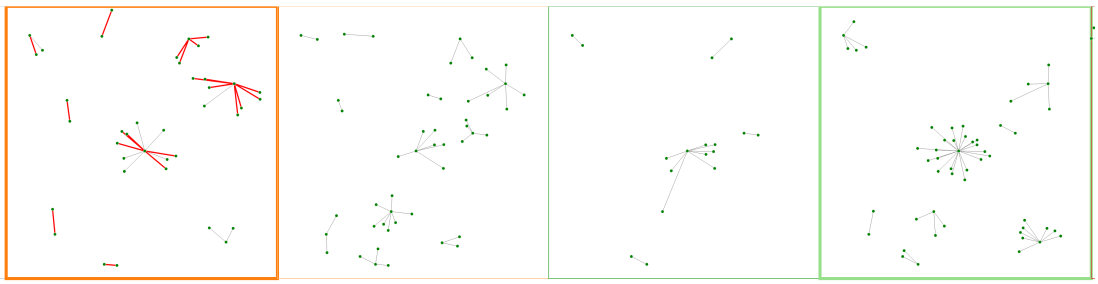


Figure 5.3: The vignettes shown in the small multiples interface for Experiment 1 on the near-in-time factor. Participants are required to compare between the vignettes with the bold borders (far left and far right). The far left vignette displays edges selected by the user in red with an increased line thickness.

For all interfaces, answers to the question were entered through lasso selection on the left-most vignette to control answer entry. Edges selected using the lasso tool were coloured red and increased 4× in width and could be deselected.

Animation. Participants dragged on the timeline starting from the end of the first red-blue pair of timeslice demarcation lines, ensuring that the vignette to define an answer was correctly created. This action would generate two vignettes: a left vignette displaying the selected time interval of the first pair of timeslice lines as a static, flattened, graph for answer entry; and a right vignette which initially displayed the flattened, static, graph covering the whole time range selected by the participant during the original time selection operation.

A timeline appeared at the bottom of the right-hand vignette displaying a zoomed version of the selected area of the main timeline, with task target timeslice areas shaded in blue. Participants were able to touch and drag on the timeline to animate the right-hand vignette. Touching and dragging in the left vignette would activate the lasso tool to allow participants to select, or deselect, edges for their task answer.

Small Multiples. Participants used either the grey time positioner (Figure 5.1, small multiples, item E) or the scroll bar attached to the vignette area (Figure 5.1, small multiples, item C) to navigate to the correct position in time. Task relevant vignettes were distinguished from non-task vignettes by increasing the border thickness 4×. Participants could touch and drag in the left vignette to use the lasso tool to define the answer set.

Interactive Timeslicing. Participants touched and dragged on the correct area of the timeline to create a time window selection (Figure 5.1, interactive timeslicing, item B). To avoid small

selection errors, user-created time windows always readjusted to the nearest 6-hour boundary, ensuring that the selection was the exact time period required by the task. After the creation of a time window, a corresponding vignette appeared below the timeline (Figure 5.1, interactive timeslicing, item C). Touching and dragging in the left-most vignette with the lasso tool defined an answer to the given task.

Measurements. Time (number of seconds) and accuracy were measured for this experiment. Accuracy involved comparing the set of edges selected by the user and the set of edges present in the correct answer of the question. In order to evaluate correctness in this case, we employ a method from pattern recognition and information retrieval. A perfect answer would have perfect *precision* (p) (no edges outside the correct answer are selected) and perfect *recall* (r) (all of the edges in the correct answer are selected). Precision and recall can be combined together into the F_1 score to give a measure between $[0,1]$:

$$F_1 = 2\frac{pr}{p+r} \tag{5.2}$$

The value of p and r are defined in the following way. Consider two edge sets: the set of participant answer edges (X), and the set of correct answer edges (Y).

$$p = \frac{|X \cap Y|}{|X|} \tag{5.3}$$

$$r = \frac{|X \cap Y|}{|Y|} \tag{5.4}$$

Results. Correctness and completion time are shown in Figures 5.4 and 5.5 respectively. After a Holm–Bonferroni correction, we found significant differences in terms of correctness for the near condition with interactive timeslicing outperforming animation with a difference of 14 percentage points ($W = 82$, $p < 0.001$) and small multiples with a difference of 6 percentage points ($W = 148$, $p = 0.003$). On the far condition, interactive timeslicing outperformed animation with a difference of 11 percentage points ($W = 109$, $p < 0.001$) and small multiples with a difference of 5 percentage points ($W = 143$, $p = 0.002$). No other differences were statistically significant. In terms of completion time, we have the same pattern. On near, interactive timeslicing outperforms animation with a difference of $42.1s$ (animation $1.61\times$ slower) ($t = 5.76$, $df = 23$, $p < 0.001$) and small multiples with a difference of $28.9s$ (small multiples $1.42\times$ slower) ($t = 4.28$, $df = 23$, $p < 0.001$). On far, interactive timeslicing outperforms animation with a difference of $52.5s$ (animation $1.91\times$ slower) ($t = 8.04$, $df = 23$, $p < 0.001$) and

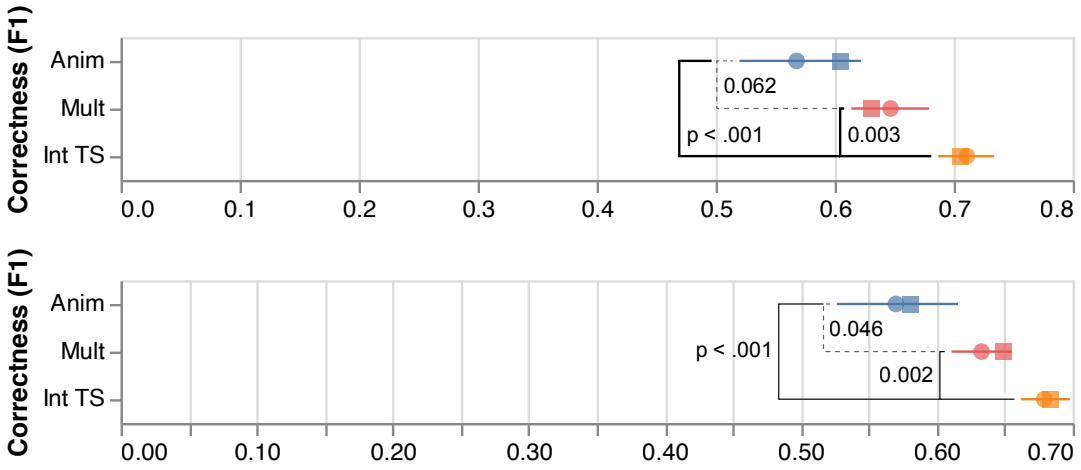


Figure 5.4: Experiment 1 correctness for the near (top) and far (bottom) conditions as computed by the F_1 score (Equation (5.2)). The dot represents the mean, square the median, and 95% CIs shown. Pairwise lines indicate p-values with solid lines indicating significant differences.

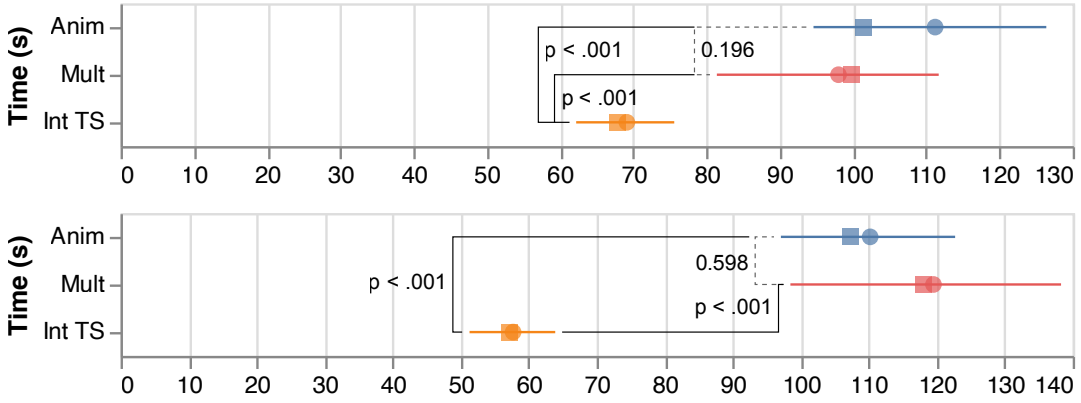


Figure 5.5: Experiment 1 completion time (s) in seconds for the near (top) and far (bottom) conditions without log transform. The dot represents the mean, square the median, and 95% CIs shown. Pairwise lines indicate p-values with solid lines indicating significant differences.

small multiples with a difference of $61.8s$ (small multiples $2.07\times$ slower) ($t = 8.01$, $df = 23$, $p < 0.001$).

Participant Survey. After completing the experiment, participants ranked each condition on a scale of 1 (best) to 3 (worst) to indicate their preference for conditions to complete the given task type. These ranking responses are in Fig. 5.6. For this experiment, 87.5% of participants indicated that they preferred the interactive timeslicing, with interactive animation second, and small multiples as the least preferred option. The primary criticism of small multiples was that,

Figure 5.6: Participant interface rankings. Cells annotated by number of votes and darker cells correspond to more votes. Rating is 1st, 2nd, or 3rd.

for tasks involving the far in time condition, it was impossible to put both timeslices on-screen simultaneously to compare them directly. Participants found it challenging to remember edge positions and edge presence when they had to scroll through many intermediate representations to reach the comparison target.

The memory cost was also probably a factor in the ranking of animation. However, this was less pronounced as the interactive animation interface made it simpler to switch backwards and forwards in time without viewing intervening timeslices.

Summary and Discussion. For both the near and far conditions, interactive timeslicing outperforms animation and small multiples in terms of correctness and completion time, confirming our conjectures for all research questions. There were no other significant differences found in the experiment. Interactive timeslicing was primarily designed for dynamic graphs spanning a long interval of time and specifically for comparing distant points in time, as required by the task. Animation incurs higher interaction costs by requiring interaction to play the animation back and forth between the distant time periods. Small multiples requires the participant to scroll back and forth between the distant time periods. Also, both animation and small multiples require participants to remember the structure at distant time points, whereas interactive timeslicing can show representations simultaneously and side-by-side on the screen. It seems that this benefit offsets the added interaction cost of interaction timeslicing. It is important to note that we did not see a significant difference between animation and small multiples for either near or far. This could be because neither of these interfaces were designed with long time series in mind. Thus, to compare distant points in time, these interfaces were too taxing on memory and interaction, which dominated the result.

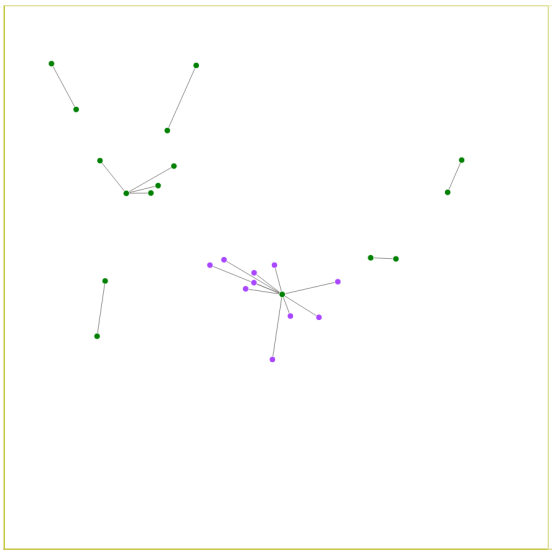


Figure 5.7: An example of a vignette with a cluster of purple nodes, as shown to participants during Exp. 2

5.5 Task 2: Graph Structure Changes Over a Time Interval

This experiment tests how interactive timeslicing performs over a time interval when examining a change in graph structure. A time interval is several consecutive timeslices. In terms of our research questions, we conjectured that interactive timeslicing would not perform as well on this task type because it is mainly designed for time points and not continuous intervals (**Q1** and **Q2**). We also felt that the performance of interactive timeslicing would decrease with far trials because a longer interval of time needed to be considered, and more information needed to be remembered (**Q3**).

Procedure. The on-screen prompt provided to participants for experiment two was: 'select a timeslice between the red lines where the cluster of purple nodes is most dense.'

The video figure in the supplementary material illustrates how participants answered this task on all three interfaces. Participants saw a pair of red lines on interface timelines (see Figure 5.1, (A), on all interfaces), indicating the beginning and end of the time interval of interest. Successful completion of the task involved investigating every six-hour timeslice within this interval to identify the timeslice in which there were the highest number of connected purple nodes (see Figure 5.7 for an example).

When participants were confident that they had found the correct answer, they would tap a

button above the timeline to switch into 'answer entry' mode. In this mode, participants would slide a green time window (the fixed size of one timeslice) to the position on the timeline containing their answer.

Measurements. Time (s) and accuracy were measured for this experiment. Accuracy involved comparing the number of edges between purple nodes in the selected timeslice (n_s) to the timeslice where the number of edges is a maximum (the correct answer) (n_a). In order to do this, we use the following measure:

$$c = 1 - \frac{n_a - n_s}{n_a} \tag{5.5}$$

The value of this measure is 1 when the correct answer is selected and diminishes to zero with a window of fewer and fewer edges between the purple nodes.

Results. Correctness and completion time are shown in Figures 5.8 and 5.9 respectively. After a Holm–Bonferroni correction, we found no significant differences in correctness between interfaces in neither near nor far conditions. In terms of completion time on near, small multiples outperformed animation by $9.3s$ (animation $1.24\times$ slower) ($t = 3.31$, $df = 23$, $p = 0.003$) and interactive timeslicing outperformed animation by $9.1s$ (animation $1.24\times$ slower) ($t = 3.54$, $df = 23$, $p = 0.002$). On far, small multiples outperformed both animation $17.8s$ (animation $1.34\times$ slower) ($t = 4.01$, $df = 23$, $p < 0.001$) and interactive timeslicing $21.8s$ (interactive timeslicing $1.42\times$ slower) ($t = -6.67$, $df = 23$, $p < 0.001$). No other significant differences were found.

Participant Survey. After completing the experiment, participants ranked each condition on a scale of 1 (best) to 3 (worst) to indicate their preference for conditions to complete the given task type. These ranking responses can be seen in Fig. 5.6. For this experiment, 92% of participants preferred small multiples for completing tasks of this type. Animation ranked second, and 67% of participants ranked interactive timeslicing as the worst.

There was widespread frustration among participants while using the interactive timeslicing condition during this experiment. The comparison of a number of time points within a time interval required much interaction effort. A large amount of very precise interactions were required to position and reposition the timeslices; in contrast, small multiples simply required scrolling the vignette display area while looking at the screen. For this task type interactive timeslicing was also vulnerable to 'fat finger' [289] problems, where participants aimed to

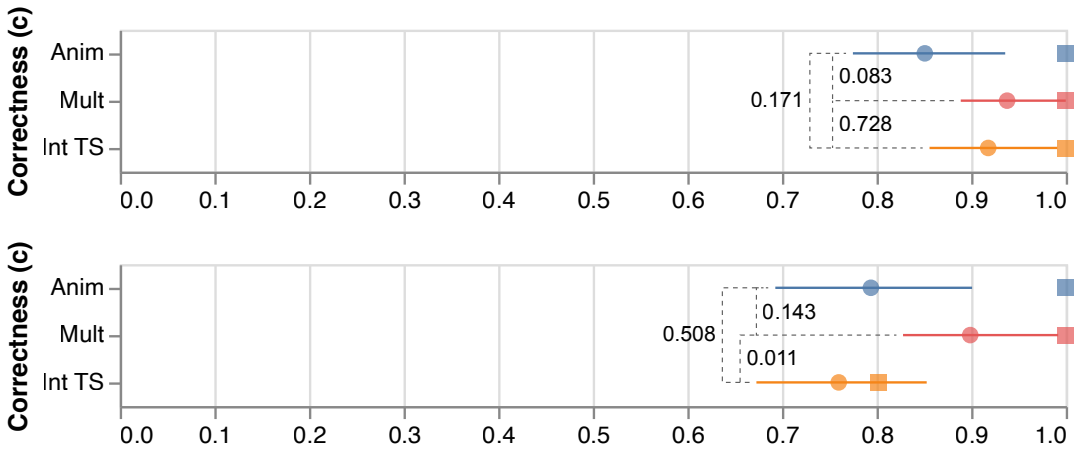


Figure 5.8: Experiment 2 correctness (Equation (5.5)) for the near (top) and far (bottom) conditions. The dot represents the mean, square the median, and 95% CIs shown. Pairwise lines indicate p-values with solid lines indicating significant differences.

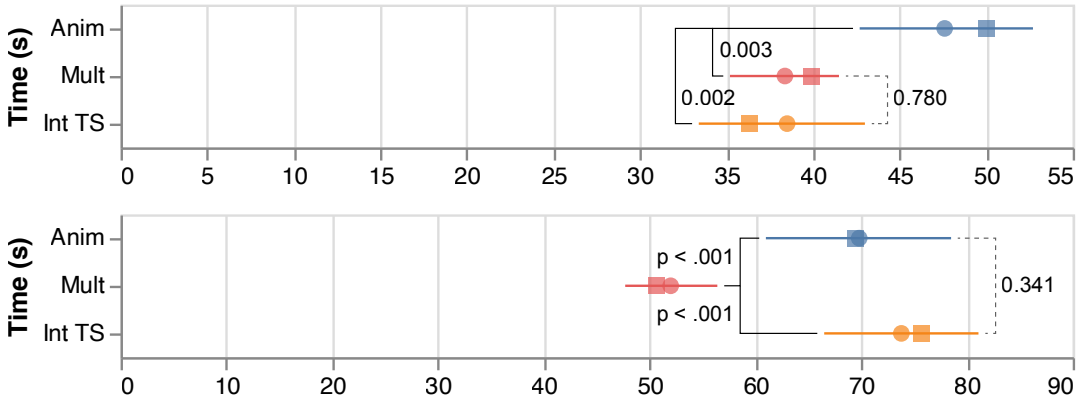


Figure 5.9: Experiment 2 completion time in seconds for the near (top) and far (bottom) conditions without log transform. The dot represents the mean, square the median, and 95% CIs shown. Pairwise lines indicate p-values with solid lines indicating significant differences.

carry out one operation but accidentally triggered a different one due to imprecise touching of the screen. A typical example occurred when a participant tried to move a time window selector but instead activated a resize operation by selecting a handle for the selected time window. The participant then had to return the time window to its previous size and attempt to carry out the repositioning operation again.

Summary and Discussion. For this experiment, we found no significant differences in correctness. Therefore, we have no evidence that any interface was more accurate than another, but

some of the interfaces were more efficient. When the time interval is smaller, we can conclude that animation is slower than small multiples and interactive timeslicing. Interactive animation is the only one of these interfaces where all timeslices of the time interval must be remembered in order to compare them. For both small multiples and interactive timeslicing, some of the time interval representation can be offloaded to the interface as multiple timeslices are shown. When the time interval is wider, we can conclude that small multiples is faster than both animation and interactive timeslicing. As the interval of time considered increases, interactive timeslicing is more strongly affected as this technique is based on individual timeslices. The implementation of small multiples from this experiment provides a more natural interaction with a time interval as all timeslices must be contiguous.

5.6 Task 3: Attribute Changes at Points in Time

This experiment tests performance when reading attribute values at multiple, disjoint, time points. In terms of research questions, we thought that interactive timeslicing would perform well here (**Q1** and **Q2**). We felt that this difference would increase for the far level (**Q3**).

Procedure. This third experiment's instruction was 'each pair of red (start) and blue (stop) lines signify a timeslice where a pink node is present. In which timeslice is the pink node smallest?'. The video figure in the supplementary material illustrates how participants answered this task on all three interfaces. A randomly selected node, present throughout the interval of interest, was given a minimum attribute value at precisely one of those timeslices and a maximum value in all others. Bright pink was chosen as the colour of the target node for contrast reasons to minimise visual search time. The standard node size is 3.3× bigger than the size of the smaller answer node. An example of a normal pink node vs the smallest pink node can be seen in Figure 5.10.

The procedure for submitting answers for this task was the same as **Exp 2**, with participants using a button to enter 'answer mode' and moving a timeslice, of fixed size, to their answer position.

Measurements. Time (s) and accuracy were measured for this experiment. There was only one correct answer where the attribute was at its minimum value. Therefore, a score of 1 was recorded for each task answered correctly and 0 for an incorrect answer.

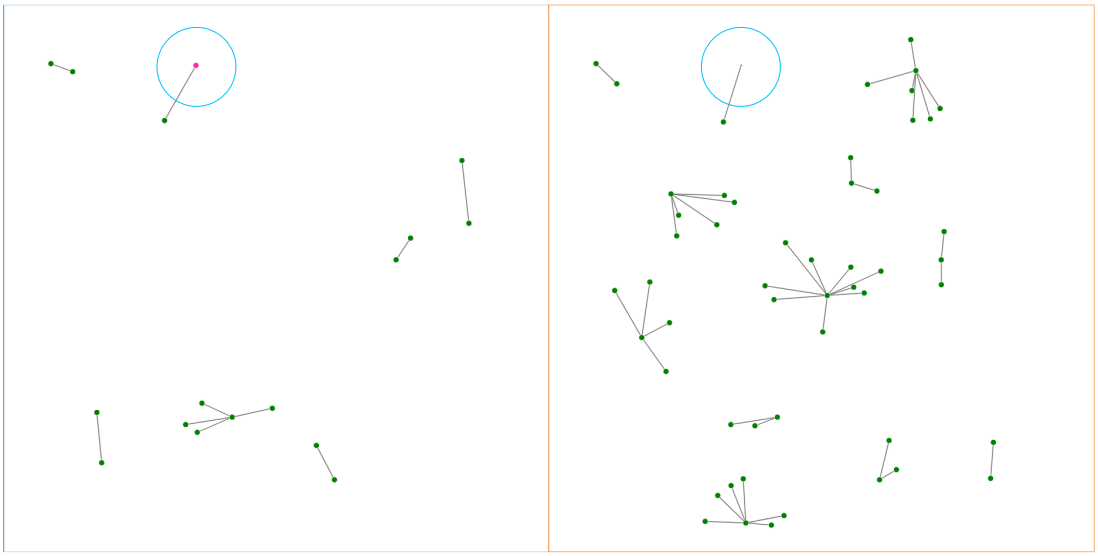


Figure 5.10: Two vignettes, as seen during task 3, with a standard size pink node (left) and a small size pink node (right). Pink nodes are highlighted with a blue circle for the purposes of readability, but this was not present on the experiment interfaces.

Results. Correctness and completion time are shown in Figures 5.11 and 5.12 respectively. After a Holm–Bonferroni correction, we found no significant differences between the interface conditions in neither the near nor the far condition. All interfaces had median 100% correctness for this experiment for both near and far conditions. In terms of completion time on near, all pairwise differences were significant with interactive timeslicing outperforming both small multiples by 6.3s (small multiples 1.19× slower) ($t = 4.45$, $df = 23$, $p < 0.001$) and animation by 13.6s (animation 1.42× slower) ($t = 8.52$, $df = 23$, $p < 0.001$), and small multiples outperforming animation by 7.3s (animation 1.19× slower) ($t = 4.80$, $df = 23$, $p < 0.001$). On the far condition, small multiples outperformed animation by 26.1s (animation 1.96× slower) ($t = 12.7$, $df = 23$, $p < 0.001$) and interactive timeslicing outperformed animation by 26.7s (animation 2.00× slower) ($t = 11.8$, $df = 23$, $p < 0.001$). The remaining pairwise difference between interactive timeslicing and small multiples was not significant.

Participant Survey. After completing the experiment, participants ranked the interfaces on a scale of 1 (best) to 3 (worst) in order of preference to complete the given task type, for full results see Fig. 5.6. There was no clear preference for a single interface for completing this task type. However, animation was intensely disliked with 87.5% participants giving it a rank of 3. There is a relatively low interaction cost for small multiples and interactive timeslicing

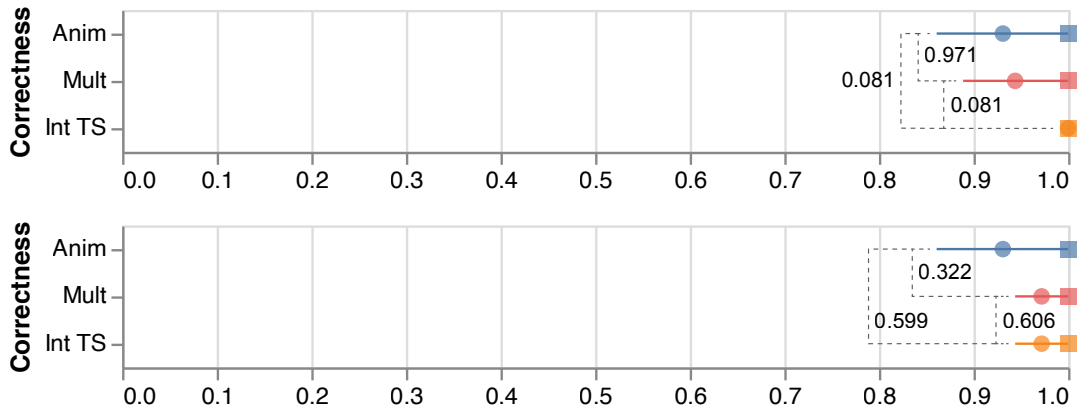


Figure 5.11: Experiment 3 correctness for the near (top) and far (bottom) conditions. The dot represents the mean, square the median, and 95% CIs shown. Pairwise lines indicate p-values with solid lines indicating significant differences.

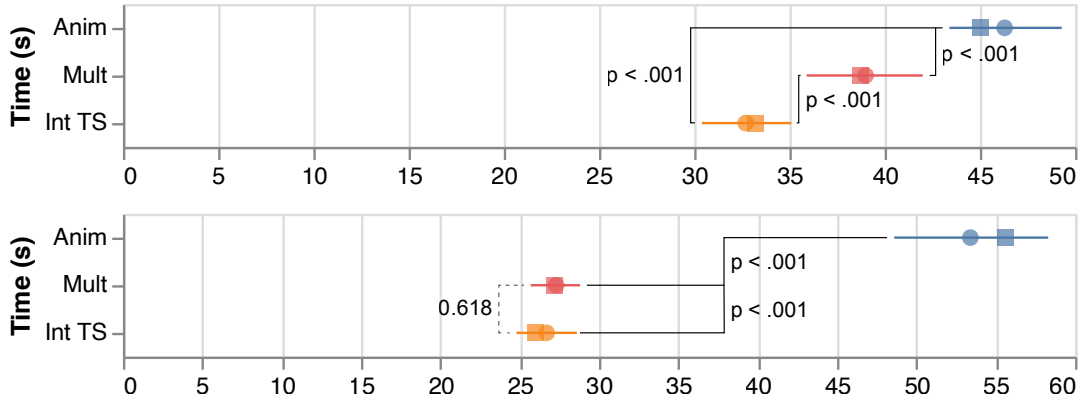


Figure 5.12: Experiment 3 completion time in seconds for the near (top) and far (bottom) conditions without log transform. The dot represents the mean, square the median, and 95% CIs shown. Pairwise lines indicate p-values with solid lines indicating significant differences.

for this task, whereas animation required participants to scroll across the timeslices. Interactive animation also required a much higher concentration level for this task type than the other interfaces, with participants having to identify the target node and then remember its position while animating the graph. The opposite is the case for interactive timeslicing and small multiples where all nodes and edges were always visible, meaning participants simply had to identify the pink node in each time window of interest rather than attempt to remember the previous node position.

Summary and Discussion. For this experiment, no significant differences were found in terms of correctness. Hence there is no evidence supporting superior accuracy of any of the interfaces. In terms of completion time for near, all pairwise differences were significant with interactive timeslicing the fastest, followed by small multiples, and then animation. As the near condition is closest to previous experiments [19, 102], small multiples outperforming interactive animation is consistent with this result. For far, we did not see a difference between interactive timeslicing and small multiples where there was one for near. However, animation is significantly slower than both interfaces. There could be many reasons why we did not find a difference between small multiples and interactive timeslicing. One possible interpretation is that the task is less demanding on participant memory (remembering a single node instead of a collection of edges), and thus, the difference is less pronounced. Further experimentation is required to test this hypothesis.

5.7 General Discussion

We first summarise the main results from the experiment based on the research questions from Section 5.2. Answers to the questions vary depending on the task, with Experiment 1 and 3 showing similar patterns but different from Experiment 2.

In Experiments 1 and 3, task completion times with the different techniques (Q1) show the clear advantage of interactive timeslicing over the other techniques. We take this to indicate that interactive timeslicing facilitates time navigation between discrete points, which is what the tasks from Experiments 1 and 3 have in common. There is one exception: in the far condition of Experiment 3, small multiples and interactive timeslicing showed similar completion times (i.e., we observed a difference between the near and far conditions — Q3). We attribute this to a reduced memory requirement for that condition. Occasionally small multiples visibly outperformed animation. In accuracy (Q2), interactive timeslicing also showed a clear advan-

tage over the other two techniques in Experiment 1, but not in Experiment 3, where accuracy was high for all interfaces and statistically indistinguishable.

Experiment 2 has very different results. Small multiples was faster than the other techniques in the far condition (Q1), although statistically indistinguishable from interactive timeslicing in the near condition. Accuracy measurements in this experiment show reasonably large differences in means, but the large variance in the trials prevents us from finding statistically reliable differences between interfaces in both near and far conditions (Q2). A possible explanation of why small multiples performed significantly better is that it naturally represents consecutive time windows that can be scrolled through easily.

When considering the survey data, no interface was preferred by the majority of participants for all tasks. Instead, interface preference is task-dependent. For the first and second experiments, the rankings did indicate clear interface preferences for completing those task types; interactive timeslicing and small multiples ranked first for those experiments, respectively. In contrast to Experiment 1 and 2, there was no clear preference for a single interface for completing the third experiment. From our observations of Experiment 3, there seemed to be fewer 'fat finger' [289] interaction mistakes with interactive timeslicing than in Experiment 2. This likely is because there was no requirement to move or resize a time window after the initial definition stage (barring participant error). One participant remarked that their version of a perfect interface would be small multiples with the ability to make selected vignettes appear and disappear, better to facilitate side-by-side comparison of highly distant time points.

Interactive animation is often slower and less preferred but with no difference in correctness, meaning it is a less efficient way of finding the correct solution. On Experiment 1, it is significantly slower than interactive timeslicing. On Experiment 2, it is significantly slower than small multiples. On Experiment 3, it is significantly slower than the other two interfaces. Thus, for tasks involving time navigation as tested in these experiments, we confirm some results of other experiments on dynamic data [18, 21, 56, 102, 255]. One possible conjecture for why it is slower is that the participant has no idea where to look in the animation when undertaking an exploratory task. Although there are some preliminary results [255], it remains an open question of whether animation works well for explanatory tasks where a presenter can point out regions of interest for another viewer to understand.

Brehmer et al. [56] compared the efficiency of animation and small multiples on mobile phones for animated scatterplots. This previous experiment found that small multiples was usually faster than animation with no difference in correctness. One could view our study as a

somewhat analogous test on large displays with a touch interface that confirms several results comparing the interactive animation and small multiples conditions.

As with all experiments, experimental design choices cause limitations in result interpretability. One such choice is that we prioritised counterbalancing the interfaces, not the experiments or near/far factor, to reduce experimental noise. Thus, later experiments might have had more tired participants, and earlier experiments had less training. Far tasks might have benefited from the experience of having done the near tasks first. Nevertheless, we hypothesise that it is unlikely that training or fatigue might have affected the interfaces differently. As we do not directly compare the performance of the near factor to the far factor this means that the effect of the potential confounds of task and factor order is minimised.

All trials in our experiment use parts of the same dynamic network data. Although this dynamic graph is long in time with much variability in the different time points and intervals of the different trials, it is important to test other datasets in the future for the sake of generalisability and to explore further which types of structures or values might affect the different interfaces.

In order to keep a reasonable experiment length, we only tested three task types, but understanding how these interfaces perform for a broader set of tasks (e.g., additional ones from [11] or [2,172]) would produce further insights. We only test our interfaces with node-link diagrams despite matrices being another popular method for representation [37]. Testing the interfaces with a wider range of graph representations and task types would ensure that results are more generalisable.

It is also important to remark that we also made specific design decisions in the implementation of the visualisation interfaces, usually to support fair comparison between techniques. For example, only four small multiples were visible at a time and scrolling was required, and the most basic form of interactive animation was used whereby individual events are controlled with a slider. Similarly, we used linear interpolation in our animations; staged animated transitions could be considered [26, 78]. Other design choices made in the development of the interfaces, such as the different colours on the background of the small multiples timeline and the set timeslice size within all interfaces, were to allow the exact answering of the questions posed to the users. The effects of some of these secondary design decisions have the potential to be important but are out of scope for this work. Future work manipulating additional parameters will be welcome and should compare and further extend our findings.

It is possible that the size of the display or the portion of the field of view that it covers

could explain some of the differences that we observed. Investigating how performance with these interfaces varies will provide generalisability, but it could also offer valuable insights to design improved variants that are even better.

In this chapter, we have presented the results of a series of experiments intended to formally evaluate methods to visualise dynamic networks on large touch displays. We were primarily interested in comparing interfaces for tasks that involve cross-time operations to see network structure and attribute variation. Two of our selected interfaces, interactive animation and small multiples, are already well studied in previous literature [18, 56, 102, 255]. Our third selected interface, interactive timeslicing, has not previously been experimentally evaluated.

Due to the lack of existing evaluations for interactive timeslicing, assessing interface performance with solo participants is a necessary starting point. However, large displays are commonly used in collaborative settings, and previous evaluations have shown the value of collaboration for graph exploration [239]. With this in mind, future experiments evaluating the usability of these interfaces for collaborative situations is vital.

We must also understand how interactive timeslicing compares to interactive animation and small multiples on standard desktop displays when the participant can use a mouse to interact with the representation, rather than the large touch screen display as tested in this chapter. It is possible that some of the advantages and disadvantages that we observed are significantly affected by the type of input (e.g., direct touch vs indirect mouse). Although we have justified testing with touch input as the more natural way to work on large collaborative displays, indirect inputs such as computer mice or touchpads are probably still a more common way to interact with dynamic network visualisations. We carry out an experiment in Chapter 6 to test the performance of interactive timeslicing against interactive animation and small multiples on personal displays with mouse input.

Chapter 6

Evaluating Time Navigation for Long in Time Dynamic Graphs on Personal Computers

Contents

When introducing a novel graph visualisation method, generalisability is a crucial component of the interface. While it is certainly possible to generate interfaces and methods which are specific to, and only work on, a handful of datasets, or work in very few particular situations, this is not efficient or ideal. Therefore the generalisability of any method across tasks, datasets, and hardware is essential.

Results presented in Chapter 5 demonstrate that interactive timeslicing outperforms interactive animation and small multiples for tasks involving distant time points. Therefore we must understand how our interactive timeslicing interface can generalise to standard displays with indirect mouse input. Large touch-screen displays are not easily accessible or available for the majority of users. This is particularly the case given the current pandemic which necessitates home working where possible.

We considered that interactive timeslicing would still outperform interactive animation and small multiples on smaller screens as the problem is not necessarily one of available screen real-estate. Therefore we adapt our interfaces from Chapter 5 to work on standard (sub 32") displays with full HD resolution (1920x1080) and mouse input and carry out a further study.

Our secondary requirement was that this experiment was run in a fully remote manner, but the required tutorial and general overhead meant that this was not an experiment suitable for distribution via unmoderated settings. Therefore, we ran the experiment via Zoom video conferencing software by giving participants control of the experimental machine [198]. We discuss the clear implications and learning opportunities for this choice in Chapter 6.7.

From the previous study results, detailed in Chapter 5, we have a loose idea of the type of tasks that interactive timeslicing performs well for on a large touch-screen display. However, we do not know if this superior performance will hold for standard displays with a conventional mouse input method. We hypothesised that it was likely that advantages inherent to interactive timeslicing would carry across, namely comparison of points that are not co-located in time; but also that some disadvantages of interactive timeslicing – such as in task 2, the so-called 'fat finger' problem – could be negated with the more accurate mouse-pointer. It was also likely that disadvantages inherent to small multiples and interactive animation, namely the memory and interaction costs, respectively, would hold. However, we were unsure if the advantage of small multiples for Experiment 2 would carry across on a smaller screen.

There are many existing experiments which test interactive animation against small multiples for normal displays with mouse input [18, 20, 21, 49, 102, 263, 271, 309], and for large wall displays with mouse input [41, 43], but as yet there have been no experiments which test

interactive timeslicing in either of these contexts. The majority of these experiments have focused on the structural properties of the network first and the temporal navigation second. For dynamic graphs that are long in time, no experiments have been run. We address this problem by formally evaluating approaches to interact with long in time dynamic graphs on standard displays with mouse input. We gained ethical approval for the experiments detailed in this chapter [1].

6.1 Experiment Interfaces

For this study, we consider the same three dynamic graph visualisation techniques as in Chapter 5: small multiples, interactive animation, and interactive timeslicing.

Interactive animation and small multiples are well-explored methods for visualising dynamic data as they allow the time dimension, a key component, to be encoded either in time, as in animation, or space, as in small multiples [37]. Multiple experiments have shown that small multiples generally outperform interactive animation for dynamic graph tasks in speed and accuracy.

We originally introduced interactive timeslicing in Chapter 3 and verified the general usability of the method across two evaluation sessions with three social science researchers, detailed in Chapter 4. The intention of interactive timeslicing was to address the primary disadvantage of small multiples; namely that it is tough to compare timeslices that are not co-located in time, and are therefore separated by some distance both in time and physically on the screen within the representation, and that by constraining timeslice size and position it means that important within-timeslice events may be lost.

Across all interfaces, node-link diagrams were used for all graph representations as they are popular in media and are well explored in literature [37]. All approaches were implemented to work on a display of any size with full HD resolution (1920x1080). Rather than reducing the maximum number of possible timeslices, we instead reduce the pixel size available for each timeslice and re-scale interfaces. In addition, all interfaces were modified to use a mouse rather than touch control as part of porting them from a touch-screen display design to a standard display with mouse input.

[1] All experimental material is available on the OSF at `https://osf.io/bdpnr/?view_only=8d2e29693b714b7d8bb4abd407ad8e56`.

6.1.1 Differences and Implications

We modify the interfaces shown in Figure 5.1 to fulfil our use case here. We uniformly re-scale the interfaces from 4k resolution (3840x2160) to full HD resolution (1920x1080) by changing width and height values in the underlying code. Therefore the primary difference is that some elements on the screen are physically smaller, though we theorise that this will have a minimal impact on the results given that participants are typically closer to their desktop displays than they were to our touch-screen display. In the first study, we frequently observed participants stepping back and fore with the large display to get an overview of the screen elements.

Also, we move from the direct touch interaction to an indirect mouse interaction method. Indirect mouse interaction is generally considered to have higher precision, and we hypothesise that this will have the most significant impact on the results of the second study. We also carry out this study in a fully remote manner via Zoom video conferencing software. Zoom allows us to give participants control over the experimenters machine while screen sharing, avoiding issues setting up the experimental software on each participants machine.

6.2 Experiment and Research Questions

Our general goal is to provide empirical evidence that can support the design of better interfaces for time navigation when exploring dynamic networks. We also want to understand the differences between interfaces on large touch-screen displays and interfaces on standard displays with mouse input. Thus, we designed three experiments, each testing a different task. For each experiment we seek to answer the following questions:

- Q1 Which interface will have the lowest completion times for the selected tasks?
- Q2 Which interface will be the most accurate for completing the selected tasks?
- Q3 How does distance in time between data elements of interest affect the performance with the different interfaces?
- Q4 How does each result compare to the study carried out on the large touch-screen display?

6.3 Experimental Approach

We describe our iterative process for experiment and task creation in Chapter 5.2. We use the same set of tasks and experiments here. Initially we explored existing dynamic graph task

taxonomies [2, 11, 172] and made use of our findings from previous observations of analysts working with dynamic networks [188].

As with the study reported in Chapter 5 we ran the three experiments with the same participants in the same session. We prioritise comparing results across interfaces on the same task; to reduce noise from order effects on our comparisons of interest, we run the three experiments in the same order for all participants (i.e., we do not randomise experiment order). Experiments took place in late October 2020 and took approximately 90 minutes to complete.

6.3.1 Participants

Twenty-four volunteers (16 male, 8 female, 18 and 45 years old) from our university participated in the study. Participants were offered a £10 Amazon voucher as an incentive. We chose to have the same number of participants as the previous experiment to allow comparisons and ensure proper counterbalancing. We used a simple questionnaire to screen participants without typical colour vision and those who did not have basic computer and mathematics experience. We also used the questionnaire to ensure that participants had screens with the correct resolution (1920x1080) and that they had an external mouse for input as we encountered usability problems using trackpads for input at the piloting stage. We did not require prior knowledge of networks.

Before beginning the first experiment, participants filled a short demographic questionnaire indicating their familiarity with networks and network graph visualisation using a 1–5 Likert scale. We also asked participants for the physical size of the display they were using and the maximum resolution of their display. Participants were also asked to ensure that their display resolution was set to 1920x1080. Six participants gave their familiarity as 1 (no knowledge), seven as 2, six as 3, two as 4, and three as 5 (high knowledge). When compared to the results of the first experiment (see Figure 6.1), we can see that participants of this study are more likely to rate themselves as having lower knowledge.

6.3.2 Apparatus

The apparatus and development method are as set out in Chapter 5. For this study, all three of our interfaces were implemented for a display of any size with full HD resolution (1920 by 1080 pixels) using mouse input.

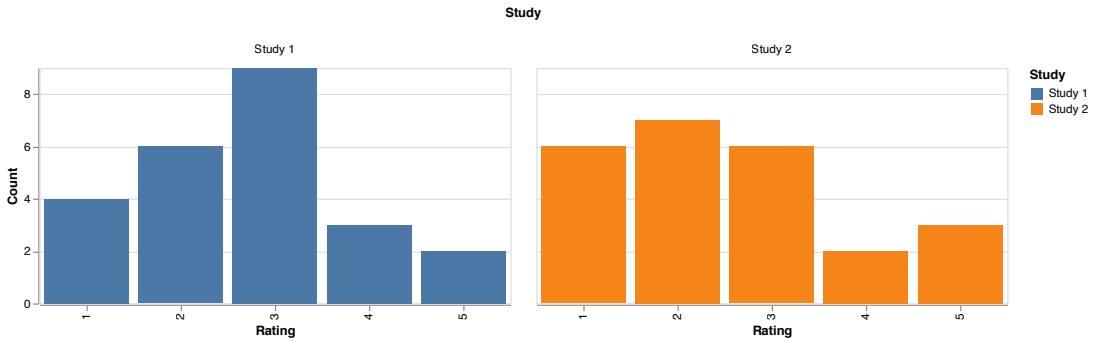


Figure 6.1: A comparison of participant responses regarding their knowledge of networks. We used a Likert scale where 1 denotes no knowledge, and 5 represents a high level of knowledge. On the left is the study detailed in Chapter 5, and the right is the study for this chapter.

6.3.3 Experimental Dataset and Graph Layout

The dataset used in these experiments is described at length in Chapter 5.3.3. We make no dataset changes for this experiment.

6.3.4 Experimental Design and Procedure

All three experiments share an identical structure in terms of factors, conditions, and repetitions. The main manipulation of interest is *interface*, with the three interfaces Anim, Mult, and Int TS as levels—see Chapter 5.1. One factor is temporal distance (near and far), which manipulates the separation of the target timeslices or the length of the time interval involved in the task. In the near condition, timeslices had a separation of 18 hours (three timeslices) while in the far condition timeslices had a separation of four days (16 timeslices).

We performed a within-participant design with participants completing three repetitions for each cell, for a total of 18 trials, as well as two additional easy tasks on each interface for training which was discarded before data analysis. Participants carried out trials (including training) in three blocks of 8, with 15-second breaks between each block. Possible ordering effects were counterbalanced using Latin squares for each participant, shown in Table 5.1 of Chapter 5. A participant always used the interfaces in the same order for each experiment that they completed. Trials in the near condition always took place before trials in the far condition. We did not counterbalance temporal distance because we are not interested in the quantitative comparison of performance between near and far conditions for the same interface.

Prior to the real trials of each interface condition, participants received a tutorial on the specific interface and carried out one trial example. Upon completion of each experiment, par-

ticipants ranked each condition on a scale of 1 (best) to 3 (worst), according to their preferred interface for completing the type of tasks tested. The experimenter also collected qualitative notes during the experiment. Participants had the opportunity to make other comments and share their thoughts about the tasks, interfaces, and interaction method.

6.3.5 Statistical Methodology

The statistical methodology was decided during the experimental design process and recorded before data collection. This experiment followed the same methodology as detailed in Chapter 5.3.5.

After applying a Shapiro-Wilk test with $\alpha = 0.05$ to each condition of near and far for each experiment separately, it was evident that most of our data had a non-normal distribution, as in the previous experiment. For this reason, our statistical analysis follows the same process outlined in Chapter 5.3.5.

The results are presented in the next section. In all result figures, blue corresponds to animation (Anim), red to small multiples (Mult), and orange for interactive timeslicing (Int TS). The mean is indicated using a circle and the median using a square. Error bars represent bootstrapped 95% confidence intervals computed using 250,000 repetitions. We report p-values to three decimal places in all figures [40, 126, 127, 326]. Solid lines indicate statistical pairwise significance after Holm-Bonferroni correction, and dashed lines indicate no statistical pairwise significance after correction.

6.4 Task 1: Graph Structure Changes at Points in Time

This study tests completion time and accuracy for tasks where the graph structure had to be compared at two separate time points. Our hypotheses for this task in the remote study were the same as those for the in-person study. We did not expect to see significantly different results to those recorded in the previous study.

Procedure. The task and procedure were the same as for the experiment detailed in Chapter 5.4, the only significant difference is that participants used a mouse for input rather than direct touch interaction with the screen. Participants were given the position of two timeslices and told to select edges in the first timeslice that did not appear in the second.

Measurements. Time (number of seconds) and accuracy were measured for this experiment. Accuracy involved comparing the set of edges selected by the user and the set of edges present in the correct answer of the question. In order to evaluate correctness in this case, we employ a method from pattern recognition and information retrieval. A perfect answer would have perfect *precision* (p) (no edges outside the correct answer are selected) and perfect *recall* (r) (all of the edges in the correct answer are selected). The equation used to score precision and recall is shown in Equation 5.2.

The value of p and r are defined in the following way. Consider two edge sets: the set of participant answer edges (X), and the set of correct answer edges (Y).

$$p = \frac{|X \cap Y|}{|X|} \tag{6.1}$$

$$r = \frac{|X \cap Y|}{|Y|} \tag{6.2}$$

Results. Correctness and completion time are shown in Figures 6.2 and 6.3 respectively. After a Holm–Bonferroni correction, we found no significant differences in terms of correctness for the near or far conditions. In terms of completion time, however, we find many significant differences. On near, interactive timeslicing outperforms animation with a difference of $28s$ (animation $1.24\times$ slower) ($t = 3.97$, $df = 23$, $p < 0.001$) and small multiples with a difference of $22s$ (small multiples $1.20\times$ slower) ($t = 2.72$, $df = 23$, $p = 0.012$). On far, interactive timeslicing outperforms animation with a difference of $26s$ (animation $1.25\times$ slower) ($t = 2.94$, $df = 23$, $p = 0.007$) and small multiples with a difference of $52s$ (small multiples $1.41\times$ slower) ($t = 5.32$, $df = 23$, $p < 0.001$).

Participant Survey. After completing the experiment, participants ranked each condition on a scale of 1 (best) to 3 (worst) to indicate their preference for conditions to complete the given task type. These ranking responses are in Fig. 6.4. For this experiment, 96% of participants indicated that they preferred the interactive timeslicing, with interactive animation second, and small multiples as the least preferred option. The primary criticism of small multiples was the same as in the previous experiment; namely, for tasks involving the far in time condition, it was impossible to put both timeslices on-screen simultaneously to compare them directly. Participants found it challenging to remember edge positions and edge presence when they had to scroll through many intermediate representations to reach the comparison target. The memory cost was also probably a factor in the ranking of animation. However, this was less

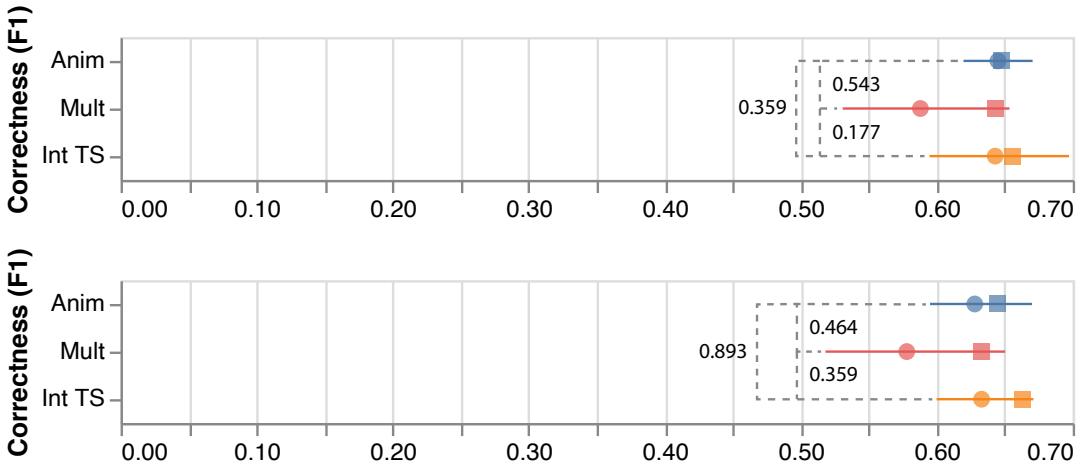


Figure 6.2: Experiment 1 correctness for the near (top) and far (bottom) conditions as computed by the F_1 score (Equation (5.2)). The dot represents the mean, square the median, and 95% CIs shown. Pairwise lines indicate p-values with solid lines indicating significant differences.

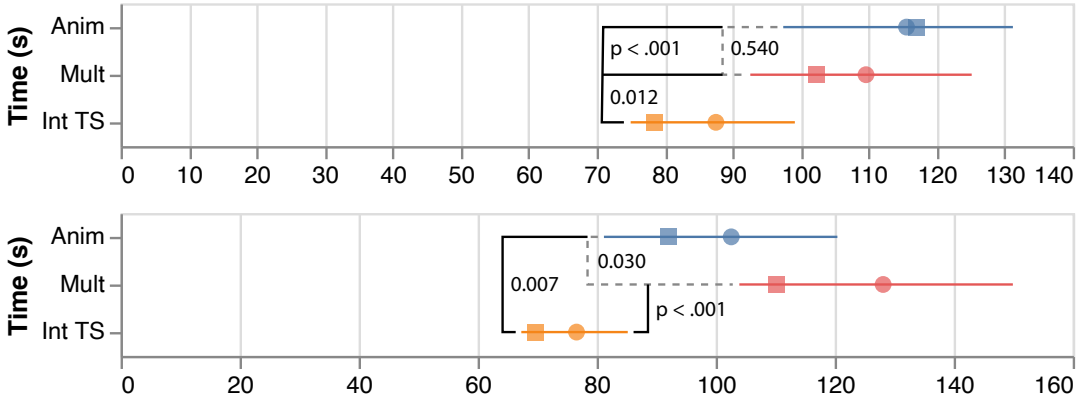


Figure 6.3: Experiment 1 completion time (s) in seconds for the near (top) and far (bottom) conditions without log transform. The dot represents the mean, square the median, and 95% CIs shown. Pairwise lines indicate p-values with solid lines indicating significant differences.

pronounced as the interactive animation interface made it simpler to switch backwards and forwards in time without viewing intervening timeslices.

Summary and Discussion. For both the near and far conditions, interactive timeslicing outperforms animation and small multiples in terms of completion time, but not correctness. Given our initial conjecture was that interactive timeslicing would outperform both other interfaces for both completion time and correctness, we were only partially correct. This also contradicts the results of our first study, where interactive timeslicing was better for both measures. Also,

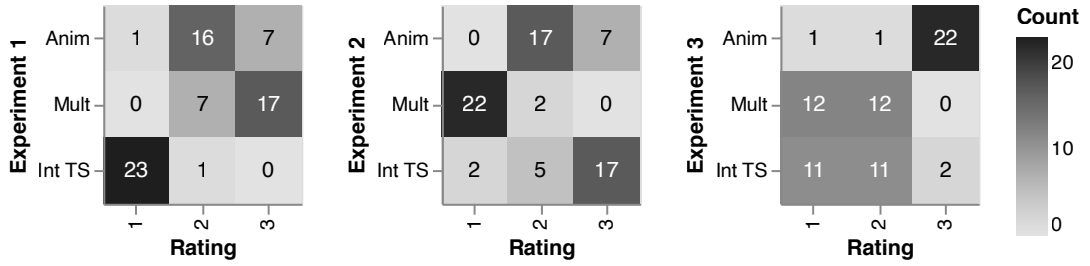


Figure 6.4: Participant interface rankings. Cells annotated by number of votes and darker cells correspond to more votes. Rating is 1st, 2nd, or 3rd.

animation significantly outperforms small multiples in terms of time for the far condition; a difference present in the first experiment but without a significance measure.

Interactive timeslicing was designed for tasks involving the comparison of disjoint time points, meaning it is particularly well-suited to this experiment. Animation incurs interaction costs for both the near and far conditions, whereas the interaction cost of small multiples appears to primarily disadvantage it for the far condition where a much larger number of interactions occur.

The lack of significant difference in any correctness measure is unusual compared to the previous experiment using the large, touch-screen, display. However, participants generally took longer to complete this task than in the previous experiment and so it is possible that the extra time spent translated to a more uniform correctness measure. The uniform correctness measures mean that each task was possible to complete with each interface, ergo each interface was usable for this task, and the primary differences were in user experience and completion time.

6.5 Task 2: Graph Structure Changes Over a Time Interval

This experiment tests how interactive timeslicing performs over a time interval when examining a change in graph structure. A time interval is several consecutive timeslices. Our conjecture for this experiment was the same as for Experiment 2 in the touch-screen display study. We did not expect interactive timeslicing to perform well due to its design not natively facilitating the comparison of multiple continuous time points. However, we hypothesised that the more precise indirect mouse interaction would improve the general performance of interactive timeslicing for this experiment.

Procedure. The task and procedure were the same as for the experiment detailed in Chapter 5, the only significant difference is that participants used a mouse for input and interaction rather than direct touch interaction with the screen. The on-screen prompt provided to participants for experiment two was: 'select a timeslice between the red lines where the cluster of purple nodes is most dense.'

Measurements. Time (s) and accuracy were measured for this experiment. Accuracy involved comparing the number of edges between purple nodes in the selected timeslice (n_s) to the timeslice where the number of edges is a maximum (the correct answer) (n_a). The equation that we used to do this is given in Equation 5.5.

The value of this measure is 1 when the correct answer is selected and diminishes to zero with a window of fewer and fewer edges between the purple nodes.

Results. Correctness and completion time are shown in Figures 6.5 and 6.6 respectively. After a Holm–Bonferroni correction, we found significant differences in correctness only for the far condition where small multiples outperformed animation by 12 percentage points ($W = 161.5$, $p = 0.005$) and small multiples also outperformed interactive timeslicing by 18 percentage points ($W = 418.5$, $p = 0.004$).

In terms of completion time on near, small multiples outperformed animation by $12s$ (animation $1.17\times$ slower) ($t = 4.36$, $df = 23$, $p < 0.001$) and interactive timeslicing outperformed animation by $13s$ (animation $1.19\times$ slower) ($t = 4.50$, $df = 23$, $p < 0.001$). On far, small multiples outperformed both animation by $23s$ (animation $1.26\times$ slower) ($t = 4.56$, $df = 23$, $p < 0.001$) and interactive timeslicing $38s$ (interactive timeslicing $1.37\times$ slower) ($t = -8.04$, $df = 23$, $p < 0.001$). For the far condition animation also outperforms interactive timeslicing by $14s$ (interactive timeslicing $1.14\times$ slower) ($t = -2.40$, $df = 23$, $p = 0.025$). No other significant differences were found.

Participant Survey. After completing the experiment, participants ranked each condition on a scale of 1 (best) to 3 (worst) to indicate their preference for conditions to complete the given task type. These ranking responses can be seen in Fig. 6.4. For this experiment, 92% of participants preferred small multiples for completing tasks of this type. Animation ranked second, and 71% of participants ranked interactive timeslicing as the worst.

Mirroring the previous study on touch-screen displays, participants mostly disliked using interactive timeslicing for this task. We hypothesise that this was due primarily to the amount

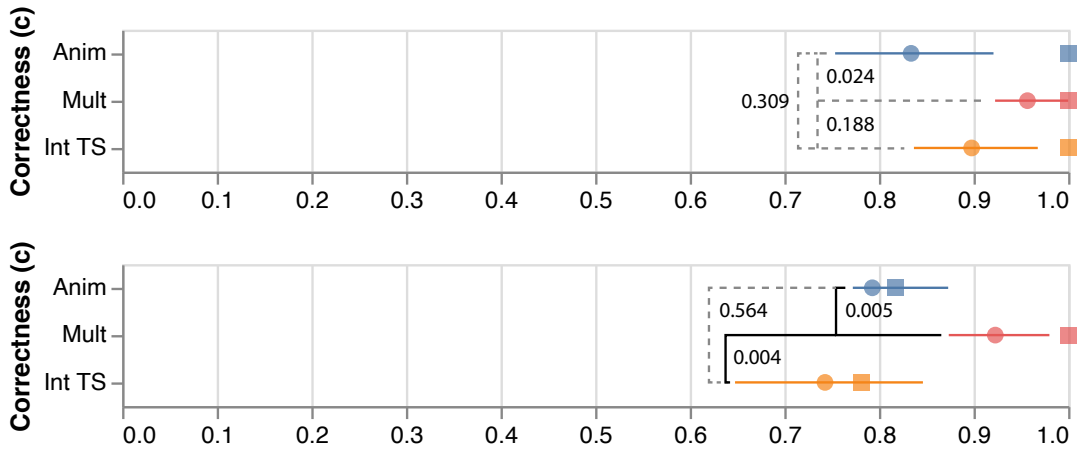


Figure 6.5: Experiment 2 correctness (Equation (5.5)) for the near (top) and far (bottom) conditions. The dot represents the mean, square the median, and 95% CIs shown. Pairwise lines indicate p-values with solid lines indicating significant differences.

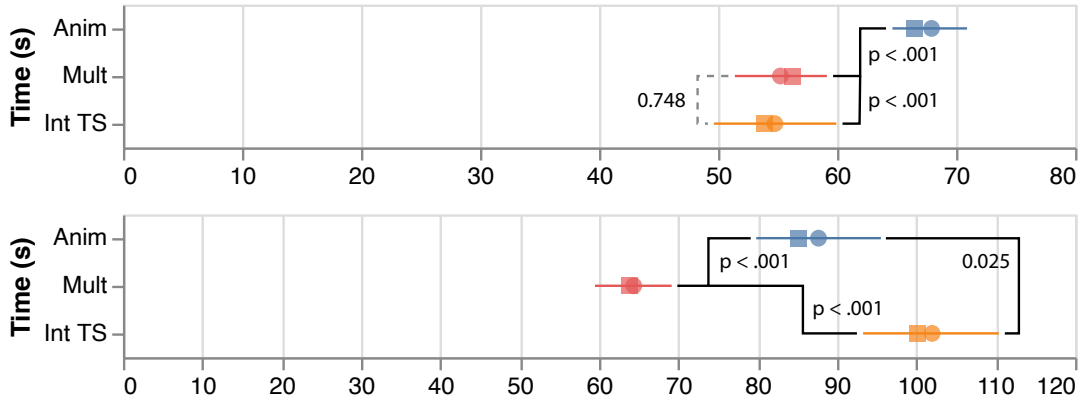


Figure 6.6: Experiment 2 completion time in seconds for the near (top) and far (bottom) conditions without log transform. The dot represents the mean, square the median, and 95% CIs shown. Pairwise lines indicate p-values with solid lines indicating significant differences.

of interaction required with interactive timeslicing, compounded by a small lag in the remote video conferencing software that meant that very precise mouse interaction was all but impossible. We discuss the presentation of this lag, and the broader implications for the results of this experiment, in Chapter 6.7.

Summary and Discussion. We found significant differences in correctness with small multiples outperforming both animation and interactive timeslicing for the far condition for this experiment.

Similar to the in-person experiment, animation is slower than both interactive timeslicing and small multiples in situations where the time interval is smaller. With smaller time intervals, interactive timeslicing does not require time windows to be moved, hence a greatly reduced interaction cost, and there is little to no memory cost to the participant.

However, where the time interval is wider, small multiples outperforms both other interfaces. For interactive timeslicing, this can be explained by both the number of interactions required and the memory cost of not having all timeslices visible. The implementation of small multiples from this experiment provides a more natural interaction with a time interval as all timeslices must be contiguous.

6.6 Task 3: Attribute Changes at Points in Time

This experiment tests performance when reading attribute values at multiple, disjoint, time points. In terms of research questions, we thought that interactive timeslicing would perform well here (**Q1** and **Q2**). We felt that this difference would increase for the far level (**Q3**).

Procedure. The task and procedure were the same as for the experiment detailed in Chapter 5, the only significant difference being that participants used a mouse for input and interaction rather than direct touch interaction with the screen. Participants were shown four timeslice positions and had to identify in which timeslice a specified hot pink node was at its smallest size.

Measurements. Time (s) and accuracy were measured for this experiment. There was only one correct answer where the attribute was at its minimum value. Therefore, a score of 1 was recorded for each task answered correctly and 0 for an incorrect answer.

Results. Correctness and completion time are shown in Figures 6.7 and 6.8 respectively. After a Holm–Bonferroni correction, we found no significant differences between the correctness of the interface conditions for either the near or the far condition. All interfaces had median 100% correctness for this experiment for both near and far conditions. In terms of completion time on near, all pairwise differences were significant with interactive timeslicing outperforming both small multiples by $5s$ (small multiples $1.11\times$ slower) ($t = 2.40$, $df = 23$, $p - value = 0.025$) and animation by $13s$ (animation $1.24\times$ slower) ($t = 5.97$, $df = 23$, $p < 0.001$) Small multiples also outperforms animation by $8s$ (animation $1.15\times$

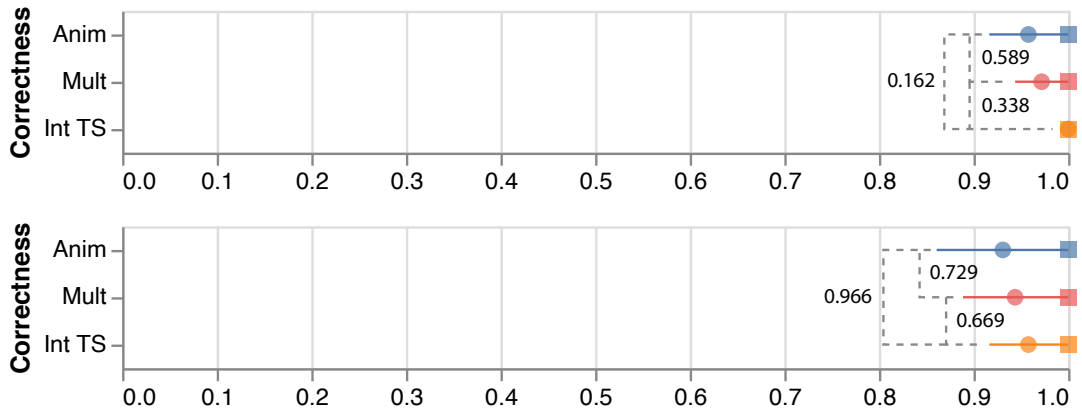


Figure 6.7: Experiment 3 correctness for the near (top) and far (bottom) conditions. The dot represents the mean, square the median, and 95% CIs shown. Pairwise lines indicate p-values with solid lines indicating significant differences.

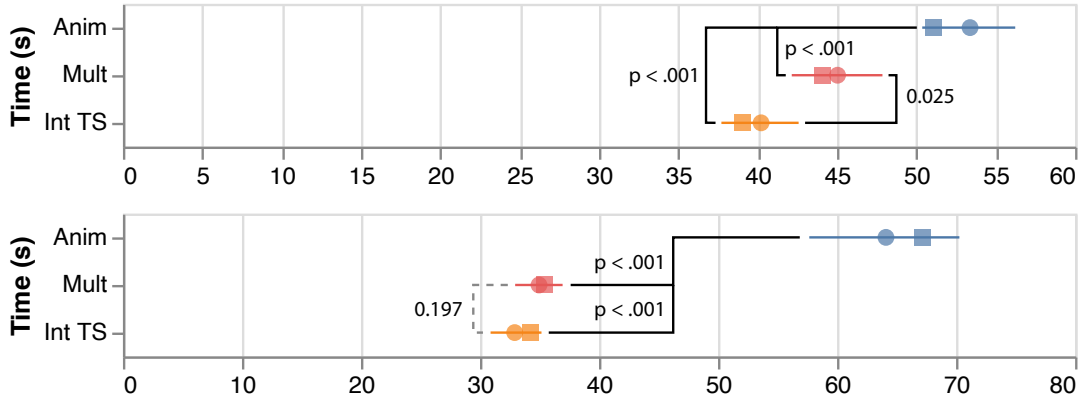


Figure 6.8: Experiment 3 completion time in seconds for the near (top) and far (bottom) conditions without log transform. The dot represents the mean, square the median, and 95% CIs shown. Pairwise lines indicate p-values with solid lines indicating significant differences.

slower) ($t = 3.88$, $df = 23$, $p < 0.001$). On the far condition, small multiples outperformed animation by $29s$ (animation $1.45\times$ slower) ($t = 10.99$, $df = 23$, $p < 0.001$) and interactive timeslicing outperformed animation by $31s$ (animation $1.48\times$ slower) ($t = 12.38$, $df = 23$, $p < 0.001$). The remaining pairwise difference between interactive timeslicing and small multiples was not significant.

Participant Survey. After completing the experiment, participants ranked the interfaces on a scale of 1 (best) to 3 (worst) in order of preference to complete the given task type. For full results see Fig. 6.4. There was no clear preference for a single interface for completing

this task type. However, animation was strongly disliked with 92% participants giving it a rank of 3. Reflective of the study using the touch-screen display, there is a relatively low interaction cost for small multiples and interactive timeslicing for this task, whereas animation required participants to scroll across the timeslices. Interactive animation also had a much higher memory cost for this task than the other interfaces; participants had to identify the target node and then remember its position while simultaneously interacting to animate the graph.

Summary and Discussion. For this experiment, no significant differences were found in terms of correctness. Hence there is no evidence supporting superior accuracy of any of the interfaces.

In terms of completion time for near, all pairwise differences were significant with interactive timeslicing the fastest, followed by small multiples, and then animation. As the near condition is closest to previous experiments [19, 102], small multiples outperforming interactive animation is consistent with this result. For far, we did not see a difference between interactive timeslicing and small multiples where there was one for near. However, animation is significantly slower than both interfaces. There could be many reasons why we did not find a difference between small multiples and interactive timeslicing. One possible interpretation is that the task is less demanding on participant memory (remembering a single node instead of a collection of edges), and thus, the difference is less pronounced.

6.7 Discussion

We first summarise the results from this experiment based on the research questions defined in Section 6.2. As with the previous study, Tasks 1 and 3 show similar patterns, while Task 2 differs.

For Tasks 1 and 3, task completion times with different techniques (Q1) show the strength of interactive timeslicing over the other interfaces. Given that the high-level shared characteristic of Tasks 1 and 3 is that they require the comparison of separate points in time, we take this as an indication that interactive timeslicing is superior for tasks involving this comparison. The one exception to this is the far factor of Task 3, where small multiples and interactive timeslicing have similar completion times (Q3). This is likely a mixture of the interaction cost inherent in interactive timeslicing, and the reduced memory cost for this type of single item comparison with small multiples. For Task 3, small multiples also outperformed interactive

animation (Q1) for both near and far factors; we hypothesise that this is because participants had to scroll through all of the timeslices between the first and last in the interactive animation interface.

Reflective of the first study, Task 2 shows a very different pattern of results. In terms of time, small multiples is faster than both other techniques for the far factor (Q1) but is not statistically significant compared to interactive timeslicing on the near factor. Unlike Tasks 1 and 3, and the previous study, we find significant differences in correctness for Task 2 though only in the far factor. In this case, small multiples outperforms both interactive animation and interactive timeslicing (Q2). For the latter, this is likely due to the technique's memory cost, whereas for the former, it is potentially a mixture of the memory cost and the high level of concentration required to accurately count the number of edges appearing within an animated timeslice.

Interactive animation is often slower than the other interfaces – i.e. it is never the fastest interface for any task – and is also less preferred. There is only a difference in terms of correctness for Task 2, meaning that interactive animation is simply a less efficient way of finding a correct solution for tasks involving the comparison of multiple separate time points. For tasks involving the continuous comparison of all events within a timeslice as in Task 2, however, interactive animation is also less accurate and less efficient. On Task 1, it is significantly slower than interactive animation for both the near in time and far in time factors. For Task 3, it is significantly slower than both small multiples and interactive animation across both time factors. This is supported by the results of other experiments for dynamic data [18, 102].

While there is previous work comparing animation and small multiples on standard displays with mouse input [18, 102], none of these experiments test interactive timeslicing or a similar operation. This study confirms the generalisability of our time navigation technique introduced in Chapter 3; the results of our two studies indicate that interactive timeslicing is the most efficient interface for navigating between discrete points in time on both wall-mounted touch screen displays and standard desktop displays with mouse input.

In this chapter, we have presented the results of a series of tasks to formally evaluate methods of visualising dynamic networks on standard displays. The study presented in the chapter is a natural extension of the study described in Chapter 5 and has important implications for the generalisability of our findings regarding time navigation techniques for dynamic network data.

6.8 Limitations

Some aspects of the study design do limit the interpretability of the results. Participant fatigue may have impacted later results as we chose to prioritise counterbalancing the interfaces, not the task or time factor, to reduce experimental noise. This may also have impacted the earlier results as participants will have had less training and may not have been fully comfortable with the interfaces. This latter limitation may have been expressed further in the results for the far in time factor as participants will have benefited from completing the near in time factor first. Despite this, we do not hypothesise that training or fatigue impacted the interfaces differently.

We test the same three task types and representations as the previous study to ensure generalisability of the results. Further work should test our interfaces for a broader range of tasks [172] and with more than one graph representation type, such as matrices [118,227,250].

One aspect of the remote nature of this study introduced a particular set of frustrations. When screen and control sharing over Zoom, regardless of if the two machines are on the same local network and regardless of network speed, there is a small but costly lag between the controlling user (in this case, our participants) moving their input device and the mouse icon moving on the screen. Participants were frequently frustrated by this phenomenon, particularly for interfaces and tasks where precise control was needed.

The main casualty of this, similar to the 'fat finger' [289] problem in the previous study, was interactive timeslicing during the second task. Moving timeslices led to some errors for this task, but resizing previously defined timeslices was by far the most time-consuming and error-prone operation. Participants also struggled with the edge selection necessary for Task 1, though this was equally distributed across interfaces and factors, so there is not much within-task variation evident. We see the overall impact of the 'Zoom lag' when comparing the correctness results for Task 1 between the two studies, and when comparing the time taken for all tasks across both studies. Eventually, we plan to carry out this study in an in-person setting to understand the real impact of this. Carrying out this study in person will also ensure that every participant uses the same hardware and a standardised input device. Interestingly, the lag is apparent only to the participant, and from the other side of the screen (i.e. to the researcher), everything generally looks fine. Losev et al. [198] discuss their use of screen control via Zoom for visualisation design and collaboration at length. They either do not encounter this problem, do not find it problematic, or develop suitable mitigation strategies. Collaborative visualisation design is a somewhat different context to completing an experimental task alone, and this change of scenario may be responsible.

A secondary factor is that the hardware used by participants was not standardised. We required participants to set their screen resolution to Full HD before starting the study, but there was a considerable variation in physical display size. As well as the physical display size, the physical display composition may have an impact. For example, an OLED panel is generally considered superior in brightness and sharpness to an LCD panel, and participants may have had very different brightness and contrast settings which could have had some effect on target finding for Task 3. That said, the considerable variation in hardware used by participants also offers an extra argument for the generalisability of our findings. Given that we measure roughly the same statistical pairwise differences as our first study, it is likely that hardware differences did not significantly impact the results of this study.

We did not control for mouse sensitivity, where a 1 inch (physical) mouse movement with one machine might translate to a 3-inch pointer movement on the screen, but with another machine, this could translate to 0.5 inches on the screen. Participants with less sensitive mice would need to physically move more for the same effect, likely increasing completion time and physical fatigue. However, participants with more sensitive mice may have had an issue with correctness, and therefore this could have increased their completion time; especially given the control problems exacerbated by the observed lag.

Remote visualisation research introduces very specific problems and potential confounds [46, 198]. In future, this study should be repeated but in an in-person context on standardised hardware. We can then understand the true impact of these limitations on our results.

Chapter 7

Discussion

Contents

The primary research question that we have worked towards addressing is:

- What are useful representations and interaction techniques that can improve the visualisation of long-in-time dynamic networks?

We chose to primarily design and test on large vertically-mounted touch-screen displays as existing work shows that physical navigation and embodied interaction techniques offer performance benefits for most users [10,32,218], and that limitations of the visualisation begin to shift away from the display and towards the user. Large displays are also able to facilitate collaborative analysis in a way that is challenging for small displays [158], and they also allow users to step back and fore for a more natural zoom experience [160]. For interaction with the large display, we primarily support direct touch input as it generally performs on-par with mouse input for most tasks [268], but touch is better able to facilitate collaborative working in groups [159].

7.1 Dynamic Network Plaid

Our first contribution is the new interface for the visual analysis of dynamic network data that we refer to as 'the Plaid'. The Plaid maps time to a dimension in space, as a small multiples representation would, but it does not automatically timeslice the data and generate the corresponding node-link diagrams. Instead, we hand control of timeslicing entirely to the user and also allow them to select any parts of the time range to compare directly. The majority of work on visualising dynamic data using time-to-space representations (see Chapter 2.3.4) focuses on timeslicing and displaying the whole graph in a continuous manner, meaning that tasks requiring the comparison of points that are distant in time become unnecessarily challenging. We use an area timeline to encode rates of activity at each time point as an entry point to the visualisation.

In the Plaid, we also prioritise the consistent visualisation of interaction provenance. The analysis tasks involved with large in time dynamic data generally result in complex interaction stacking and, without some form of provenance data, this can make it very challenging and time-consuming to reproduce, explain, or repeat a previous analysis. Very few dynamic graph visualisation systems provide information about interaction provenance, let alone maintain its presence on the screen as a constant feature, despite generally positive results reported for systems with integrated interaction provenance [68,93,140,244,245,304]. Within this feature,

we also provide support for interaction cascades – the user can complete their analysis on one part of the graph and then duplicate all steps instantly by defining a new timeslice.

The large vertically-mounted touch-screen display provides extra screen real-estate that we can leverage for the display of the interaction steps and corresponding states of the visual analysis. We know, from previous work [10,32,218], that large displays offer performance benefits for most users and facilitate collaboration [158], assisted by the touch-screen interaction which is also beneficial for groups of users carrying out analysis on the same interface [159].

In this system, we also facilitate multiple views of the graph via node-link diagrams, matrices, and scatterplots. Node-link diagrams are superior for tasks involving pathfinding or following and are more accessible for non-expert users [118, 227, 250], but they suffer from problems with occlusion and are not ideal for large or dense graphs. As a mitigation strategy for these issues, we also support matrices as they perform well for basic tasks regarding single graph structure aspects [227] and are more suited to dense graphs [118]. Few existing dynamic graph visualisation systems integrate and coordinate multiple view types, despite this being highlighted as a mitigation strategy for the separate drawbacks of node-link and matrix representations [143, 144].

To address the principal handicap of timeslicing, namely the flattening of within-timeslicing events and loss of event order [290], we also provide a per-timeslice view of individual events and their order of occurrence. While admittedly a simple visualisation technique, it is unusual to see the integration of event-based visualisation alongside timesliced dynamic graph visualisation in existing work. During the evaluation of the system, we also saw that our participants used this additional view where they felt that there was an event-order effect on the graph structure within a timeslice. A secondary strategy that we used to preserve event order was to offer a view where nodes were coloured, using a light grey to black scale, based on the time of their final appearance in the timeslice. However, this was an imperfect solution as nodes can occur at multiple points in a timeslice, and we found that our event-order timeline-style feature was more useful for participants who took part in the evaluation sessions.

As some of our user-defined timeslices were large and therefore dense in terms of events and actors, we knew from previous work that the node-link diagrams would be less readable for some tasks [282]. We also recognised that user-defined timeslicing was a sub-optimal solution for tasks where users would want to see other time points at which specific nodes, or groups of nodes, appear. To support this we introduced functionality where users could select a node or groups of nodes, and 'flag lines' would appear on the main timeline to show not only the

position in time of other appearances of the selected items, but we also coloured these flags according to the node attribute at the flag time. This is analogous to provenance situations where users select a pattern or other visualisation aspect, and the system highlights where else the selected items appear [83, 216, 322, 327].

Selecting nodes also highlighted them in the event-order visualisation timeline. As we deal with a directed graph with source and target nodes, it was important that, for this feature, we differentiated between times that a node was a source and times that a node was a target. To achieve this, we encoded source and target status on the y-axis; with points at which a node is a target placed below the middle timeline of all events, and source node event points raised above the middle timeline. We also encode the primary node attribute here as colour – a single actor can make multiple posts with different primary attributes within a user-defined timeslice. These different posts might have different engagement as a target node depending on their primary attribute. By encoding the attribute as colour, we further facilitate the analysis and understanding of within-timeslice events. This is our primary contribution to the central ethos of the thesis as the Plaid introduces a number of novel representations and also leverages the interaction method afforded to us by the large touch-screen wall display to present a new mechanism for time interaction and selection in the context of dynamic network visualisation and analysis.

7.2 Evaluation of Dynamic Network Plaid with Public Health Experts

In Chapter 4 we describe the evaluation of the visualisation system that we introduced in Chapter 3. This took the form of two domain situation evaluation sessions with the intended target users of the Plaid. We observed that the main activity timeline was a good jumping-off point for the participants, particularly in situations where they were somewhat unsure of where in the data they may find insight. Participants were frequently concerned with comparing the network structure during areas of high activity to areas of low activity. These areas of activity were expressed on the activity timeline as peaks and troughs. We observed that they would select several temporally distant points in time to compare, an activity not well supported by either small multiples or animation for long in time dynamic graphs, with the initial intention of comparing the entire graph structure at these points. However, after a quick overview of the entire graph structure, our participants would focus only on subsets of the graph within

each user-defined timeslice. Kerracher et al. [172] identify a non-trivial number of graph tasks that can be described, at a high level, as this type of operation; despite very few dynamic graph visualisation approaches actively supporting the comparison of graph structure at disjoint, user-selected, points in time. Therefore, we find evidence in support of the aim of the thesis for useful interactions and representations of long-in-time dynamic network data.

Further to the central point of the research, we also observed our participants using the Plaid to follow a single node over time [13, 172]. The features that we introduced for selection and linking were vital here as they facilitated the quick completion of complex tasks. Our choice to provide alternative representations of the graph, in an attempt to mitigate the separate drawbacks of node-link and matrix-based representations, was also useful for the evaluation participants. They were able to use the scatterplot representations to hypothesise that actor activity was related to the time of day. We take this to indicate that providing multiple representations of the same data was a good design decision.

The second session demonstrated the value and importance of a tightly integrated event-based visualisation within the wider dynamic graph visualisation system. On eight occasions, we observed participants querying a single actor's activity distribution within the user-defined timeslice. The point in time at which an image was posted may explain why some networks appeared denser than others due to an artefact intrinsic to timeslicing. If a post is made at the beginning of a timeslice, it is likely to have received more likes/comments than one that is posted at the end of a timeslice. This feature allowed participants to hypothesise in more depth about the reasons for the network structure in their selected timeslices.

During the first session, we initially observed that the analysts were less inclined to work collaboratively when they were unsure of the data and the interface – in this case they preferred to observe others carrying out analysis. After their first observation, however, the second analyst chose to use an existing mid analysis point from the first participant as the jumping-off point for their work. This is also the point at which participants began to work collaboratively, with the first participant explaining their analysis thoughts in greater detail to the second participant; and the third participant theorising about the things that they might see in the data if they took a slightly different analysis path. The large display made it easy for participants to gather around and point at specific areas of interest. The multi-touch display also meant that no one person had control of the input method, and participants were more inclined to try out avenues of analysis that they would have otherwise discounted. For the second session, analysts chose to work collaboratively from the start. We observed them making use of the interaction

provenance to both repeat and avoid repeating some analyses, though they highlighted that perhaps a tree-based display of interaction provenance might be easier to navigate at the end of a multi-hour analysis session [68, 89, 93, 140].

Despite animation generally performing poorly for complex graph analysis tasks [102] this was a feature requested by our participants. We did not encounter any situations where they felt it was necessary to compare many consecutive timeslices, and perhaps this is why no request was made for a small multiples-type interface, or for algorithmically continuously timesliced data. While this validation was somewhat limited, it was able to demonstrate how the Plaid can facilitate the completion of complex graph tasks [172], event-based analysis [282], and the uses of interaction provenance encoded within the visual analysis system [140, 244, 245], adding to the universe of useful representations and interactions for the visual analysis of long-in-time dynamic network data.

7.3 Time Navigation on Large Touch Displays

To further validate the novel method of time navigation that we introduced in the Plaid, user-defined interactive timeslicing, we carried out the first of two experimental studies to validate this against small multiples and interactive animation. These studies also enable us to recognise the validity of our contribution to interaction techniques with the novel method of time navigation. Multiple pieces of existing work experimentally compare small multiples and interactive animation for dynamic data [18, 102], but none also include interactive timeslicing. We had three research questions:

- Q1 Which interface will have the lowest completion times for the selected tasks?
- Q2 Which interface will be the most accurate for completing the selected tasks?
- Q3 How does distance in time between data elements of interest affect the performance with the different interfaces?

and chose three common graph tasks to form our experiment tasks around:

- E1 Detecting graph structure changes at discrete points in time.
- E2 Detecting graph structure changes over a time interval.
- E3 Detecting single attribute change at discrete points in time.

From these sets of research questions and tasks, we were able to form several hypotheses in conjunction with previous work.

For **Q1**, we expected interactive timeslicing to outperform small multiples and interactive animation for tasks *E1* and *E3*. Further, and based on previous work [18,102], we hypothesised that small multiples would outperform interactive animation for all tasks. In terms of correctness, **Q2**, we did not anticipate any meaningful differences between interfaces for any task. After pilot studies, however, we revised this and instead conjectured that interactive timeslicing would outperform both small multiples and interactive animation for **E1**. For the time factor, **Q3**, we considered that the far-in-time factor would increase completion time across all interfaces and all tasks, but that there would be a particularly large increase in completion time for small multiples and interactive animation on *E1* when compared to interactive timeslicing.

For *E2*, however, we were generally unsure of the impact of a different task type – the comparison of multiple continuous timeslices – on interactive timeslicing. Due to the level of interaction required to complete this task with interactive timeslicing, we hypothesised that interactive timeslicing would be slower than small multiples (**Q1**), particularly for the far-in-time factor (**Q3**), but that there would be no meaningful difference in correctness between any of the interfaces (**Q2**) due to previous research [56] generally finding that small multiples is quicker but not more correct than animation.

Supporting our first hypotheses, we found that for *E1* and *E3* interactive timeslicing significantly outperformed small multiples and interactive animation in terms of completion time. Also, interactive timeslicing outperforms small multiples and animation for the correctness measure on *E1*, contrary to our expectations. We interpret this as an indication that interactive timeslicing is better able to facilitate time navigation between multiple discrete points. The only exception here is that there is no significant completion time difference between small multiples and interactive timeslicing on the far condition of *E3*, which we attribute to a reduced memory requirement for this task.

Regarding the time factor, we see that the far-in-time factor generally has much higher completion times than the near-in-time across all tasks and interfaces (**Q3**). This is particularly apparent for *E1* with small multiples and interactive animation.

For *E2*, as expected, we see no significant differences in correctness across the interfaces and factors, supporting previous work [56]. For completion time, however, the story is somewhat different. Our hypothesis that interactive timeslicing would be slower than small multiples (**Q1**) was partly correct as this only holds for the far-in-time factor (**Q3**). An unexpected

result, however, was that interactive timeslicing was quicker than interactive animation for the near-in-time factor (**Q1, Q3**) but that interactive animation was faster than interactive timeslicing on the far-in-time factor (**Q1, Q3**), though this difference was not statistically significant. We theorise that this was related to the 'fat finger' [289] problem that our participants encountered when moving the previously defined time windows for *E2* as this was the only task where this operation was required. There is some argument to be made that this result is an artefact of deficiencies in the implementation method of interactive timeslicing, and we did expect this result (animation faster than interactive timeslicing) to be reversed when we ran the second study using the more precise mouse-pointer interaction method.

Based on the result of this experiment, we find that interactive timeslicing is more suitable for tasks requiring the comparison of distant points in time when compared to small multiples or interactive animation. Secondarily, we find that small multiples generally outperforms interactive animation, supporting previous experiments comparing the two [18, 21, 56, 102, 255]. One possible conjecture for why interactive animation is slower is that the participant has no idea where to look in the animation when undertaking an exploratory task. Although there are some preliminary results [255], it remains an open question of whether animation works well for explanatory tasks where a presenter can point out regions of interest for another viewer to understand.

7.4 Time Navigation on Standard Displays with Mouse Interaction

To ensure the generalisability of the interactive timeslicing method for time navigation, we also carried out an experimental evaluation of the interface on standard displays with indirect mouse-pointer input. Ensuring that the method generalised to different display and input types was key, especially given that large vertically-mounted touch displays are specialist hardware that might not be available to all users. The second study uses the same protocol as the first study to facilitate the comparison between the two. We hypothesised that pairwise significance would hold across the two studies, except for *E2* where we expected that the more precise nature of the mouse-pointer would bring interactive timeslicing closer to small multiples in terms of completion time for the far-in-time factor.

For *E1* we find no significant differences in correctness between any of the interfaces or the factors. Based on the results of the first study we were expecting interactive timeslicing

to have a significantly greater correctness measure than the other two interfaces, though this does align with our very first hypothesis from the first study that there would be no significant difference in correctness between the interfaces for *E1* (**Q2**). For completion time, our results mirror those of the first study; with interactive timeslicing significantly outperforming small multiples and interactive animation for both near-in-time and far-in-time factors (**Q1, Q3**).

In *E2*, we obtain results that are contrary to our original hypothesis that there would be no meaningful difference in correctness between any of the interfaces (**Q2**). We see that small multiples significantly outperform interactive timeslicing and interactive animation in terms of correctness for the far-in-time factor. This is contrary to existing work which finds that small multiples are generally more efficient than animation but that there is no significant difference for correctness [56]. Participant strategies for completing *E2* were generally different compared to the strategies used by participants of the first study, and we attribute this significant difference to this aspect.

With respect to completion time, our hypothesis that interactive timeslicing would be slower than small multiples (**Q1**) was again only partly correct as this only held for the far-in-time factor (**Q3**). Similar to the first study, we saw that interactive timeslicing was quicker than interactive animation for the near-in-time factor (**Q1, Q3**) but that interactive animation was faster than interactive timeslicing on the far-in-time factor (**Q1, Q3**), and for this study the difference was statistically significant. This poor performance by interactive timeslicing was unexpected with the more precise mouse-pointer input, though there was a costly lag because the experiment was carried out by using Zoom to give participants control of the experimenter's machine. It is probable that this lag between a participant moving a mouse and seeing the resultant movement on-screen negated any benefits of the generally more precise mouse-pointer interaction method.

The final experiment, *E3*, had no significant differences in error rate, with median correctness of 1 across all interfaces and both time factors (**Q2**). For completion time, we see a similar pattern to the first study with interactive timeslicing significantly outperforming small multiples and interactive animation (**Q1**) for the near-in-time factor (**Q3**), and interactive timeslicing and small multiples outperforming interactive animation for the far-in-time factor (**Q3**). Returning to the near-in-time factor, small multiples also significantly outperforms interactive animation here. We theorise that this result is due to the reduced memory cost of *E3* when compared to *E1*, and so small multiples perform relatively well as there is no onerous comparison of the graph structure of two complete timeslices.

7.5 Comparing the Expert Evaluation and the First Study

During the evaluation of the Plaid, Chapter 4, we frequently observed the analysts using the interactive timeslicing feature as a mechanism for the comparison of multiple points that were separated by some distance in time. Indeed, this operation was nearly always their entry point to analysis and thinking about the contrast between several different points in time, generally related to areas of high and low overall network activity, was intuitive to them.

> "I would like to see what is the difference between this day [the largest Wednesday peak] and any other Wednesday."
>
> "I would also like to see what makes a Tuesday afternoon different to a Wednesday afternoon, and what happens on a Sunday."

We observed that the time navigation feature, in particular, was useful for our analysts, but we were unsure if it would experimentally outperform existing methods of time navigation – small multiples and interactive animation – and felt that a formal experiment would also allow us to make more accurate judgements about the generalisability of the Plaid.

From the results of the first study, using the touch screen vertically-mounted display, we find that interactive timeslicing is best suited to comparing points that are distant in time, but for the comparison of continuous timeslices, small multiples are still the superior solution. To compare distant points in time within a simple graph task, such as the comparison of only one highlighted node across timeslices, we also find that small multiples are an acceptable solution. For more complex tasks, however, interactive timeslicing is the most suitable choice. This is likely because interactive timeslicing allows users to place their target timeslices on the screen simultaneously, and even directly next to each other, and at this point, it is trivial to compare between them as there is no memory requirement. However, the story is somewhat different for small multiples where participants would have to scroll between representations that could be separated by many other drawn graphs; incurring a steep memory cost. Supporting previous work, we also find that interactive animation is nearly always the worst performer for all tasks and both factors [18, 21, 56, 102, 255].

Concerning continuous time comparison and the relatively low performance of interactive timeslicing, this is primarily due to interactive timeslicing not being designed for this type of comparison. During our evaluation with target users, we only saw a need for continuous time comparison on one occasion, and though our users were able to carry out this operation, they did not gain any insight from the results. However, it could be argued that we did not provide

the tools for the easy comparison of continuous timeslices and, as our users were not familiar with dynamic graph visualisation, the evaluation participants did not consider that this was an operation that they might typically like to carry out.

Interactive timeslicing reported a weak performance on *E2* across both studies. For the first study, using the large touch-screen display, we initially observed and theorised that this was due to the 'fat finger' [289] problem and the relative imprecision of touch screen displays when trying to hit a small or occluded target [191]. This theory was also informed by observations of the Plaid evaluation sessions, particularly the first session with the beta version of the interface, where our expert users did occasionally have an issue with the vagueness of the touch screen. This issue was most apparent in cases where they worked collaboratively and quickly tried an avenue suggested by one of their co-analysts.

During the first evaluation session, the operation to resize timeslices caused the most significant amount of error, and it was for this reason that we introduced the larger interaction handles for timeslicing resizing for our formal experiments. However, perhaps we went too far in the other direction and the handles used in the experiments were too large, leading to occlusion problems. We carried out pilot studies to verify the handle size, but a formal evaluation of the most usable and accurate timeslice resizing method would be a valuable contribution.

Alternatively, and based on observations from the first study, the issue with this operation may have been related to the lack of physical feedback with the operation. Participants found that their hands blocked their view of the side of the timeslice that they were resizing and so they could only visually confirm that the operation was successfully underway after the target timeslice was over a certain size, and this was frequently larger than their desired size.

Remaining with purely touch-based interaction will likely require some other non-visual indication to the user that the operation has successfully begun; haptic feedback would be ideal [235] though nigh on impossible to integrate with an existing mass-produced display. Audio feedback could be another alternative [148] though this has the great potential to become both annoying and distracting [226], the former leading to lower user satisfaction and the latter to lower accuracy and greater completion times.

In lieu of touch-based interaction, there may be value in transferring this operation to a mid-air interaction type [163, 164, 221, 248, 266, 321, 333, 347] therefore avoiding the problem of the hand occluding the target element. However, mid-air interactions are not necessarily more precise [268] and may not be the best choice for collaborative situations [163, 164, 266]. Ray-pointing techniques [165] are another available alternative interaction type, but these can lead

to indirect conflicts in collaborative situations where multiple users can interact simultaneously. Extrapolating the work of Jakobsen et al. [159], where they find that using a single mouse as input in collaborative situations leads to user conflict, we can theorise that giving a single point of input – whether mouse or ray-pointer – will nearly always lead to a sub-optimal collaborative experience. A further option would be to integrate handheld mobile devices and large displays, with Langner et al. [186] reporting positive results for this method, though we are unsure how this would perform for interactions where high precision is required.

Naturally, rather than trying to work around the required high-precision interaction to resize a timeslice, we could instead modify the operation trigger. One option would be to allow users to isolate a timeslice, similar to the functionality of 'layers' within Adobe Photoshop, before resizing. However, this is at least one extra interaction step that could prove costly in lengthy, sophisticated, analyses where this operation frequently occurs. Analysts from the Plaid evaluation suggested that each timeslice could have an associated pop-up box, or set of boundaries in text boxes, to allow more precise resize operations.

As a third option, we could instead extend the features of interactive timeslicing to make a 'small multiples lite'. Users could be offered the ability to select a very large continuous period of time, and then have some interaction available to split this selected time into continuous timeslices. Whether this would function better as equal time separation (e.g. each timeslice covers 8 hours) or as equal event separation [291] (e.g. each timeslice has 100 events, regardless of the total time) remains to be seen.

7.6 Comparing the Studies

After experimentally evaluating the generalisability of interactive timeslicing on a large touch-screen display, we felt it necessary to ensure that this method would generalise further to standard displays with mouse-pointer interaction. Vertically-mounted touch-screen displays are specialist hardware: they are expensive, take up a significant amount of both wall space (for mounting) and floor space (for users to access the screen to directly interact), and may not be accessible for all potential users. By validating our novel method for time navigation on 'normal' screens, we can make the method more available and, generally, more helpful to non-expert users.

Each study contained the same set of three tasks. The only interface differences between the two studies are that the interfaces from the first study had a 4K scale and resolution, whereas we down-scaled them for the second study to Full HD due to difficulties recruiting participants

			IT, IA		IT, SM		SM, IA	
Task	Measure	Experiment type	Near	Far	Near	Far	Near	Far
1	Correctness	In Person	Y	Y	Y	Y	N	N
		Remote	N	N	N	N	N	N
	Time	In Person	Y	Y	Y	Y	N	N
		Remote	Y	Y	Y	Y	N	N
2	Correctness	In Person	N	N	N	N	N	N
		Remote	N	N	N	Y	N	Y
	Time	In Person	Y	N	N	Y	Y	Y
		Remote	Y	Y	N	Y	Y	Y
3	Correctness	In Person	N	N	N	N	N	N
		Remote	N	N	N	N	N	N
	Time	In Person	Y	Y	Y	N	Y	Y
		Remote	Y	Y	Y	N	Y	Y

Key IT = Interactive timeslicing, IA = Interactive Animation, SM = Small Multiples, Y = a significant pairwise difference exists, N = no significant pairwise difference present

Table 7.1: The statistical pairwise differences between interfaces, conditions, and tasks for both experiments. Cells that are shaded in blue are cells where there is a difference in pairwise significance between the two experiments.

who had access to 4K displays. Participants were not shared across studies, i.e. participants who completed the first study were excluded from completing the second due to the potential learning effect.

There are very few differences between study results that hold for every task, interface, and time factor. The exception to this, however, is the completion time. Across the two studies, the completion time was longer for the study that we carried out remotely, with an average increase in completion time of 11 seconds for Task 1, 12 seconds for Task 2, and 7 seconds for Task 3. This could partly be explained by the 'Zoom lag' problem, though in this case it would be more expected for the increase in completion time to be equal across all tasks. An alternative interpretation could be that this is indeed caused by the 'Zoom lag' making interactions with the interfaces more challenging and that by the time participants have reached Task 3, they have mentally adapted to the non-immediate visual feedback of their interaction and are therefore better at predicting the result of their interaction; so the issue becomes less noticeable. A third interpretation could be that, as Task 3 requires relatively little interaction, the 'Zoom lag' was less costly.

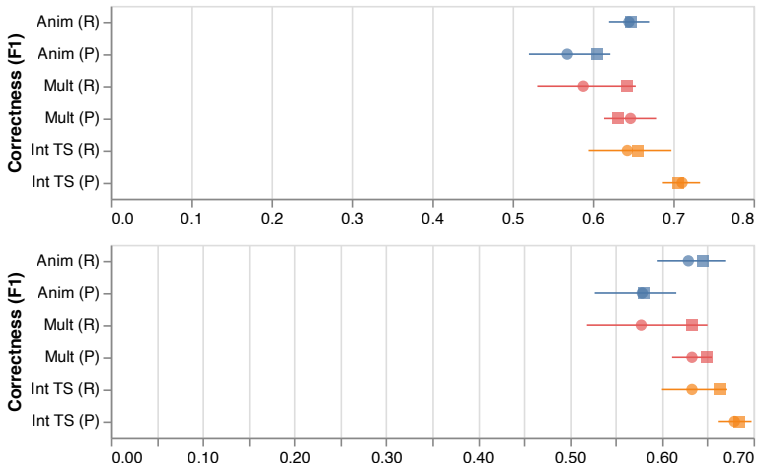


Figure 7.1: Task 1 correctness for the near (top) and far (bottom) conditions. The dot represents the mean, square the median, and 95% CIs shown. Interfaces with (R) are the remote experiment results, and with (P) are the in-person results.

Task 1 Comparing the two experiments for the first task, we can see that the greatest number of pairwise significance differences occurs within the correctness measure, with no difference in pairwise significance for the time measure, as shown in the first two rows of Table 7.1.

To help us understand this result, we examine the numeric correctness data, as shown in Figure 7.1, where there are slight variations between the in-person and remote studies. The remote experiment has a higher correctness score for interactive animation on both near and far than the in-person experiment. However, for small multiples and interactive timeslicing, the remote experiment has a lower average correctness measure than in-person. It is also evident that the mean correctness of all interfaces and conditions for the remote experiment is approximately the same, unlike the median interface correctness for the in-person experiment; this is why there is no pairwise significance found for any interfaces in the remote experiment.

We see that the median correctness for the interactive timeslicing interface is approximately 5% lower for the remote experiment. It is not the case that the other two interfaces have a highly improved correctness for the remote experiment, but instead that the correctness of the interactive timeslicing interface was reduced. The explanation here comes back again to the 'Zoom lag' – participants struggled to select individual edges in some cases as it would take a second or two to see the result of any interaction on their screen and, as a result, many participants became frustrated and would submit an answer that they considered merely 'good enough' rather than one that they believed was 100% correct.

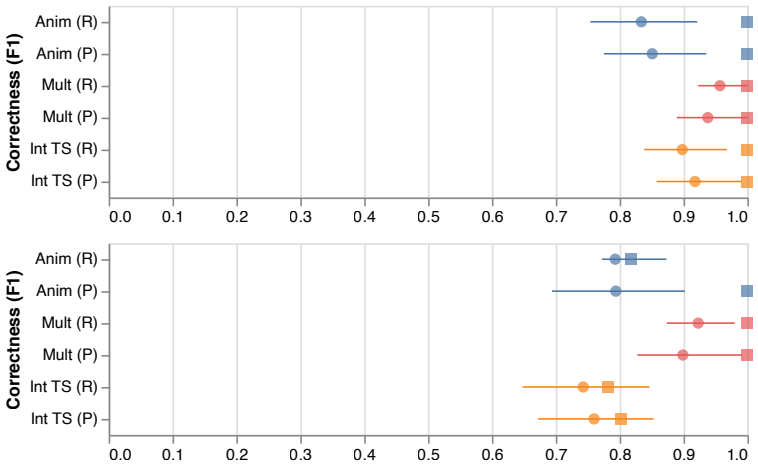


Figure 7.2: Task 2 correctness for the near (top) and far (bottom) conditions. The dot represents the mean, square the median, and 95% CIs shown. Interfaces with (R) are the remote experiment results, and with (P) are the in-person results.

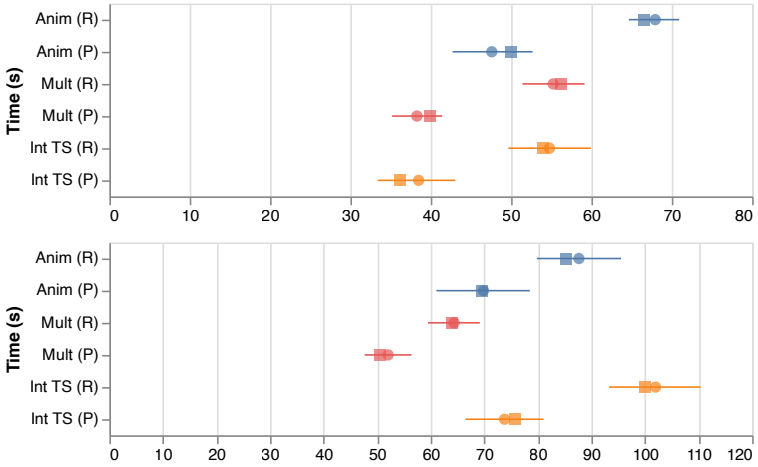


Figure 7.3: Task 2 completion time (s) in seconds for the near (top) and far (bottom) conditions without log transform. The dot represents the mean, square the median, and 95% CIs shown. Interfaces with (R) are the remote experiment results, and with (P) are the in-person results.

Task 2 For Task 2, there are three differences in pairwise significance, two for the correctness and one for completion time. For the far condition of the remote experiment, small multiples significantly outperforms interactive animation and interactive timeslicing in terms of correctness, whereas the in-person experiment finds no pairwise significance for the Task 2 correctness measure in any context. The numeric results data (see Figure 7.2) shows that, for the far condition, the correctness measure remains approximately the same for small multiples but reduces

for the other two interfaces within the remote experiment. Given that interactive timeslicing and interactive animation were the two interfaces requiring the most interaction to complete this task, this could express the higher interaction load, the observed zoom lag, or the change from direct touch input to indirect mouse input.

In terms of completion time, we also find a difference in pairwise significance with significance for the interactive timeslicing/interactive animation pair of the far condition for the remote experiment. This difference favours interactive animation, i.e. interactive animation is quicker than interactive timeslicing for the completion of this task in a remote context with direct mouse interaction. This result aligns with our expectations that moving to the more precise mouse interaction would eliminate the interaction issues that were present in the in-person experiment, for example, the 'fat finger' problem [289], and this would therefore improve completion times. A further experiment in an in-person setting but on standard displays with mouse interaction should verify if this is due to the 'Zoom lag' phenomena or some other underlying factor driving this difference.

Task 3 For Task 3, median correctness is 1 across both studies, all interfaces, and all factors. There is no meaningful difference between the remote and in-person study. For time, we see the same pattern as Tasks 1 and 2, with completion times lower for the in-person experiment and suspect that the reason for this is similar to the reasons already explored – primarily the impact of participants not being able to immediately see their interaction on screen due to lag.

Summary To summarise, we ran a remote experiment using the interfaces of interactive timeslicing, small multiples, and interactive timeslicing. Our participants used their standard desktop displays with mouse input to complete given tasks. Overall, the results of this study mirror the results of the in-person study detailed in Chapter 5 despite the change in display devices and interaction types. We find that the second study tasks take longer to complete than the first study and theorise that this is related to issues specific to the remote manner in which we carried out the experiment.

The minor differences in overall correctness between the two studies validate our hypothesis that interactive timeslicing is usable across different hardware and differing input methods. The difference in the time measure is more complex to understand. Limitations of remote visualisation research [46, 198] will undoubtedly have impacted our results, particularly the fact that participants could not see their input movements as they were happening due to the problem that we have taken to calling 'Zoom lag'.

On the whole, when comparing the results of the first and second studies, we can see that the change of interaction type and the display hardware does not seem to have had a significant effect on the efficiency or effectiveness of interactive timeslicing when compared to small multiples and interactive animation, except in cases where interactive timeslicing incurs a heavy interaction burden as in Task 2. Thus, we have a robust result that seems to carry independent of interaction method (mouse or touch) and display size (large or standard).

7.7 Our Place in Dynamic Graph Visualisation

There is a large body of work within dynamic graph visualisation literature comparing small multiples and animation. We have more evidence that small multiples are generally faster than animated approaches [18, 21, 102, 242, 255]. Based on our two studies we can now also provide evidence that, for the analysis of long-in-time dynamic graphs, interactive timeslicing outperforms both small multiples and interactive animation where analysis tasks involve the comparison of points that are distant in time. In terms of the primary research aim of the thesis, this is a contribution to interaction techniques. Though it is not a new method for interacting with hardware, it is a novel method for interacting with and exploring the temporal dimension present in long-in-time dynamic network data. We have also contributed an early version of an integrated, synchronised, view of event-based and dynamic graph visualisation and demonstrated the value of providing these sort of views in situations where timeslicing is essential due to scalability or other constraints. Through this we address the first part of our research aim – providing useful representations of long-in-time dynamic network data.

Per the survey of Beck et al. [37], existing approaches for time navigation can be abstracted, at a high level, to time-to-time and time-to-space approaches. Our novel interactive timeslicing approach sits firmly within the time-to-space area, however unlike other representations we do not timeslice the data for the user before display, and neither do we attempt to draw every possible timeslice simultaneously.

Through two formal studies and one evaluation of the DNP with public health researchers, we find that our novel time navigation technique is best suited to facilitate complex graph tasks involving the comparison of time points that are distant in time. Interactive timeslicing is also efficient for more straightforward graph tasks involving the comparison of single nodes across points that are far in time; however, it has no great benefit over small multiples here. We hypothesise that had we not prioritised the stability of the graph for our formal studies, and instead had relied on different highlighting techniques, there would be an even larger difference

in correctness and completion time between interactive timeslicing and small multiples (in favour of interactive timeslicing) for *E1* and *E3*.

We also find that interactive timeslicing generally performs poorly for tasks involving the comparison of many continuous timeslices. However, there are open questions around if this was caused by a deficiency in the implementation and whether we could take steps to negate the problems that we observed. However, during our expert evaluation, we did not frequently see that our analysts were necessarily interested in these types of tasks. It may be that in their field, social science, they are more interested in large differences over the whole graph, and there is the potential that they did not perform this operation as we did not sufficiently support it with the Plaid.

Our novel method for time navigation has the potential to be applied to any graph with a temporal component, for example, graphs from the cybersecurity field [234], graphs relating to multi-morbidity diagnoses, and those modelling infection spread within the field of epidemiology [36]. Indeed, it will potentially generalise beyond graphs. We chose to apply this method to networks, but it could also be utilised alongside scalar data and scatterplot-type representations. We believe that interactive timeslicing will likely be a useful time navigation method regardless of the underlying classification of the data.

We have also used the concept of maintaining interaction provenance along one dimension with time along a separate dimension. This might prove useful in collaborative settings in the future, particularly where users might not articulate clearly to their collaborators their analysis goals or thought processes. Including this feature type in future interfaces, not necessarily in the format that we created, may also galvanise wider discussion amongst collaborators who may not necessarily know or have previously worked with each other. It is possible that, had our user evaluation of the Plaid been carried out with several analysts who were unfamiliar to each other, we might have gained different insights and been able to make a more emphatic statement regarding the use of provenance information to generate discussion.

7.8 Limitations

In research, there are always trade-offs that introduce limitations by the decisions that we make. We can be constrained by hardware, software, or the availability of data, and frequently have to compromise in research design due to these constraints.

Dataset availability is a particular limitation of this thesis. We design, evaluate, and experiment with a limited number of datasets, and therefore we cannot make a concrete statement

about the generalisability of any of our features to other fields or to datasets of different compositions (for example, a long in time graph that is very sparse with 300 events over 60 days).

An extension of this issue is that we only use dynamic graph data to test our time navigation techniques, whereas testing a more comprehensive range of dynamic data types would enable us to test the applicability of interactive timeslicing to a broader range of users. For example, testing temporal scalar data would allow us to understand if the method generalises to this.

We also realise that long in time dynamic data has a specific temporal resolution and therefore a specific duration, and so we have no way of knowing how our interface or time interaction method would generalise to large time spans with a different temporal resolution.

We expect that data with a coarser temporal resolution would have less need for interactive timeslicing, whereas data with a finer temporal resolution would find more use for it.

For the Plaid evaluation (Chapter 4), we only have one application for the system with our social science researchers. Further work with a greater number of datasets, research domains, and users would ensure that the Plaid generalises out to other tasks and domains.

Regarding the limitations of the two formal studies, these are discussed at length in their respective chapters (Chapter 5.7 and Chapter 6.8). The primary limitation is that we chose a specific set of three tasks, whereas other task types may report different performance measures, and it is an important next step to test these. However, it is important to avoid confounds in formal studies, and, for this reason, it was necessary to restrict the scope and therefore the number of tasks that we tested.

One prominent feature of the Plaid that we do not validate in this thesis is our implementation of interaction provenance visualisation. While existing work supports the benefits of visualising this information [140, 244, 245], there is no research testing our method of encoding time in one dimension and interaction provenance in another dimension. The Plaid also does not offer comprehensive coverage of interaction provenance. For example, changes in the span or position of a timeslice are not recoverable or visible in the current version, and it could be challenging to provide comprehensive coverage without over-complicating the interface.

We also do not experimentally validate our approach of providing event-order timelines alongside dynamic network visualisations, nor the utility of facilitating the simultaneous display of multiple alternate representation types (i.e. matrices, node-link diagrams, scatterplots).

7.9 Future Work

Initially, further research should look to better quantify and understand the true scale of the impact of the 'Zoom lag' that we encountered during the second study. This is particularly important as much ongoing research during the pandemic has relied on remote communication and control tools, and we believe that this could cast a large shadow over results recorded in a remote moderated manner [198]. However, for unmoderated remote studies, such as those that take place on Amazon's Mechanical Turk platform, there is no such issue; though these suffer from their own unique set of limitations [46, 139]. As an immediate next step, and post-pandemic, we plan to repeat the second study for an in-person context on standardised hardware to validate that the difference in completion time is related to the experimental medium; rather than being an artefact of the visualisation interfaces when used on standard displays with indirect mouse input or of the remote nature of the experiment.

Further extending the generalisability of interactive timeslicing, this time navigation method must be tested on a broader range of graph tasks – both complex and straightforward – and tested using representations beyond the standard node-link diagrams. It will be necessary to consider that testing different representations will naturally involve testing different task types, especially for tasks where certain representations are already known to perform poorly, regardless of the time navigation technique.

Given we originally designed the Plaid, and interactive timeslicing, for large touch-screen displays which facilitate collaborative working this is another use-case that requires experimental validation. Our original intention for Chapter 6 was that this study would have verified the use of interactive timeslicing in collaborative situations. Unfortunately, the start of this planned study coincided with the first UK lockdown, and so we modified the experimental procedure for a remote context on standard displays [198]. Beyond further, varied, studies, we also believe that there will be value in repeating our current pair of studies with a much wider range of tasks and datasets; particularly with tasks and datasets that can validate the performance of interactive timeslicing for varying time granularities.

An issue that plagued interactive timeslicing, across both the direct touch-input and the indirect mouse-pointer input, was the 'fat finger' [289] problem where imprecise operations led to user frustration. One such event prompted a participant to remark that their ideal interface for the tasks that we gave them would be based upon small multiples but where the user could dismiss or collapse the time steps they did not want to see. In theory, and at a base level, this would effectively be the inverse of interactive timeslicing – with interactive timeslicing our

users select the time windows they want to see, whereas with the collapsible small multiples users would select a time that they did not want to see. However, this type of interface would return to a situation where data is algorithmically timesliced, and therefore the user has little-no control over timeslice size, with there being no potential to vary the size of the timeslices. In our observation of the target users of the Plaid, we noticed that they frequently compared timeslices of unequal size, and so removing this functionality is not necessarily the best choice.

During our observation of the evaluation of the Plaid, we also saw that the visualisation of interaction provenance was another feature that our target users found helpful. Some formal validation of the visualisation of interaction provenance alone and with our novel method of time navigation could provide data to better inform the design of future systems for the visual analysis of long in time dynamic data. If we are to formally evaluate interaction provenance, it will also be essential to test provenance-type tasks such as asking participants to repeat analysis on a distantly related part of the dataset and asking a user to decipher what has been done to the data before reaching a given ending state. To ensure the generalisability of these provenance experiments, it will also be important to test data types beyond dynamic network data, ensuring that it is useful for the widest possible audience.

Our visual encoding of interaction provenance can be best described as a provenance tree, and an open question is regarding the best way to deal with repeated interface states within this tree [68, 89, 93, 140]. Given the known benefits of data provenance [339], and with our target users commenting that they would typically pre-process their data, there would be some benefit to exploring ways in which we could offer both types of provenance; particularly in systems where the user carrying out the visual analysis would not necessarily be the person dealing directly with data processing. Annotating the process steps may help increase understanding for the visual analyst.

Regarding the underlying data, we have shown through the Plaid and our formal studies that interactive timeslicing is scalable in time and that the node-link diagrams generated from user-timesliced data are generally readable. However, despite having scalability in time, we do not have scalability in terms of the number of events in situations where events are dense. It would be useful to consider ways in which we could interactively apply non-uniform timeslicing methods.

Related to the event-density scalability, we must also consider how to best draw ‘good’ node-link layouts in ways that align with the users’ aims. Drawing graphs for dynamic data with many events is an open and particularly challenging problem, with some approaches

choosing to timeslice the network and compute the layout at each stage [50, 157, 263], others flattening the whole graph and calculating node positions as though they are static [115], and a small number opting to maintain the constant temporal component of the graph when calculating node positions [290]. Scalable methods for layouts in this context would provide an excellent direction for future work.

Chapter 8

Conclusion

Contents

8.1 Conclusion

In this thesis, we have explored useful representations and interaction techniques for the visualisation of long in time dynamic networks. We have created and tested a new visualisation system and formally evaluated the performance of our novel method of time navigation via two formal controlled experiments in different contexts, increasing generalisability of our results.

Chapter 3 introduces Dynamic Network Plaid (DNP), a novel system for the visualisation and analysis of long in time dynamic network data. Inspired by the reported advantages of large displays [31, 32, 86], we use an 84" touch screen vertically-mounted display with a maximum resolution of 4K (3840x2160). Effectively using the space provided by these devices gave us additional room to facilitate the simultaneous comparison of distant points in the time series and multiple linked graph representations within the DNP. As highlighted by previous work [232, 341], and as requested by our target users, we also provide integrated graph statistics within the visualisation system. Due to the importance of event order within a timeslice [282, 291], we also provide a per-timeslice event-based visualisation which, through a selection operation, is linked to the central timesliced graph representations.

Within the system, we also prioritise the visualisation of interaction provenance due to previous work reporting that this can aid the process memory of the user [140, 244, 245] and with several systems with support for interaction histories [68, 93, 304] report positive results related to insight discovery and exploration recall. Via the evaluation with public health researchers (Chapter 4), we find that this approach is particularly powerful for collaborative situations where users needed to communicate their previous analysis steps and choices. This integration also facilitates analysis cascades – users are able to automatically reproduce their analysis steps on a new part of the graph simply by defining a new timeslice, and the interaction provenance visualisation simply fills in the blanks for the newly selected time window.

In terms of time navigation, we also introduce a novel method for timeslice specification. Rather than setting an arbitrary timeslice size or limit, we instead hand control of timeslicing to the user and also allow them to only select regions of interest (rather than the whole graph). This reduces visual complexity and ensures that their attention is only consumed by regions relevant to the analysis. This choice also facilitates the comparison of points that are distant in time in a way not currently supported by other visualisation systems. With existing approaches, small multiples would traditionally insert many timeslices between the time points of interest, and for animated visualisations the user would need to watch the video of all intermediate time steps and remember its contents. Both of these make a direct comparison challenging, even for

points that are close in time, and where the time dimension is long this is particularly costly.

Chapter 4 describes the two evaluation sessions of the DNP that we carried out with public health researchers using one of their research datasets. We observed that the large touch-screen display facilitated collaboration between our users and that the visualisation of interaction provenance was powerful for generating discussion, particularly for engaging users who had not been closely observing the ongoing analysis. We also frequently observed that our users did not concern themselves with the comparison of subsequent timeslices, and instead, they would choose two very distant points in time to compare.

Based on this observation, we felt that it was essential to evaluate interactive timeslicing via a formal laboratory study. While many experiments have described the differences between time-to-time and time-to-space mappings, and the visualisation types within these broad categories, none have tested non-uniform user-defined timeslices, and so we have no understanding of the options for time control and treatment outside of continuous-time (e.g. animation) or uniform continuous timeslicing (e.g. small multiples). We first carry out an experiment using our large touch-screen vertical display to describe the differences between these three methods of time interaction. Later, to verify the generalisability of our method, we carried out a further experiment with mouse pointer input on standard displays in a remote setting.

Chapter 5 characterises the study that we carried out to evaluate the performance of interactive timeslicing against small multiples and interactive animation on our large touch-screen display. We were primarily interested in comparing interfaces for tasks that involve cross-time operations. Two of our selected interfaces, interactive animation and small multiples, have been well studied in previous literature [18,56,102,255], yet there are no experiments evaluating the performance of these against interactive timeslicing.

For tasks involving comparison of specific time points, interactive timeslicing offered greater speed than interactive animation or small multiples; when the comparison was of network structure (collections of edges) there was also an important difference in accuracy. In several instances, small multiples was also better than interactive animation, supporting previous work [18, 102]. For navigating time intervals, small multiples was faster than both interactive animation and interactive timeslicing when time intervals were larger (the far condition).

To further validate these results, we carried out an experiment in Chapter 6 to test the performance of interactive timeslicing against interactive animation and small multiples on standard displays with mouse input. We also reduced the screen resolution from 4K to Full HD. Structurally this was the same as the first study, but we carried it out remotely over Zoom

video conferencing software due to the COVID-19 pandemic. As in the first study, we find that interactive timeslicing outperforms other interfaces for tasks involving the comparison of points separate in time, but is less suited to tasks requiring the comparison of points that are sequential over an interval of time. Given that there was variation between our participants' physical screen size and screen construction, and potential differences in mouse sensitivity, we have a further amount of generalisability.

In this thesis we have introduced a new method of time treatment and navigation for the visualisation of long in time dynamic graphs. We have validated this via an expert user evaluation and two formal laboratory studies in different settings, showing that our results generalise beyond the DNP system. Through these studies, we show that interactive timeslicing outperforms small multiples and interactive animation for the comparison of points that are far apart in time, a function not previously supported by existing work. We have also demonstrated the value of the close integration of the visualisation of interaction provenance within a system, particularly for analysis cascades and situations where users expect to be working collaboratively. In the future, we plan to experimentally evaluate our integration of interaction provenance and verify the utility of interactive timeslicing on large touch-screen displays for collaborative situations.

Bibliography

[1] G. Abla, G. Wallace, D. Schissel, S. M. Flanagan, Q. Peng, and J. R. Burruss. Shared display wall based collaboration environment in the control room of the DIII-D national fusion facility. In *Proceedings of Workshop on Advanced Collaborative Environments*, 2005.

[2] J. Ahn, C. Plaisant, and B. Shneiderman. A task taxonomy for network evolution analysis. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):365–376, 2014.

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

[4] S. Al-Megren and R. A. Ruddle. Comparing tangible and multi-touch interaction for interactive data visualization tasks. In *Proceedings of the ACM International Conference on Tangible, Embedded, and Embodied Interaction*, page 279–286, 2016.

[5] A. B. Alencar, K. Börner, F. V. Paulovich, and M. C. F. de Oliveira. Time-aware visualization of document collections. In *Proceedings of the ACM Symposium on Applied Computing*, pages 997–1004, 2012.

[6] B. Alper, B. Bach, N. Henry Riche, T. Isenberg, and J.-D. Fekete. Weighted graph comparison techniques for brain connectivity analysis. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 483–492, 2013.

[7] R. Amar, J. Eagan, and J. Stasko. Low-level components of analytic activity in information visualization. In *IEEE Symposium on Information Visualization*, pages 111–117, 2005.

[8] R. Amar and J. Stasko. A knowledge task-based framework for design and evaluation of information visualizations. In *IEEE Symposium on Information Visualization*, pages 143–150, 2004.

[9] C. Andrews, A. Endert, B. Yost, and C. North. Information visualization on large, high-resolution displays: Issues, challenges, and opportunities. *Information Visualization*, 10(4):341–355, 2011.

[10] C. Andrews and C. North. The impact of physical navigation on spatial organization for sensemaking. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2207–2216, 2013.

[11] N. Andrienko and G. Andrienko. *Exploratory Analysis of Spatial and Temporal Data: A Systematic Approach*. Springer-Verlag, 2005.

[12] N. Andrienko and G. Andrienko. Designing visual analytics methods for massive collections of movement data. *Cartographica*, 42(2):117–138, 2007.

[13] N. Andrienko, G. Andrienko, and P. Gatalsky. Impact of data and task characteristics on design of spatio-temporal data visualization tools. In *Exploring Geovisualization*, pages 201 – 222. Elsevier, 2005.

[14] F. J. Anscombe. Graphs in statistical analysis. *The American Statistician*, 27(1):17–21, 1973.

[15] D. Archambault and H. C. Purchase. The mental map and memorability in dynamic graphs. In *IEEE Visualization Symposium*, pages 89–96, 2012.

[16] D. Archambault and H. C. Purchase. The "map" in the mental map: Experimental results in dynamic graph drawing. *International Journal of Human-Computer Studies*, 71(11):1044–1055, 2013.

[17] D. Archambault and H. C. Purchase. Mental map preservation helps user orientation in dynamic graphs. In *Proceedings of Graph Drawing*, pages 475–486, 2013.

[18] D. Archambault and H. C. Purchase. Can animation support the visualisation of dynamic graphs? *Information Sciences*, 330(C):495–509, 2016.

[19] D. Archambault and H. C. Purchase. On the effective visualisation of dynamic attribute cascades. *Information Visualization*, 15(1):51–63, 2016.

[20] D. Archambault, H. C. Purchase, and B. Pinaud. Difference map readability for dynamic graphs. In *International Symposium on Graph Drawing*, pages 50–61, 2010.

[21] D. Archambault, H. C. Purchase, and B. Pinaud. Animation, small multiples, and the effect of mental Map preservation in dynamic graphs. *IEEE Transactions on Visualization and Computer Graphics*, 17(4):539–552, 2011.

[22] C. Auerbach and L. B. Silverstein. *Qualitative data: An introduction to coding and analysis*, volume 21. NYU press, 2003.

[23] V. Auletta, D. Ferraioli, and V. Savarese. Manipulating an election in social networks through edge addition. In *International Conference of the Italian Association for Artificial Intelligence*, pages 495–510. Springer, 2019.

[24] B. Bach, P. Dragicevic, D. Archambault, C. Hurter, and S. Carpendale. A review of temporal data visualizations based on space-time cube operations. In *Proceedings of the Eurographics Conference on Visualization*, 2014.

[25] B. Bach, N. Henry-Riche, T. Dwyer, T. Madhyastha, J.-D. Fekete, and T. Grabowski. Small MultiPiles: Piling time to explore temporal patterns in dynamic networks. *Computer Graphics Forum*, 34(3):31–40, 2015.

[26] B. Bach, E. Pietriga, and J.-D. Fekete. GraphDiaries: animated transitions and temporal navigation for dynamic networks. *IEEE Transactions on Visualization and Computer Graphics*, 20(5):740–754, 2014.

[27] B. Bach, E. Pietriga, and J.-D. Fekete. Visualizing dynamic networks with matrix cubes. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 877–886, 2014.

[28] B. Bach, C. Shi, N. Heulot, T. Madhyastha, T. Grabowski, and P. Dragicevic. Time curves: Folding time to visualize patterns of temporal evolution in data. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):559–568, 2016.

[29] S. K. Badam, F. Amini, N. Elmqvist, and P. Irani. Supporting visual exploration for multiple users in large display environments. In *IEEE Conference on Visual Analytics Science and Technology*, pages 1–10, 2016.

[30] R. Ball and C. North. Analysis of user behavior on high-resolution tiled displays. In *Proceedings of the International Conference on Human-Computer Interaction*, pages 350–363, 2005.

[31] R. Ball and C. North. Effects of tiled high-resolution display on basic visualization and navigation tasks. In *Proceedings of the ACM SIGCHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 1196–1199, 2005.

[32] R. Ball, C. North, and D. A. Bowman. Move to improve: promoting physical navigation to increase user performance with large displays. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 191–200, 2007.

[33] R. C. Basole. Visualization of interfirm relations in a converging mobile ecosystem. *Journal of information Technology*, 24(2):144–159, 2009.

[34] M. Bastian, S. Heymann, and M. Jacomy. Gephi: an open source software for exploring and manipulating networks. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*, pages 361–362, 2009.

[35] L. Battle, R. Chang, and M. Stonebraker. Dynamic prefetching of data tiles for interactive visualization. In *Proceedings of the International Conference on Management of Data*, page 1363–1375, 2016.

[36] T. Baumgartl, M. Petzold, M. Wunderlich, M. Hohn, D. Archambault, M. Lieser, A. Dalpke, S. Scheithauer, M. Marschollek, V. Eichel, et al. In search of patient zero: visual analytics of pathogen transmission pathways in hospitals. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):711–721, 2020.

[37] F. Beck, M. Burch, S. Diehl, and D. Weiskopf. A taxonomy and survey of dynamic graph visualization. *Computer Graphics Forum*, 36(1):133–159, 2017.

[38] R. Becker, S. Eick, and A. Wilks. Visualizing network data. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):16–28, 1995.

[39] B. B. Bederson, J. Grosjean, and J. Meyer. Toolkit design for interactive structured graphics. *IEEE Trans. Softw. Eng.*, 30(8):535–546, Aug. 2004.

[40] L. Besançon and P. Dragicevic. The continued prevalence of dichotomous inferences at CHI. In *Proceedings of the ACM SIGCHI Conference Extended Abstracts on Human Factors in Computing Systems*, page 1–11, 2019.

[41] A. Bezerianos and R. Balakrishnan. Interaction and visualization techniques for very large scale high resolution displays. *University of Toronto Technical Report DGP-TR-2004-002*, 2004.

[42] A. Bezerianos and R. Balakrishnan. The vacuum: facilitating the manipulation of distant objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 361–370, 2005.

[43] A. Bezerianos and P. Isenberg. Perception of visual variables on tiled wall-sized displays for information visualization applications. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2516–2525, 2012.

[44] X. Bi and R. Balakrishnan. Comparing usage of a large high-resolution display to single or dual desktop displays for daily work. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 1005–1014, 2009.

[45] P. Boolchand, G. Lucovsky, J. Phillips, and M. Thorpe. Self-organization and the physics of glassy networks. *Philosophical Magazine*, 85(32):3823–3838, 2005.

[46] R. Borgo, L. Micallef, B. Bach, F. McGee, and B. Lee. Information visualization evaluation using crowdsourcing. *Computer Graphics Forum*, 37(3):573–595, 2018.

[47] C. Bors, T. Gschwandtner, and S. Miksch. Capturing and visualizing provenance from data wrangling. *IEEE Computer Graphics and Applications*, 39(6):61–75, 2019.

[48] M. Bostock, V. Ogievetsky, and J. Heer. D3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.

[49] I. Boyandin, E. Bertini, and D. Lalanne. A qualitative study on the exploration of temporal changes in flow maps with animation and small-multiples. *Computer Graphics Forum*, 31(3):1005–1014, 2012.

[50] U. Brandes and S. R. Corman. Visual unrolling of network evolution and the analysis of dynamic discourse. *Information Visualization*, 2(1):40–50, 2003.

[51] U. Brandes and M. Mader. A quantitative comparison of stress-minimization approaches for offline dynamic graph drawing. In *Graph Drawing*, pages 99–110, 2012.

[52] U. Brandes and B. Nick. Asymmetric relations in longitudinal social networks. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2283–2290, 2011.

[53] U. Brandes and D. Wagner. Analysis and visualization of social networks. In *Graph Drawing Software*, pages 321–340, 2004.

[54] C. Brauer, O. Ariza, and F. Steinicke. An active tangible device for multitouch-display interaction. In *Proceedings of ACM Mensch Und Computer*, page 439–444, 2019.

[55] R. Bredereck and E. Elkind. Manipulating opinion diffusion in social networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 894–900, 2017.

[56] M. Brehmer, B. Lee, P. Isenberg, and E. K. Choe. A comparative evaluation of animation and small multiples for trend visualization on mobile phones. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):364–374, 2020.

[57] M. Brehmer and T. Munzner. A multi-level typology of abstract visualization tasks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2376–2385, 2013.

[58] E. T. Brown, A. Ottley, H. Zhao, Q. Lin, R. Souvenir, A. Endert, and R. Chang. Finding waldo: Learning about users from their interactions. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1663–1672, 2014.

[59] R. C. Brown, T. Fischer, A. D. Goldwich, F. Keller, R. Young, and P. L. Plener. #cutting: Non-suicidal self-injury (NSSI) on Instagram. *Psychological Medicine*, pages 1–10, 2017.

[60] V. Bruder, H. B. Lahmar, M. Hlawatsch, S. Frey, M. Burch, D. Weiskopf, M. Herschel, and T. Ertl. Volume-based large dynamic graph analysis supported by evolution provenance. *Multimedia Tools and Applications*, 78(23):32939–32965, 2019.

[61] M. Burch. Isoline-enhanced dynamic graph visualization. In *Proceedings of the International Conference on Information Visualisation*, pages 1–8, 2016.

[62] M. Burch. The dynamic graph wall: Visualizing evolving graphs with multiple visual metaphors. *Journal of Visualisation*, 20(3):461–469, 2017.

[63] M. Burch, F. Beck, and D. Weiskopf. Radial edge splatting for visualizing dynamic directed graphs. In *Proceedings of the International Conference on Computer Graphics Theory and Applications*, pages 603–612, 2012.

[64] M. Burch and T. Reinhardt. Dynamic graph visualization on different temporal granularities. In *Proceedings of the International Conference on Information Visualisation*, pages 230–235, 2017.

[65] M. Burch, B. Schmidt, and D. Weiskopf. A matrix-based visualization for exploring dynamic compound digraphs. In *Proceedings of the International Conference on Information Visualisation*, pages 66–73, 2013.

[66] M. Burch, C. Vehlow, F. Beck, S. Diehl, and D. Weiskopf. Parallel edge splatting for scalable dynamic graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2344–2353, 2011.

[67] E. Cakmak, U. Schlegel, D. Jackle, D. Keim, and T. Schreck. Multiscale snapshots: Visual analysis of temporal summaries in dynamic graphs. *IEEE Transactions on Visualization and Computer Graphics*, page pp, 2020.

[68] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo. VisTrails: visualization meets data management. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 745–747, 2006.

[69] S. Carpendale. *Evaluating Information Visualizations*, page 19–45. Springer, 2008.

[70] D. Ceneda, T. Gschwandtner, T. May, S. Miksch, H. Schulz, M. Streit, and C. Tominski. Characterizing guidance in visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):111–120, 2017.

[71] D. Ceneda, T. Gschwandtner, and S. Miksch. A review of guidance approaches in visual data analysis: A multifocal perspective. *Computer Graphics Forum*, 38(3):861–879, 2019.

[72] H. Chen. Compound brushing [dynamic data visualization]. In *IEEE Symposium on Information Visualization*, pages 181–188, 2003.

[73] H. Chen, U. Soni, Y. Lu, V. Huroyan, R. Maciejewski, and S. Kobourov. Same stats, different graphs: Exploring the space of graphs in terms of graph properties. *IEEE Transactions on Visualization and Computer Graphics*, 27(3):2056–2072, 2021.

[74] Y. Chen, S. Barlowe, and J. Yang. Click2Annotate: Automated insight externalization with rich semantics. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology*, pages 155–162, 2010.

[75] Y. Chen, D. W. Clark, A. Finkelstein, T. C. Housel, and K. Li. Automatic alignment of high-resolution multi-projector displays using an uncalibrated camera. In *Proceedings of IEEE Visualization*, pages 125–130, 2000.

[76] Y. Chen, P. Xu, and L. Ren. Sequence synopsis: Optimize visual summary of temporal event data. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):45–55, 2018.

[77] Y. V. Chen, Z. C. Qian, R. Woodbury, J. Dill, and C. D. Shaw. Employing a parametric model for analytic provenance. *ACM Transactions on Interactive Intelligent Systems*, 4(1):1–32, 2014.

[78] F. Chevalier, P. Dragicevic, and S. Franconeri. The not-so-staggering effect of staggered animated transitions on visual tracking. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2241–2250, 2014.

[79] I. Cho, R. Wesslen, A. Karduni, S. Santhanam, S. Shaikh, and W. Dou. The anchoring effect in decision-making with visual analytics. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, pages 116–126, 2017.

[80] A. Cockburn, A. Karlson, and B. B. Bederson. A review of overview+detail, zooming, and focus+context interfaces. *ACM Computing Surveys*, 41(1), 2009.

[81] C. Collins, N. Andrienko, T. Schreck, J. Yang, J. Choo, U. Engelke, A. Jena, and T. Dwyer. Guidance in the human–machine analytics process. *Visual Informatics*, 2(3):166 – 180, 2018.

[82] C. Collins and S. Carpendale. VisLink: Revealing relationships amongst visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1192–1199, 2007.

[83] M. Correll and M. Gleicher. The semantics of sketch: Flexibility in visual query systems for time series data. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, pages 131–140, 2016.

[84] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti. Surround-screen projection-based virtual reality: the design and implementation of the cave. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques*, pages 135–142, 1993.

[85] M. Czajkowski. Lambda Vision. In *Geospatial InfoFusion and Video Analytics IV; and Motion Imagery for ISR and Situational Awareness II*, pages 232 – 244, 2014.

[86] M. Czerwinski, G. Smith, T. Regan, B. Meyers, G. G. Robertson, and G. Starkweather. Toward characterizing the productivity benefits of very large displays. In *Interact*, pages 9–16, 2003.

[87] N. Dakiche, F. Benbouzid-Si Tayeb, Y. Slimani, and K. Benatchba. Tracking community evolution in social networks: A survey. *Information Processing & Management*, 56(3):1084–1102, 2019.

[88] S. B. Davidson and J. Freire. Provenance and scientific workflows: challenges and opportunities. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1345–1350, 2008.

[89] K. Dextras-Romagnino and T. Munzner. Segmentifier: Interactive refinement of clickstream data. *Computer Graphics Forum*, 38(3):623–634, 2019.

[90] M. Dörk, S. Carpendale, and C. Williamson. The information flaneur: A fresh look at information seeking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, page 1215–1224, 2011.

[91] W. Dou, D. H. Jeong, F. Stukes, W. Ribarsky, H. R. Lipford, and R. Chang. Recovering reasoning processes from user interactions. *IEEE Computer Graphics and Applications*, 29(3):52–61, 2009.

[92] F. Du, B. Shneiderman, C. Plaisant, S. Malik, and A. Perer. Coping with volume and variety in temporal event sequences: Strategies for sharpening analytic focus. *IEEE Transactions on Visualization and Computer Graphics*, 23(6):1636–1649, 2017.

[93] C. Dunne, N. Henry Riche, B. Lee, R. Metoyer, and G. G. Robertson. GraphTrail: Analyzing large multivariate, heterogeneous networks while supporting exploration history. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 1663–1672, 2012.

[94] T. Dwyer and P. Eades. Visualising a fund manager flow graph with columns and worms. In *Information Visualization*, pages 147–152, 2002.

[95] G. Ellis and A. Dix. An explorative analysis of user evaluation studies in information visualisation. In *Proceedings of the AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*, page 1–7, 2006.

[96] T. Ellkvist, D. Koop, E. W. Anderson, J. Freire, and C. T. Silva. Using provenance to support real-time collaborative design of workflows. In *Proceedings of the ACM International Provenance and Annotation Workshop*, pages 266–279, 2008.

[97] N. Elmqvist and P. Tsigas. Causality visualization using animated growing polygons. In *Information Visualization*, pages 189–196, 2003.

[98] J. Epps, S. Lichman, and M. Wu. A study of hand shape use in tabletop gesture interaction. In *Proceedings of the ACM SIGCHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 748–753, 2006.

[99] C. Erten, P. J. Harding, S. G. Kobourov, K. Wampler, and G. Yee. Exploring the computing literature using temporal graph visualization. In *Proceedings of Visualization and Data Analysis*, pages 45–56, 2004.

[100] C. Fan and H. Hauser. Fast and accurate CNN-based brushing in scatterplots. *Computer Graphics Forum*, 37(3):111–120, 2018.

[101] M. Fandáková, K. Zábovská, B. Bučko, and M. Zábovskỳ. Improvements of computer assisted virtual environment (CAVE). In *Proceedings of the IEEE International Convention on Information, Communication and Electronic Technology*, pages 1680–1684, 2020.

[102] M. Farrugia and A. Quigley. Effective temporal graph layout: A comparative study of animation versus static display methods. *Information Visualization*, 10(1):47–64, 2011.

[103] P. Federico, W. Aigner, S. Miksch, F. Windhager, and L. Zenk. A visual analytics approach to dynamic social networks. In *Proceedings of the International Conference on Knowledge Management and Knowledge Technologies*, pages 1–8, 2011.

[104] P. Federico and S. Miksch. Evaluation of two interaction techniques for visualization of dynamic graphs. In *Proceedings of the International Symposium on Graph Drawing and Network Visualization*, pages 557–571, 2016.

[105] M. Feng, E. Peck, and L. Harrison. Patterns and pace: Quantifying diverse exploration behavior with visualizations on the web. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):501–511, 2019.

[106] N. Ferreira, J. Poco, H. T. Vo, J. Freire, and C. T. Silva. 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, 2013.

[107] D. Fisher, S. M. Drucker, M. Czerwinski, R. DeLine, and K. Rowan. Understanding the breadth of the event space: Learning from Logan. In *Proceedings of the IEEE VIS Workshop on Temporal & Sequential Event Analysis*, 2016.

[108] G. Fitzmaurice, A. Khan, G. Kurtenbach, and G. Binks. Cinematic meeting facilities using large displays. *IEEE Computer Graphics and Applications*, 25(4):17–21, 2005.

[109] C. Forlines, D. Wigdor, C. Shen, and R. Balakrishnan. Direct-touch vs. mouse input for tabletop displays. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 647–656, 2007.

[110] Y. Frishman and A. Tal. Dynamic drawing of clustered graphs. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 191–198, 2004.

[111] T. M. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991.

[112] R. Fuchs, J. Waser, and M. E. Groller. Visual human+machine learning. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1327–1334, 2009.

[113] G. W. Furnas and B. B. Bederson. Space-scale diagrams: Understanding multiscale interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, page 234–241, 1995.

[114] R. Gallotti and M. Barthelemy. The multilayer temporal network of public transport in Great Britain. *Scientific Data*, 2(1):1–8, 2015.

[115] E. R. Gansner, Y. Hu, and S. C. North. Interactive visualization of streaming text data with dynamic maps. *Journal of Graph Algorithms and Applications*, pages 515–540, 2013.

[116] H. Garg and M. Garg. Analysing the quality attributes of AOP using CYVIS tool. *International Journal of Computers & Technology*, pages 648–653, 2013.

[117] S. Garg, J. E. Nam, I. V. Ramakrishnan, and K. Mueller. Model-driven visual analytics. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology*, pages 19–26, 2008.

[118] M. Ghoniem, J.-D. Fekete, and P. Castagliola. A comparison of the readability of graphs using node-link and matrix-based representations. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 17–24, 2004.

[119] M. Ghoniem, J.-D. Fekete, and P. Castagliola. On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. *Information Visualization*, 4(2):114–135, 2005.

[120] P. A. Gloor and Y. Zhao. Tecflow-a temporal communication flow visualizer for social networks analysis. In *ACM CSCW Workshop on Social Networks*, volume 6, 2004.

[121] P. B. Goes, M. Lin, and C. Au Yeung. "Popularity effect" in user-generated content: Evidence from online product reviews. *Information Systems Research*, 25(2):222–238, 2014.

[122] T. E. Gorochowski, M. d. Bernardo, and C. S. Grierson. Using aging to visually uncover evolutionary processes on networks. *IEEE Transactions on Visualization and Computer Graphics*, 18(8):1343–1352, 2012.

[123] D. Gotz and Z. Wen. Behavior-driven visualization recommendation. In *Proceedings of the International Conference on Intelligent User Interfaces*, page 315–324, 2009.

[124] S. Gratzl, A. Lex, N. Gehlenborg, N. Cosgrove, and M. Streit. From visual exploration to storytelling and back again. *Computer Graphics Forum*, 35(3):491–500, 2016.

[125] D. Greene, D. Doyle, and P. Cunningham. Tracking the evolution of communities in dynamic social networks. In *2010 International Conference on Advances in Social Networks Analysis and Mining*, pages 176–183, 2010.

[126] S. Greenland. Invited commentary: The need for cognitive science in methodology. *American Journal of Epidemiology*, 186(6):639–645, 2017.

[127] S. Greenland, S. J. Senn, K. J. Rothman, J. B. Carlin, C. Poole, S. N. Goodman, and D. G. Altman. Statistical tests, p values, confidence intervals, and power: a guide to misinterpretations. *European Journal of Epidemiology*, 31:337–350, 2016.

[128] G. Groh, H. Hanstein, and W. Wörndl. Interactively visualizing dynamic social networks with DySo. In *Proceedings of the Intelligent User Interfaces Workshop on Visual Interfaces to the Social and the Semantic Web*, volume 2, 2009.

[129] F. Guimbretière, M. Stone, and T. Winograd. Fluid interaction with high-resolution wall-size displays. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 21–30, 2001.

[130] S. Guo, F. Du, S. Malik, E. Koh, S. Kim, Z. Liu, D. Kim, H. Zha, and N. Cao. Visualizing uncertainty and alternatives in event sequence predictions. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, page 1–12, 2019.

[131] V. Ha, K. M. Inkpen, R. L. Mandryk, and T. Whalen. Direct intentions: The effects of input devices on collaboration around a tabletop display. In *Proceedings of the IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, pages 177–184, 2006.

[132] S. Hachul and M. Jünger. An experimental comparison of fast algorithms for drawing general large graphs. In *Proceedings of Graph Drawing*, pages 235–250, 2006.

[133] S. Hadlak, H.-J. Schulz, and H. Schumann. In situ exploration of large dynamic networks. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2334–2343, 2011.

[134] S. Hameed, T. Ferris, S. Jayaraman, and N. Sarter. Using informative peripheral visual and tactile cues to support task and interruption management. *SAGE Human Factors*, 51(2):126–135, 2009.

[135] M. S. Hancock, F. D. Vernier, D. Wigdor, S. Carpendale, and C. Shen. Rotation and translation mechanisms for tabletop interaction. In *Proceedings of the IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, pages 79–88, 2006.

[136] N. Hashimoto, S. Jeong, Y. Takeyama, and M. Sato. Immersive multi-projector display on hybrid screens with human-scale haptic and locomotion interfaces. In *Proceedings of the IEEE International Conference on Cyberworlds*, pages 361–368, 2004.

[137] H. Hauser, F. Ledermann, and H. Doleisch. Angular brushing of extended parallel coordinates. In *IEEE Symposium on Information Visualization*, pages 127–130, 2002.

[138] A. Hayashi, T. Matsubayashi, T. Hoshide, and T. Uchiyama. Initial positioning method for online and real-time dynamic graph drawing of time varying data. In *Proceedings of the IEEE International Conference on Information Visualisation*, pages 435–444, 2013.

[139] J. Heer and M. Bostock. Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 203–212, 2010.

[140] J. Heer, J. Mackinlay, C. Stolte, and M. Agrawala. Graphical histories for visualization: Supporting analysis, communication, and evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1189–1196, 2008.

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

[142] S. Hellberg, D. Eklund, D. R. Gawel, M. Köpsén, H. Zhang, C. E. Nestor, I. Kockum, T. Olsson, T. Skogh, A. Kastbom, et al. Dynamic response genes in cd4+ t cells reveal a network of interactive proteins that classifies disease activity in multiple sclerosis. *Cell reports*, 16(11):2928–2939, 2016.

[143] N. Henry and J.-D. Fekete. MatrixExplorer: a dual-representation system to explore social networks. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):677–684, 2006.

[144] N. Henry and J.-D. Fekete. Matlink: Enhanced matrix visualization for analyzing social networks. In *Proceedings of the IFIP Conference on Human-Computer Interaction*, pages 288–302, 2007.

[145] M. Hereld, I. R. Judson, and R. Stevens. Dottytoto: A measurement engine for aligning multi-projector display systems. In *Proceedings of Projection Displays*, pages 73–86, 2002.

[146] M. Hereld, I. R. Judson, and R. L. Stevens. Introduction to building projection-based tiled display systems. *IEEE Computer Graphics and Applications*, 20(4):22–28, 2000.

[147] I. Herman, G. Melancon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.

[148] E. Hoggan, A. Crossan, S. A. Brewster, and T. Kaaresoja. Audio or tactile feedback: which modality when? In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 2253–2256, 2009.

[149] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

[150] P. Holme and J. Saramäki. Temporal networks. *Physics Reports*, 519(3):97–125, 2012.

[151] E. Hoque, V. Setlur, M. Tory, and I. Dykeman. Applying pragmatics principles for interaction with visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):309–318, 2018.

[152] D. Hutchings, M. Czerwinski, B. Meyers, and J. Stasko. Exploring the use and affordances of multiple display environments. In *Workshop on Ubiquitous Display Environments at UbiComp*, pages 1–6, 2004.

[153] P. Isenberg, S. Carpendale, A. Bezerianos, N. Henry, and J.-D. Fekete. Coconuttrix: Collaborative retrofitting for information visualization. *IEEE Computer Graphics and Applications*, 29(5):44–57, 2009.

[154] P. Isenberg, P. Dragicevic, W. Willett, A. Bezerianos, and J.-D. Fekete. Hybrid-image visualization for large viewing environments. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2346–2355, 2013.

[155] P. Isenberg, T. Isenberg, T. Hesselmann, B. Lee, U. Von Zadow, and A. Tang. Data visualization on interactive surfaces: A research agenda. *IEEE Computer Graphics and Applications*, 33(2):16–24, 2013.

[156] M. Itoh and M. Toyoda. EventStacks: Integration of event visualizations for physical and social sensor data. In *Proceedings of the IEEE VIS Workshop on Temporal & Sequential Event Analysis*, 2016.

[157] M. Itoh, M. Toyoda, and M. Kitsuregawa. An interactive visualization framework for time-series of web graphs in a 3D environment. In *Proceedings of the IEEE International Conference Information Visualisation*, pages 54–60, 2010.

[158] M. R. Jakobsen and K. HornbÆk. Up close and personal: Collaborative work on a high-resolution multitouch wall display. *ACM Transactions on Computer-Human Interaction*, 21(2), 2014.

[159] M. R. Jakobsen and K. Hornbæk. Negotiating for space? Collaborative work using a wall display with mouse and touch input. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, page 2050–2061, 2016.

[160] M. R. Jakobsen and K. Hornbæk. Is moving improving? some effects of locomotion in wall-display interaction. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 4169–4178, 2015.

[161] Y. Jansen, J. Schjerlund, and K. Hornbæk. Effects of locomotion and visual overview on spatial memory when interacting with wall displays. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 1–12, 2019.

[162] W. Javed and N. Elmqvist. Exploring the design space of composite visualization. In *IEEE Pacific Visualization Symposium*, pages 1–8, 2012.

[163] H. Jiang, C. Gao, T. Mao, H. Li, and Z. Wang. Efficiency group interaction between participants and large display. In *In Proceedings of the International Conference on Virtual Reality and Visualization*, pages 274–275, 2017.

[164] H. Jiang, C. Gao, T. Mao, H. Li, and Z. Wang. Exploring the effects of group interaction in large display systems. *International Journal of Performability Engineering*, 14(1):159–167, 2018.

[165] R. Jota, M. A. Nacenta, J. A. Jorge, S. Carpendale, and S. Greenberg. A comparison of ray pointing techniques for very large displays. In *Proceedings of Graphics Interface*, page 269–276, 2010.

[166] N. Kadivar, V. Chen, D. Dunsmuir, E. Lee, C. Qian, J. Dill, C. Shaw, and R. Woodbury. Capturing and supporting the analysis process. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology*, pages 131–138, 2009.

[167] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, page 3363–3372, 2011.

[168] Y.-a. Kang and J. Stasko. Characterizing the intelligence analysis process through a longitudinal field study: Implications for visual analytics. *Information Visualization*, 13(2):134–158, 2014.

[169] N. Kerracher. *Tasks and visual techniques for the exploration of temporal graph data*. PhD thesis, Edinburgh Napier University, 2017.

[170] N. Kerracher, J. Kennedy, and K. Chalmers. The design space of temporal graph visualisation. In *EuroVis - Short Papers*, pages 7–11, 2014.

[171] N. Kerracher, J. Kennedy, and K. Chalmers. Tasks for temporal graph visualisation, 2014.

[172] N. Kerracher, J. Kennedy, and K. Chalmers. A task taxonomy for temporal graph visualisation. *IEEE Transactions on Visualization and Computer Graphics*, 21(10):1160–1172, 2015.

[173] A. Khan, G. Fitzmaurice, D. Almeida, N. Burtnyk, and G. Kurtenbach. A remote control interface for large displays. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 127–136, 2004.

[174] M. A. Khan and A. Nandi. Flux capacitors for JavaScript Deloreans: Approximate caching for physics-based data interaction. In *Proceedings of the International Conference on Intelligent User Interfaces*, page 177–185, 2019.

[175] J. Kim, J. Park, H. Kim, and C. Lee. HCI (human computer interaction) using multi-touch tabletop display. In *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pages 391–394, 2007.

[176] U. Kister, K. Klamka, C. Tominski, and R. Dachselt. Grasp: Combining spatially-aware mobile devices and a display wall for graph visualization and interaction. *Computer Graphics Forum*, 36(3):503–514, 2017.

[177] S. Knudsen, M. R. Jakobsen, and K. Hornbæk. An exploratory study of how abundant display space may support data analysis. In *Proceedings of the ACM Nordic Conference on Human-Computer Interaction*, page 558–567, 2012.

[178] S. Koch, H. Bosch, M. Giereth, and T. Ertl. Iterative integration of visual insights during patent search and analysis. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology*, pages 203–210, 2009.

[179] R. Kosara, C. G. Healey, V. Interrante, D. H. Laidlaw, and C. Ware. User studies: Why, how, and when? *IEEE Computer Graphics and Applications*, 23(4):20–25, 2003.

[180] J. Krause, A. Perer, and H. Stavropoulos. Supporting iterative cohort construction with visual temporal queries. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):91–100, 2016.

[181] C. Krumbholz, J. Leigh, A. Johnson, L. Renambot, and R. Kooima. Lambda table: high resolution tiled display table for interacting with large visualizations. In *Proceedings of Workshop on Advanced Collaborative Environments*, 2005.

[182] G. Kumar and M. Garland. Visual exploration of complex time-varying graphs. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):805–812, 2006.

[183] T. Lammarsch, A. Rind, W. Aigner, and S. Miksch. Developing an extended task framework for exploratory data analysis along the structure of time. In *International Workshop on Visual Analytics*. The Eurographics Association, 2012.

[184] A. Lancichinetti and S. Fortunato. Community detection algorithms: A comparative analysis. *Physical Review E*, 80(5):056117, 2009.

[185] R. Langner and R. Dachselt. Towards visual data exploration at wall-sized displays by combining physical navigation with spatially-aware devices. In *Proceedings of the IEEE VIS Poster Program*, 2018.

[186] R. Langner, U. Von Zadow, T. Horak, A. Mitschick, and R. Dachselt. Content sharing between spatially-aware mobile phones and large vertical displays supporting collaborative work. *Collaboration Meets Interactive Spaces*, pages 75–96, 2016.

[187] A. Lee and D. Archambault. Communities found by users – not algorithms: Comparing human and algorithmically generated communities. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 2396–2400, 2016.

[188] A. Lee, D. Archambault, and M. Nacenta. Dynamic Network Plaid: A tool for the analysis of dynamic networks. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 1–14, 2019.

[189] A. Lee, D. Archambault, and M. A. Nacenta. The effectiveness of interactive visualization techniques for time navigation of dynamic graphs on large displays. *IEEE Transactions on Visualization & Computer Graphics*, 27(02):528–538, 2021.

[190] B. Lee, C. Plaisant, C. S. Parr, J.-D. Fekete, and N. Henry. Task taxonomy for graph visualization. In *Proceedings of the AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*, page 1–5. Association for Computing Machinery, 2006.

[191] H. Lee, I. Ko, Y. R. Tan, D. Zhang, and C. Li. Usability impact of occlusion-free techniques on commonly-used multitouch actions. In *Proceedings of the International ACM In-Cooperation HCI and UX Conference*, pages 96–105, 2019.

[192] S. Lewis. Qualitative inquiry and research design: Choosing among five approaches. *Health Promotion Practice*, 16(4):473–475, 2015.

[193] H. Liao, M. Iwahara, N. Hata, I. Sakuma, T. Dohi, T. Koike, Y. Momoi, T. Minakawa, M. Yamasaki, F. Tajima, et al. High-resolution integral videography autostereoscopic display using multi-projector. In *Proceedings of the International Display Workshop*, 2002.

[194] C. D. G. Linhares, B. A. N. Travençolo, J. G. S. Paiva, and L. E. C. Rocha. DyNetVis: A system for visualization of dynamic networks. In *Proceedings of the ACM Symposium on Applied Computing*, page 187–194, 2017.

[195] C. Liu, O. Chapuis, M. Beaudouin-Lafon, and E. Lecolinet. Shared interaction on a wall-sized display in a data manipulation task. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, page 2075–2086, 2016.

[196] C. Liu, O. Chapuis, M. Beaudouin-Lafon, and E. Lecolinet. CoReach: Cooperative gestures for data manipulation on wall-sized displays. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, page 6730–6741, 2017.

[197] C. Liu, K. Zhang, H. Xiong, G. Jiang, and Q. Yang. Temporal skeletonization on sequential data: Patterns, categorization, and visualization. *IEEE Transactions on Knowledge and Data Engineering*, 28(1):211–223, 2016.

[198] T. Losev, S. Storteboom, S. Carpendale, and S. Knudsen. Distributed synchronous visualization design: Challenges and strategies. In *IEEE Workshop on Evaluation and Beyond - Methodological Approaches to Visualization*, pages 1–10, 2020.

[199] J. D. Mackinlay and J. Heer. Wideband displays: mitigating multiple monitor seams. In *Proceedings of the ACM SIGCHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 1521–1524, 2004.

[200] M. Madkour, J. Du, H.-Y. Song, and C. Tao. Temporal clinical event clustering and visualization. In *Proceedings of the IEEE VIS Workshop on Temporal & Sequential Event Analysis*, 2016.

[201] A. Majumder and R. Stevens. LAM: Luminance attenuation map for photometric uniformity in projection based displays. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, pages 147–154, 2002.

[202] S. Malik, F. Du, M. Monroe, E. Onukwugha, C. Plaisant, and B. Shneiderman. Cohort comparison of event sequences with balanced integration of visual analytics and statistics. In *Proceedings of the ACM International Conference on Intelligent User Interfaces*, page 38–49, 2015.

[203] M. R. Marner, R. T. Smith, B. H. Thomas, K. Klein, P. Eades, and S.-H. Hong. GION: interactively untangling large graphs on wall-sized displays. In *Proceedings of Graph Drawing*, pages 113–124, 2014.

[204] N. Masuda and P. Holme. Predicting and controlling infectious disease epidemics using temporal networks. *F1000Prime Reports*, 5, 2013.

[205] A. Mathisen, T. Horak, C. N. Klokmose, K. Grønbæk, and N. Elmqvist. InsideInsights: Integrating data-driven reporting in collaborative visual analytics. *Computer Graphics Forum*, 38(3):649–661, 2019.

[206] M. L. Mauriello, B. Shneiderman, F. Du, S. Malik, and C. Plaisant. Simplifying overviews of temporal event sequences. In *Proceedings of the ACM SIGCHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 2217–2224, 2016.

[207] J. E. McGrath. *Methodology Matters: Doing Research in the Behavioral and Social Sciences*, page 152–169. Elsevier, 1995.

[208] A. C. McLaughlin, W. A. Rogers, and A. D. Fisk. Using direct and indirect input devices: Attention demands and age-related differences. *ACM Transactions on Computer-Human Interaction*, 16(1):1–15, 2009.

[209] M. Monroe, R. Lan, H. Lee, C. Plaisant, and B. Shneiderman. Temporal event sequence simplification. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2227–2236, 2013.

[210] M. Monroe, R. Lan, J. Morales del Olmo, B. Shneiderman, C. Plaisant, and J. Millstein. The challenges of specifying intervals and absences in temporal queries: A graphical language approach. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, page 2349–2358, 2013.

[211] M. R. Morris, J. Lombardo, and D. Wigdor. WeSearch: supporting collaborative search and sensemaking on a tabletop display. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pages 401–410, 2010.

[212] M. R. Morris, J. O. Wobbrock, and A. D. Wilson. Understanding users' preferences for surface gestures. In *Proceedings of Graphics Interface*, page 261–268, 2010.

[213] W. Muller and H. Schumann. Visualization methods for time-dependent data–an overview. In *Proceedings of the IEEE Winter Simulation Conference*, pages 737–745, 2003.

[214] T. Munzner. A nested model for visualization design and validation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):921–928, 2009.

[215] T. Munzner. *Visualization analysis and design*. CRC Press, 2014.

[216] P. K. Muthumanickam, K. Vrotsou, M. Cooper, and J. Johansson. Shape grammar extraction for efficient query-by-sketch pattern matching in long time series. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, pages 121–130, 2016.

[217] B. Mutlu, E. Veas, and C. Trattner. Vizrec: Recommending personalized visualizations. *ACM Transactions on Interactive Intelligent Systems*, 6(4):1–39, 2016.

[218] M. Nacenta, U. Hinrichs, and S. Carpendale. FatFonts: combining the symbolic and visual aspects of numbers. In *Proceedings of the ACM International Working Conference on Advanced Visual Interfaces*, pages 407–414, 2012.

[219] M. A. Nacenta, D. Pinelle, D. Stuckel, and C. Gutwin. The effects of interaction technique on coordination in tabletop groupware. In *Proceedings of ACM Graphics Interface*, page 191–198, 2007.

[220] M. A. Nacenta, S. Sakurai, T. Yamaguchi, Y. Miki, Y. Itoh, Y. Kitamura, S. Subramanian, and C. Gutwin. E-conic: A perspective-aware interface for multi-display environments. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, page 279–288, 2007.

[221] M. Nancel, J. Wagner, E. Pietriga, O. Chapuis, and W. Mackay. Mid-air pan-and-zoom on wall-sized displays. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, page 177–186, 2011.

[222] P. H. Nguyen, K. Xu, A. Bardill, B. Salman, K. Herd, and B. L. W. Wong. SenseMap: Supporting browser-based online sensemaking through analytic provenance. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, pages 91–100, 2016.

[223] T. Ni, G. Schmidt, O. Staadt, M. Livingston, R. Ball, and R. May. A survey of large high-resolution display technologies, techniques, and applications. In *Proceedings of the IEEE Virtual Reality Conference*, pages 223–236, 2006.

[224] S. Ntoa, C. Birliraki, G. Drossis, G. Margetis, I. Adami, and C. Stephanidis. UX design of a big data visualization application supporting gesture-based interaction with a large display. In *Proceedings of the International Conference on Human Interface and the Management of Information*, pages 248–265, 2017.

[225] J. R. Nuñez, C. R. Anderton, and R. S. Renslow. Optimizing colormaps with consideration for color vision deficiency to enable accurate interpretation of scientific data. *PLOS ONE*, 13(7):1–14, 2018.

[226] R. O'Dea, R. Jedir, and F. Neff. Auditory distraction in HCI: Towards a framework for the design of hierarchically-graded auditory notifications. In *Proceedings of the International Audio Mostly Conference*, pages 61–66, 2019.

[227] M. Okoe, R. Jianu, and S. Kobourov. Node-link or adjacency matrices: Old question, new insights. *IEEE Transactions on Visualization and Computer Graphics*, 25(10):2940–2952, 2019.

[228] J. Paay, D. Raptis, J. Kjeldskov, M. B. Skov, E. V. Ruder, and B. M. Lauridsen. Investigating cross-device interaction between a handheld device and a large display. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, page 6608–6619, 2017.

[229] G. Palla, A.-L. Barabási, and T. Vicsek. Quantifying social group evolution. *Nature*, 446(7136):664–667, 2007.

[230] H. Pashler. Attention and visual perception: Analyzing divided attention. *Visual cognition*, 2(1):71–100, 1995.

[231] M. Q. Patton. *Qualitative research & evaluation methods: Integrating theory and practice*. Sage publications, 2014.

[232] A. Perer and B. Shneiderman. Integrating statistics and visualization for exploratory power: From long-term case studies to design guidelines. *IEEE Computer Graphics and Applications*, 29(3):39–51, 2009.

[233] D. J. Peuquet. 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, 1994.

[234] A. Piplai, S. Mittal, A. Joshi, T. Finin, J. Holt, and R. Zak. Creating cybersecurity knowledge graphs from malware after action reports. *IEEE Access*, 8(1):211691–211703, 2020.

[235] M. J. Pitts, G. Burnett, L. Skrypchuk, T. Wellings, A. Attridge, and M. A. Williams. Visual–haptic feedback interaction in automotive touchscreens. *Displays*, 33(1):7–16, 2012.

[236] C. Plaisant and B. Shneiderman. The diversity of data and tasks in event analytics. In *Proceedings of the IEEE VIS Workshop on Temporal & Sequential Event Analysis*, 2016.

[237] M. D. Plumlee and C. Ware. Zooming versus multiple window interfaces: Cognitive costs of visual comparisons. *ACM Transactions on Computer-Human Interaction*, 13(2):179–209, 2006.

[238] M. Pohl and P. Birke. Interactive exploration of large dynamic networks. In *Proceedings of the International Conference on Advances in Visual Information Systems*, pages 56–67, 2008.

[239] A. Prouzeau, A. Bezerianos, and O. Chapuis. Evaluating multi-user selection for exploring graph topology on wall-displays. *IEEE Transactions on Visualization and Computer Graphics*, 23(8):1936–1951, 2017.

[240] S. Pupyrev and A. Tikhonov. Analyzing conversations with dynamic graph visualization. In *Proceedings of the IEEE International Conference on Intelligent Systems Design and Applications*, pages 748–753, 2010.

[241] H. C. Purchase, E. Hoggan, and C. Görg. How important is the "mental map"?–an empirical investigation of a dynamic graph layout algorithm. In *Proceedings of the International Symposium on Graph Drawing*, pages 184–195, 2006.

[242] H. C. Purchase and A. Samra. Extremes are better: Investigating mental map preservation in dynamic graphs. In *Proceedings of the International Conference on Theory and Application of Diagrams*, pages 60–73, 2008.

[243] Qingsong Liu, Y. Hu, L. Shi, Xinzhu Mu, Y. Zhang, and J. Tang. EgoNetCloud: Event-based egocentric dynamic network visualization. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, pages 65–72, 2015.

[244] E. D. Ragan, A. Endert, J. Sanyal, and J. Chen. Characterizing provenance in visualization and data analysis: an organizational framework of provenance types and purposes. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):31–40, 2016.

[245] E. D. Ragan, J. R. Goodall, and A. Tung. Evaluating how level of detail of visual history affects process memory. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 2711–2720, 2015.

[246] D. Raja, D. Bowman, J. Lucas, and C. North. Exploring the benefits of immersion in abstract information visualization. In *Proceedings of the Immersive Projection Technology Workshop*, pages 61–69, 2004.

[247] R. Raskar, J. van Baar, P. Beardsley, T. Willwacher, S. Rao, and C. Forlines. ILamps: Geometrically aware and self-configuring projectors. In *Proceedings of the ACM Transactions on Graphics*, page 809–818, 2003.

[248] H. Rateau, Y. Rekik, L. Grisoni, and J. Jorge. Talaria: Continuous drag & drop on a wall display. In *Proceedings of the ACM International Conference on Interactive Surfaces and Spaces*, page 199–204, 2016.

[249] F. Reitz. A framework for an ego-centered and time-aware visualization of relations in arbitrary data repositories. *arXiv preprint arXiv:1009.5183*, 2010.

[250] D. Ren, L. R. Marusich, J. O'Donovan, J. Z. Bakdash, J. A. Schaffer, D. N. Cassenti, S. E. Kase, H. E. Roy, W.-y. S. Lin, T. Höllerer, and et al. Understanding node-link and matrix visualizations of networks: A large-scale online experiment. *Network Science*, 7(2):242–264, 2019.

[251] P. Riehmann, G. Molina León, J. Reibert, F. Echtler, and B. Froehlich. Short-contact touch-manipulation of scatterplot matrices on wall displays. *Computer Graphics Forum*, 39(3):265–276, 2020.

[252] A. Rind, W. Aigner, M. Wagner, S. Miksch, and T. Lammarsch. User tasks for evaluation: Untangling the terminology throughout visualization design and development. In *Proceedings of the Workshop on Beyond Time and Errors: Novel Evaluation Methods for Visualization*, page 9–15. Association for Computing Machinery, 2014.

[253] J. C. Roberts. Exploratory visualization with multiple linked views. In *Exploring Geovisualization*, pages 159–180. Elsevier, 2005.

[254] J. C. Roberts and M. A. E. Wright. Towards ubiquitous brushing for information visualization. In *Proceedings of the Conference on Information Visualization*, page 151–156, 2006.

[255] G. Robertson, R. Fernandez, D. Fisher, B. Lee, and J. Stasko. Effectiveness of animation in trend visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1325–1332, 2008.

[256] M. Robinson, J. Laurence, A. Hogue, J. Zacher, A. German, and M. Jenkin. Ivy: Basic design and construction details. In *Proceedings of the International Conference on Augmented Reality and Teleexistence*, pages 1–5, 2002.

[257] Y. Rogers, W. Hazlewood, E. Blevis, and Y.-K. Lim. Finger talk: Collaborative decision-making using talk and fingertip interaction around a tabletop display. In *Proceedings of the ACM SIGCHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 1271–1274, 2004.

[258] C. Rooney and R. A. Ruddle. Hired: A high-resolution multi-window visualisation environment for cluster-driven displays. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, page 2–11, 2015.

[259] M. Rosvall, D. Axelsson, and C. T. Bergstrom. The map equation. *The European Physical Journal-Special Topics*, 178(1):13–23, 2009.

[260] O. Rübel and B. P. Bowen. Bastet: Shareable and reproducible analysis and visualization of mass spectrometry imaging data via OpenMSI. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):1025–1035, 2018.

[261] R. Ruddle, J. Bernard, T. May, H. Lücke-Tieke, and J. Kohlhammer. Methods and a research agenda for the evaluation of event sequence visualization techniques. In *Proceedings of the IEEE VIS Workshop on Temporal & Sequential Event Analysis*, 2016.

[262] R. Ruddle, R. Thomas, R. Randell, P. Quirke, and D. Treanor. Performance and interaction behaviour during visual search on large, high-resolution displays. *SAGE Information Visualization*, 14(2):137 – 147, 2015.

[263] S. Rufiange and M. J. McGuffin. DiffAni: Visualizing dynamic graphs with a hybrid of difference maps and animation. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2556–2565, 2013.

[264] S. Rufiange and G. Melançon. AniMatrix: A matrix-based visualization of software evolution. In *Proceedings of the IEEE Working Conference on Software Visualization*, pages 137–146, 2014.

[265] K. Ryall, C. Forlines, C. Shen, and M. R. Morris. Exploring the effects of group size and table size on interactions with tabletop shared-display groupware. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pages 284–293, 2004.

[266] Y. Sakakibara, Y. Matsuda, T. Komuro, and K. Ogawa. Simultaneous interaction with a large display by many users. In *Proceedings of the ACM International Symposium on Pervasive Displays*, pages 1–2, 2019.

[267] A. Sallaberry, C. Muelder, and K.-L. Ma. Clustering, visualizing, and navigating for large dynamic graphs. In *International Symposium on Graph Drawing*, pages 487–498, 2012.

[268] L. Sambrooks and B. Wilkinson. Comparison of gestural, touch, and mouse interaction with fitts' law. In *Proceedings of the ACM Australian Computer-Human Interaction Conference*, page 119–122, 2013.

[269] D. J. Sandin, T. Margolis, J. Ge, J. Girado, T. Peterka, and T. A. DeFanti. The varrier-tm autostereoscopic virtual reality display. In *Proceedings of ACM Transactions on Graphics*, page 894–903, 2005.

[270] T. A. Sandstrom, C. Henze, and C. Levit. The hyperwall. In *Proceedings of the IEEE International Conference on Coordinated and Multiple Views in Exploratory Visualization*, pages 124–133, 2003.

[271] P. Saraiya, P. Lee, and C. North. Visualization of graphs with associated timeseries data. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 225–232, 2005.

[272] D. R. Schikore, R. A. Fischer, R. Frank, R. Gaunt, J. Hobson, and B. Whitlock. High-resolution multiprojector display walls. *IEEE Computer Graphics and Applications*, 20(4):38–44, 2000.

[273] B. Schindler, J. Waser, H. Ribičić, R. Fuchs, and R. Peikert. Multiverse data-flow control. *IEEE Transactions on Visualization and Computer Graphics*, 19(6):1005–1019, 2013.

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

[275] H.-J. Schulz, T. Nocke, M. Heitzler, and H. Schumann. A design space of visualization tasks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2366–2375, 2013.

[276] K. Sedig and P. Parsons. Interaction design for complex cognitive activities with visual representations: A pattern-based approach. *AIS Transactions on Human-Computer Interaction*, 5(2):84–133, 2013.

[277] V. Setlur, S. E. Battersby, M. Tory, R. Gossweiler, and A. X. Chang. Eviza: A natural language interface for visual analysis. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, page 365–377, 2016.

[278] L. Shi, C. Wang, and Z. Wen. Dynamic network visualization in 1.5 D. In *Proceedings of the IEEE Pacific Visualization Symposium*, pages 179–186, 2011.

[279] S. Shiroi, K. Misue, and J. Tanaka. Chronoview: Visualization technique for many temporal data. In *Proceedings of the International Conference on Information Visualisation*, pages 112–117, 2012.

[280] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages*, pages 336–343, 1996.

[281] B. Shneiderman. Direct manipulation for comprehensible, predictable and controllable user interfaces. In *International Conference on Intelligent User Interfaces*, page 33–39, 1997.

[282] B. Shneiderman. The event quartet: How visual analytics works for temporal data. In *Proceedings of the IEEE VIS Workshop on Temporal & Sequential Event Analysis*, 2016.

[283] B. Shneiderman and A. Aris. Network visualization by semantic substrates. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):733–740, 2006.

[284] B. Shneiderman and C. Plaisant. Strategies for evaluating information visualization tools: Multi-dimensional in-depth long-term case studies. In *Proceedings of the AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*, page 1–7, 2006.

[285] B. Shneiderman and C. Plaisant. Interactive visual event analytics: Opportunities and challenges. *IEEE Computer*, 52(1):27–35, 2019.

[286] Y. B. Shrinivasan, D. Gotzy, and J. Lu. Connecting the dots in visual analysis. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology*, pages 123–130, 2009.

[287] Y. B. Shrinivasan and J. J. van Wijk. Supporting the analytical reasoning process in information visualization. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, page 1237–1246, 2008.

[288] T. Siddiqui, A. Kim, J. Lee, K. Karahalios, and A. Parameswaran. Effortless data exploration with Zenvisage: An expressive and interactive visual analytics system. *VLDB Endownment*, 10(4):457–468, Nov. 2016.

[289] K. A. Siek, Y. Rogers, and K. H. Connelly. Fat finger worries: How older and younger users physically interact with PDAs. In *Proceedings of Human-Computer Interaction*, pages 267–280, 2005.

[290] P. Simonetto, D. Archambault, and S. Kobourov. Drawing dynamic graphs without timeslices. In *Proceedings of Graph Drawing*, pages 394–409, 2018.

[291] P. Simonetto, D. Archambault, and S. Kobourov. Event-based dynamic graph visualisation. *IEEE Transactions on Visualization and Computer Graphics*, 26(7):2373–2386, 2020.

[292] M. Spindler, S. Stellmach, and R. Dachselt. PaperLens: Advanced magic lens interaction above the tabletop. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 69–76, 2009.

[293] A. Srinivasan, H. Park, A. Endert, and R. C. Basole. Graphiti: Interactive specification of attribute-based edges for network modeling and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):226–235, 2018.

[294] K. Stein, R. Wegener, and C. Schlieder. Pixel-oriented visualization of change in social networks. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining*, pages 233–240, 2010.

[295] H. Stitz, S. Gratzl, H. Piringer, T. Zichner, and M. Streit. Knowledgepearls: Provenance-based visualization retrieval. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):120–130, 2019.

[296] M. C. Stone. Color and brightness appearance issues in tiled displays. *IEEE Computer Graphics and Applications*, 21(5):58–66, 2001.

[297] M.-A. Storey and H. Muller. Manipulating and documenting software structures using shrimp views. In *Proceedings of International Conference on Software Maintenance*, pages 275–284, 1995.

[298] N. A. Streitz, J. Geißler, and T. Holmer. Roomware for cooperative buildings: Integrated design of architectural spaces and information spaces. In *Proceedings of the International Workshop on Cooperative Buildings*, pages 4–21, 1998.

[299] N. A. Streitz, J. Geißler, T. Holmer, S. Konomi, C. Müller-Tomfelde, W. Reischl, P. Rexroth, P. Seitz, and R. Steinmetz. i-LAND: an interactive landscape for creativity and innovation. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 120–127, 1999.

[300] D. S. Tan and M. Czerwinski. Effects of visual separation and physical discontinuities when distributing information across multiple displays. In *Proceedings of IFIP International Conference on Human-Computer Interaction*, pages 252–255, 2003.

[301] D. S. Tan, D. Gergle, P. Scupelli, and R. Pausch. Physically large displays improve performance on spatial tasks. *ACM Transactions on Computer-Human Interaction*, 13(1):71–99, 2006.

[302] D. S. Tan, D. Gergle, P. G. Scupelli, and R. Pausch. Physically large displays improve path integration in 3D virtual navigation tasks. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 439–446, 2004.

[303] C. Tominski. Event-based concepts for user-driven visualization. *Information Visualization*, 10(1):65–81, 2011.

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

[305] M. Tory and T. Moller. Evaluating visualizations: do expert reviews work? *IEEE Computer Graphics and Applications*, 25(5):8–11, 2005.

[306] D. Tougaw and J. Will. Visualizing the future of virtual reality. *IEEE Computing in Science & Engineering*, 5(4):8–11, 2003.

[307] J. Tukey. *Exploratory Data Analysis*, pages 192–194. Springer New York, 2008.

[308] U. C. Turker and S. Balcisoy. A visualisation technique for large temporal social network datasets in Hyperbolic space. *Journal of Visual Languages & Computing*, 25(3):227–242, 2014.

[309] B. Tversky, J. B. Morrison, and M. Betrancourt. Animation: can it facilitate? *International Journal of Human-Computer Studies*, 57(4):247–262, 2002.

[310] S. van den Elzen, D. Holten, J. Blaas, and J. J. van Wijk. Reducing snapshots to points: A visual analytics approach to dynamic network exploration. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):1–10, 2016.

[311] S. van den Elzen and J. J. van Wijk. Small multiples, large singles: A new approach for visual data exploration. *Computer Graphics Forum*, 32(3):191–200, 2013.

[312] F. Van Ham, H.-J. Schulz, and J. M. Dimicco. Honeycomb: Visual analysis of large scale social networks. In *Proceedings of the IFIP Conference on Human-Computer Interaction*, pages 429–442, 2009.

[313] C. Vehlow, M. Burch, H. Schmauder, and D. Weiskopf. Radial layered matrix visualization of dynamic graphs. In *Proceedings of the IEEE International Conference on Information Visualisation*, pages 51–58, 2013.

[314] A. Vogogias, J. Kennedy, D. Archambault, B. Bach, V. Anne Smith, and H. Currant. Bayespiles: Visualisation support for Bayesian network structure learning. *ACM Transactions on Intelligent Systems and Technology*, 10(1):1–23, 2018.

[315] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. van Wijk, J.-D. Fekete, and D. Fellner. Visual analysis of large graphs: State-of-the-art and future research challenges. *Computer Graphics Forum*, 30(6):1719–1749, 2011.

[316] K. Vrotsou, J. Johansson, and M. Cooper. Activitree: Interactive visual exploration of sequences in event-based data using graph similarity. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):945–952, 2009.

[317] A. Waje, K. Tearo, R. V. Sampangi, and D. Reilly. Grab this, swipe that: Combining tangible and gestural interaction in multiple display collaborative gameplay. In *Proceedings of the ACM International Conference on Interactive Surfaces and Spaces*, pages 433–438, 2016.

[318] A. Walch, M. Schwärzler, C. Luksch, E. Eisemann, and T. Gschwandtner. Lightguider: Guiding interactive lighting design using suggestions, provenance, and quality visualization. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):569–578, 2020.

[319] R. Walker, A. Slingsby, J. Dykes, K. Xu, J. Wood, P. H. Nguyen, D. Stephens, B. L. W. Wong, and Y. Zheng. An extensible framework for provenance in human terrain visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2139–2148, 2013.

[320] G. Wallace, H. Chen, and K. Li. Color gamut matching for tiled display walls. In *Proceedings of the Workshop on Virtual Environments*, pages 293–302, 2003.

[321] R. Walter, G. Bailly, N. Valkanova, and J. Müller. Cuenesics: Using mid-air gestures to select items on interactive public displays. In *Proceedings of the ACM International Conference on Human-Computer Interaction with Mobile Devices & Services*, page 299–308, 2014.

[322] F. Wang, W. Chen, F. Wu, Y. Zhao, H. Hong, T. Gu, L. Wang, R. Liang, and H. Bao. A visual reasoning approach for data-driven transport assessment on urban roads. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, pages 103–112, 2014.

[323] Y. Wang, D. Archambault, H. Haleem, T. Moeller, Y. Wu, and H. Qu. Nonuniform timeslicing of dynamic graphs based on visual complexity. In *Proceedings of the IEEE Visualization Conference*, pages 1–5, 2019.

[324] M. Q. Wang Baldonado, A. Woodruff, and A. Kuchinsky. Guidelines for using multiple views in information visualization. In *Proceedings of the ACM Working Conference on Advanced Visual Interfaces*, pages 110–119, 2000.

[325] C. Ware. *Information visualization: perception for design*. Morgan Kaufmann, 2019.

[326] R. L. Wasserstein and N. A. Lazar. The ASA statement on p-values: Context, process, and purpose. *The American Statistician*, 70(2):129–133, 2016.

[327] M. Wattenberg. Sketching a graph to query a time-series database. In *Proceedings of the ACM SIGCHI Conference Extended Abstracts on Human Factors in Computing Systems*, page 381–382, 2001.

[328] C. Weaver. Conjunctive visual forms. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):929–936, 2009.

[329] R. R. Wehbe, T. Dickson, A. Kuzminykh, L. E. Nacke, and E. Lank. Personal space in play: Physical and digital boundaries in large-display cooperative and competitive games. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, page 1–14, 2020.

[330] S. Wehrend and C. Lewis. A problem-oriented classification of visualization techniques. In *Proceedings of the Conference on Visualization*, page 139–143, 1990.

[331] B. Wei, C. Silva, E. Koutsofios, S. Krishnan, and S. North. Visualization research with large displays. *IEEE Computer Graphics and Applications*, 20(4):50–54, 2000.

[332] C. Williamson and B. Shneiderman. The dynamic homefinder: Evaluating dynamic queries in a real-estate information exploration system. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 338–346, 1992.

[333] M. L. Wittorf and M. R. Jakobsen. Eliciting mid-air gestures for wall-display interaction. In *Proceedings of the ACM Nordic Conference on Human-Computer Interaction*, pages 1–4, 2016.

[334] J. O. Wobbrock, M. R. Morris, and A. D. Wilson. User-defined gestures for surface computing. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 1083–1092, 2009.

[335] K. Wongsuphasawat, J. A. Guerra Gómez, C. Plaisant, T. D. Wang, M. Taieb-Maimon, and B. Shneiderman. Lifeflow: Visualizing an overview of event sequences. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, page 1747–1756, 2011.

[336] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):649–658, 2016.

[337] L. Xiao, J. Gerth, and P. Hanrahan. Enhancing visual analysis of network traffic using a knowledge representation. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology*, pages 107–114, 2006.

[338] R. Xiao, M. A. Nacenta, R. L. Mandryk, A. Cockburn, and C. Gutwin. Ubiquitous cursor: A comparison of direct and indirect pointing feedback in multi-display environments. In *Proceedings of Graphics Interface*, page 135–142, 2011.

[339] K. Xu, A. Ottley, C. Walchshofer, M. Streit, R. Chang, and J. Wenskovitch. Survey on the analysis of user interactions and visualization provenance. *Computer Graphics Forum*, 39(3):757–783, 2020.

[340] S. Xu, C. Bryan, J. K. Li, J. Zhao, and K.-L. Ma. Chart constellations: Effective chart summarization for collaborative and multi-user analyses. *Computer Graphics Forum*, 37(3):75–86, 2018.

[341] J. S. Yi, N. Elmqvist, and S. Lee. TimeMatrix: Analyzing temporal social networks using interactive matrix-based visualizations. *International Journal of Human-Computer Interaction*, 26(11-12):1031–1051, 2010.

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

[343] J. S. Yi, Y. a. Kang, J. Stasko, and J. Jacko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1224–1231, Nov. 2007.

[344] V. Yoghourdjian, D. Archambault, S. Diehl, T. Dwyer, K. Klein, H. C. Purchase, and H.-Y. Wu. Exploring the limits of complexity: A survey of empirical studies on graph visualisation. *Visual Informatics*, 2(4):264 – 282, 2018.

[345] V. Yoghourdjian, T. Dwyer, K. Klein, K. Marriott, and M. Wybrow. Graph thumbnails: Identifying and comparing multiple graphs at a glance. *IEEE Transactions on Visualization and Computer Graphics*, 24(12):3081–3095, 2018.

[346] B. Yost, Y. Haciahmetoglu, and C. North. Beyond visual acuity: the perceptual scalability of information visualizations for large displays. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 101–110, 2007.

[347] Y. Zhai, G. Zhao, T. Alatalo, J. Heikkilä, T. Ojala, and X. Huang. Gesture interaction for wall-sized touchscreen display. In *Proceedings of the ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, page 175–178, 2013.

[348] Y. Zhang, S. D. Bartolomeo, F. Sheng, H. Jimison, and C. Dunne. Evaluating alignment approaches in superimposed time-series and temporal event-sequence visualizations. In *Proceedings of the IEEE Visualization Conference*, pages 1–5, 2019.