JOHANNES KEPLER
UNIVERSITY LINZ

Submitted by
Holger Stitz, MSc.

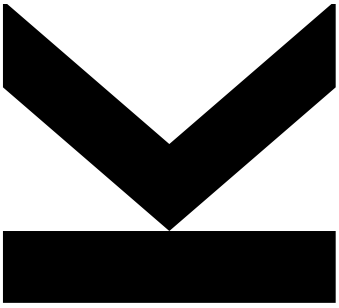
Submitted at
**Institute of
Computer Graphics**

Supervisor and
First Examiner
**Univ.-Prof. Dipl.-Ing.
Dr. Marc Streit**

Second Examiner
**Assoc.-Prof. Dr.
Miriah Meyer**

April 2019

Interactive Focus+Context Analysis of Time-Series and Provenance Data



Doctoral Thesis
to obtain the academic degree of
Doktor der technischen Wissenschaften
in the Doctoral Program
Technische Wissenschaften

**JOHANNES KEPLER
UNIVERSITY LINZ**
Altenbergerstraße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

Abstract

Efficient exploration of large and complex datasets, as for instance, time-series and provenance data, is an ongoing research challenge in visual analytics. Visualizing such datasets in one go often leads to visual clutter, making it hard for users to identify potentially interesting data subsets. A possible solution to reduce the clutter is Focus+Context techniques, which visualize selected regions in greater detail while preserving an overview with reduced details. For large datasets, however, selecting focus regions can become a time-consuming task if each region must be selected individually. Furthermore, in the case of temporal data, the interest in a particular data subset might not remain constant but, rather, shift over time or switch to other data subsets. Consequently, it is necessary to develop Focus+Context solutions tailored to large temporal data. This thesis presents four interactive visualization approaches for highlighting potentially interesting subsets in time-series and provenance data. The solutions utilize modular degree of interest functions that are driven by one or multiple data attributes, the topology of the graph, or a combination of both. The practical applicability of these approaches is demonstrated by means of different case studies from cloud computing, finance, and biomedical research.

Zusammenfassung

Die effiziente Exploration großer und komplexer Datensätze ist eine fortwährende Herausforderung in Visual Analytics. Visualisiert man solche Datensätze als Ganzes führt das in der Regel zu einer chaotischen Darstellung, die es für den Anwender schwierig macht, potenziell interessante Teile aus den Daten zu identifizieren. Eine mögliche Lösung zur Reduzierung der Unordnung sind Fokus- und Kontexttechniken, welche ausgewählte Regionen detaillierter darstellen und gleichzeitig eine Übersicht mit reduzierten Details geben. Für große Datensätze bleibt die Auswahl der Fokusbereiche jedoch eine zeitaufwändige Aufgabe für den Anwender, sofern diese einzeln ausgewählt werden müssen. Des Weiteren ist das Interesse für eine bestimmte Teilmenge im Fall von zeitbasierten Daten möglicherweise nicht konstant, sondern verschiebt sich über die Zeit oder wechselt zu einer anderen Teilmenge. Folglich ist es notwendig, Fokus- und Kontexttechniken zu entwickeln, die speziell auf große zeitbasierten Daten zugeschnitten sind. Diese Arbeit stellt vier interaktive Visualisierungsansätze zur Hervorhebung interessanter Teilmengen aus Zeitreihen und Provenienzdaten vor. Die Lösungen verwenden modulare Funktionen mit denen der Anwender das Interesse für die Teilmengen definieren kann. Die Funktionen können von einem oder mehreren Datenattributen, der Topologie des Graphen oder einer Kombination aus beidem gesteuert werden. Die praktische Anwendbarkeit wird anhand von verschiedene Fallstudien aus den Bereichen Cloud Computing, Finanzwesen und biomedizinische Forschung gezeigt.

Acknowledgements

First and foremost, I wish to thank my supervisor, Marc Streit, for showing me that research in our field can and must go beyond the programming with which we implement our ideas. I appreciate all his contributions of time, ideas, and funding. It was a pleasure to be a PhD student under his guidance.

I would further like to thank my colleague Samuel Gratzl who co-authored many of my publications. We had many fruitful discussions over the years and I am thankful for his support in my research.

This thesis is also the result of collaborations with talented researchers, domain experts, and students that contributed their time and expertise in countless brainstormings, discussions, and prototypes. My thanks go to Wolfgang Aigner, Nils Gehlenborg, Michael Krieger, Stefan Luger, Harald Piringer, Thomas Zichner, and many others.

I would also like to thank my colleagues at the Institute of Computer Graphics for the shared breaks that made our daily work more joyful and diverse.

What remains is to thank the people most important in my life, who helped me get to where I am now. I am deeply grateful to my wife Reem for her never-ending support and love during the past years. Thank you for your understanding and endurance in bearing with (and without) me when the next deadline was coming up. Thanks to my parents and my sister for their support and for believing in me. Last but not least thanks to my friends for the fun we had and still have.

This thesis and individual papers were supported in part by the Austrian Research Promotion Agency (FFG) (840232, 845598, 851460), the Austrian Science Fund (FWF) (P27975-NBL, P25489), the US National Institutes of Health (R00 HG007583), the Harvard Stem Cell Institute, Boehringer Ingelheim Regional Center Vienna, and VRVis Forschungs-GmbH. The VRVis Forschungs-GmbH is funded by COMET – Competence Centers for Excellent Technologies (854174) by BMVIT, BMWFW, Styria, Styrian Business Promotion Agency – SFG and Vienna Business Agency. The COMET Programme is managed by FFG.

Contents

List of Figures

List of Tables

1 | Introduction

Contents

1.1. Motivation and Problem Statement

In recent decades, technological advances in data collection and storage have resulted in significantly larger datasets. The immense growth of data is driven, among others, by the aim of extracting new knowledge from data [Row07]. However, finding and generating insights is often the result of complex and time-consuming exploratory analysis [CEH+09]. Visual analytics attempts to support these extraction processes by combining human capabilities with computational power [KMS+08]. A prerequisite, however, is that the analysis methods must scale to large amounts of data and complex data structures.

The central challenge for visualization, in particular, is the representation of vast amounts of data on limited space [ED07]. Displaying the entire dataset with all its details at once would lead to overplotting and occlusion of entities. The introduced visual clutter diminishes the usefulness of the visualization [ED07] and degrades the user's performance in accomplishing the analysis task [RLMJ05]. Hence, a fundamental goal when developing efficient visualizations is to reduce or avoid visual clutter.

Visual clutter can rarely be reduced solely by (static) visualization and often requires user interaction as an additional component. The result is a visual analysis process that runs iteratively between the user and visualization. The *Information Visualization Reference Model* [Chi00, CMS99] (also known as the *visualization pipeline*) formalizes this process and describes also how data is transformed into a visual representation (see Figure 1.1). As a first step, raw data in arbitrary formats is transformed into data tables including data filtering and generating derived attributes (i.e., metadata). The prepared data is then mapped to visual channels (e.g., size, position, color), resulting in visual structures.

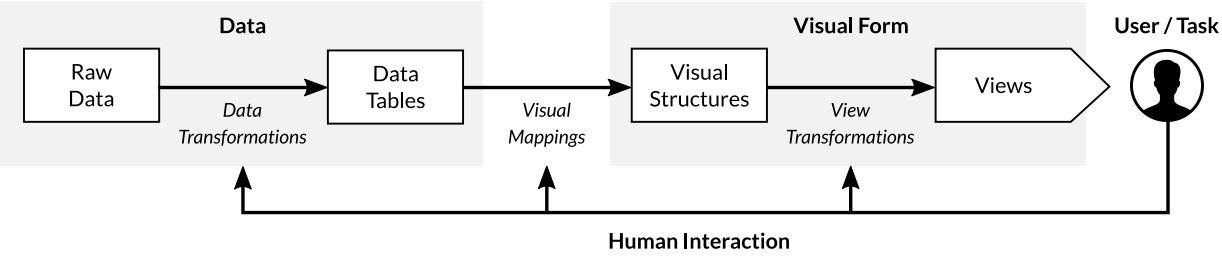


Figure 1.1.: The *Information Visualization Reference Model* by Card et al. [CMS99] describes the transformation process of data into a visual representation and indicates possible locations where users can influence the process.

The last step transforms the visual structures into the final view (i.e., visualization), which is perceived by users. Depending on the task, users interact with this visualization and, thus, influence the various transformation steps along the entire pipeline.

The combination of data, visual representation, and interactions allows many degrees of freedom when creating visual analysis tools. The challenge, however, is to find combinations that effectively support users when analyzing large amounts of data. A well-known technique for interactive visual analysis is *Focus+Context*, which, visualizes selected regions in greater detail while preserving an overview with reduced details [CMS99, Hau06]. One of the earliest approaches was proposed by Furnas in the *Generalized Fisheye View* [Fur86]—an interactive tree visualization, where users can select nodes as focus points based on their interest. A *degree of interest* (DOI) function calculates a DOI value for each node by combining an *a priori importance* (API) for features defined in the dataset (e.g., leaf nodes are more important) and the *distance* of the current node to the selected focus point. As a result, selected nodes have the highest DOI value and are therefore shown with the highest details. Adjacent nodes are visualized with fewer details, and remote nodes have the lowest DOI value and are rendered with the least detail. Naturally, the selected focus nodes and their surroundings draw the user's attention and provide the space to show further information, while the distant nodes preserve the context. Due to their versatility, Focus+Context approaches have also been applied to a large spectrum of data structures (e.g., tabular data, geospatial data) and visualizations [CKB08].

In this thesis, we apply Focus+Context techniques to time-series and provenance data from the domains of cloud computing, finance, and biomedical research. Datasets in these domains usually consist of multiple items or entities, each of which contains one or more attributes. Further, the values of these attributes may change over time and result in time-series data. Examples are the price (attribute) of a stock (entity) at the stock exchange or the memory consumption (attribute) of a server (entity) when monitoring a cloud infrastructure. Typically, value changes of multiple attributes for an entity are recorded simultaneously, resulting in a multi-attribute time series. Additional attributes for stocks are, for example, the opening, highest, lowest, and closing prices per day and, for servers, CPU load and the amount of incoming and outgoing network traffic. Analyzing time-series

data can become complex due to its inherent semantic structure [AMST11]. For example, time has multiple granularities (e.g., minutes, hours, days, and weeks) with various forms of divisions (e.g., 24 hours correspond to one day, while 7 days correspond to one week), and can be combined into various calendar systems (e.g., Gregorian, lunar, national, or fiscal calendars). These special characteristics of time are not only a challenge, but can also support the data modeling process, as values of a time series can be aggregated along the granularities.

Besides the temporal development of attributes, individual entities can be related to one another, forming a multivariate network or graph with nodes and edges. For example, in the stock market, companies can be grouped by economy sectors of a stock index (e.g., S&P500, DAX) or, in cloud computing infrastructures, all servers are connected to enable data exchange and load distribution. Both entities—servers as nodes and connections as edges—can again have one or more attributes that change over time. In addition to these attributes, graph topology may also change over time. The result is a dynamic multivariate graph in which nodes and edges can be added or removed over time [BBDW14].

A special kind of dynamic multivariate graphs are provenance graphs [RESC15], which describe the process of an analysis. In biomedical research, for example, every step of an analysis is recorded, which thus ensures subsequent communication and reproducibility of the results. In addition, there are new possibilities for the analysis itself, as users can restore previous analysis steps and continue on a different path in their exploration. In this case, branches are introduced in the (previously) linear series of analysis steps, resulting in complex, dynamic graph structures.

The complexity of the aforementioned data structures, the temporal dimension, and the sheer amount of data require new visualization approaches [WEF+14]. In the course of this thesis, we address these challenges with novel Focus+Context techniques for time-series and provenance data.

A key requirement for Focus+Context techniques is the abstraction of data on multiple levels, which reduces the amount of data while preserving important features. Examples for data abstraction are aggregation, clustering, or dimensionality reduction techniques. Time-series data and provenance graphs can be abstracted by employing the temporal dimension (e.g., aggregating hours to days) or graph topology (e.g., aggregating nodes into super- or metanodes) making Focus+Context techniques applicable.

The selection of focus points becomes a challenging task when applying Focus+Context techniques to large datasets with lots of entities. In the *Generalized Fisheye View* [Fur86], users can select interesting nodes individually using direct manipulation. However, when each entity must be selected individually, the process of identifying and selecting potentially interesting entities as focus points can become tedious for large datasets. For time-series data and provenance graphs, examples are the selection of entities with a certain attribute value or highlighting a path in a graph. Hence, Focus+Context approaches

must provide the means for users to express their interest and **facilitate the selection of focus points**.

A requirement that is unique to Focus+Context visualizations of temporal data is the shift of interest over time. The users' interest in a particular entity might not remain constant but, rather, shift over time and switch to other entities. For example, a stock is interesting as long as the price is above a certain value, but it becomes less interesting when the price decreases. Hence, Focus+Context approaches for temporal data must **consider and encode the temporal development of entities**.

Besides the aforementioned requirements, visualizations must be tailored to the users' needs and match their tasks. Understanding these tasks is important in order to design effective visual analysis solutions [MA14] and make them comparable [Mun14, p. 36]. Generalized task taxonomies as, for instance, proposed by Shneiderman [Shn96], Schulz et al. [SNHS13], or Brehmer et al. [BM13] provide general guidance for visualization designers. However, for the presented approaches in this thesis, we consider additional task taxonomies tailored to time-series data and multivariate graphs.

Shurkhovetskyy et al. [SAAF18] proposed a task taxonomy with the focus on abstraction methods of large time-series data. According to the authors, the key user task when working with time-series data is the discovery of particular patterns of interest [SAAF18], such as daily or seasonal patterns. However, in many cases, users are also interested entities deviating from a pattern. For example, in cloud computing users may want to investigate entities showing anomalies like a high usage of bandwidth or, in a stock market context, users may want to find companies with a negative long-term development but a recent positive trend for future investments. Focus+Context techniques can support this task, as users can specify patterns for entities they are interested in. Furthermore, matching or deviating entities from the pattern can be visually highlighted while preserving the remaining entities as context.

We use the additional taxonomy from Pretorius et al. [PPS14] to describe the user tasks for multivariate graphs. This taxonomy generalizes the tasks proposed by Lee et al. [LPP+06] and provides a set of 25 tasks grouped into four categories. In the following, we briefly summarize the categories and describe how they relate to Focus+Context techniques. We refer the reader to the publication [PPS14] for further details about the different tasks.

Task A: Structure-based (topology-based) tasks employ the graph topology and relationship between nodes (e.g., adjacency, common connection, or connectivity). A representative Focus+Context approach supporting this task category is the *Generalized Fisheye View* [Fur86], where selected focus points also highlight adjacent nodes. In the case of provenance graphs, Focus+Context techniques can, e.g., extract the shortest path between two selected nodes or highlight clusters of selected nodes.

Task B: Attribute-based tasks infer knowledge about nodes and links and their attributes (e.g., find nodes with specific attribute values). In the context of cloud computing, a Focus+Context approach could support this task category by selecting nodes with a high bandwidth usage (link attribute) or high CPU load (node attributes). Note that this task can also be applied to regular time-series data, where items with a particular attribute value are selected. For example, if users wanted to highlight stocks with a market share greater than a given threshold.

Task C: Browsing tasks infer further knowledge by following a given path in a graph or revisiting a previously visited entity. Focus+Context approaches excel in this task category by highlighting the path between two selected graph nodes, or emphasizing recently selected nodes and thus support revisiting them.

Task D: Estimation tasks aim to gain a more complete understanding of the information (e.g., characterize clusters, common attributes on nodes and links). Similar to Task B, Focus+Context approaches supporting this task category could highlight entities with a particular attribute value. Since selected nodes are shown with more details, users gain a better understanding of these entities. In an additional subcategory, Pretorius et al. address graph changes over time and the comparison of different time steps. The analysis of the temporal development aligns with the aforementioned taxonomy from Shurkhovetskyy et al. [SAAF18]. Focus+Context approaches can address the comparison task by extracting similar entities from different time steps and highlighting similarities and differences.

We believe that Focus+Context techniques support the visual analysis process of time-series and provenance data, if they meet the aforementioned requirements and tasks. However, specialized DOI functions, visual representations, and interaction capabilities are necessary to enable an effective visual analysis.

1.2. Approach

We address the previously introduced requirements and tasks with Focus+Context techniques tailored to time-series and provenance data. The approaches utilize modular DOI functions driven by one or **multiple data attributes**, **the topology of the graph**, or **a combination of both**. Before describing our approaches in more detail, we generalize the applied Focus+Context techniques and contextualize them as processes along the visualization pipeline.

Our Focus+Context process, shown in Figure 1.2, is based on the work by Elmqvist and Fekete [EF10] that proposes a model for multiscale representations using hierarchical aggregation. Similar to the visualization pipeline (see Figure 1.1), the Focus+Context process

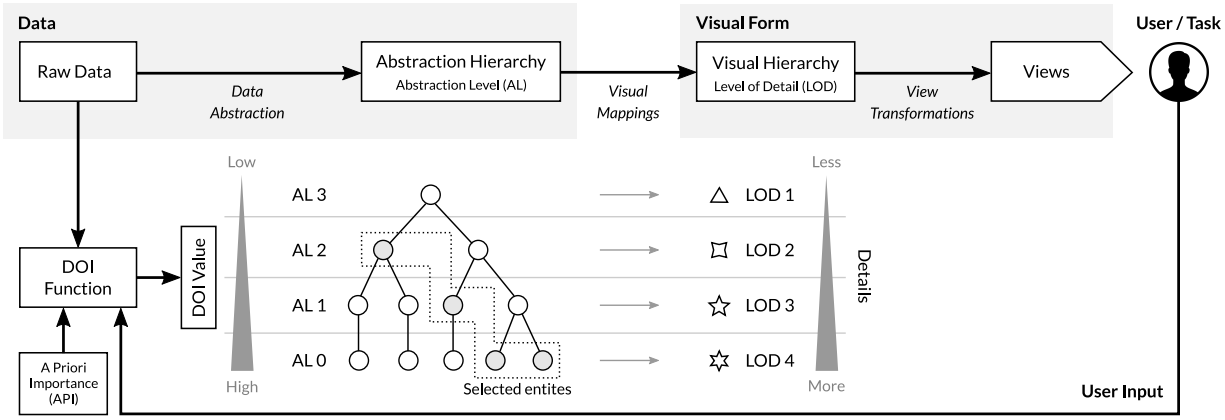


Figure 1.2.: Focus+Context process. An abstraction hierarchy is generated from the raw data and mapped to a visual hierarchy. Users select interesting entities and attributes from the visualization. Considering the users' input, the DOI function computes a DOI value for each entity, which determines the aggregation level and the level of detail (LOD). The updated view visualizes entities of interest in more details while preserving an overview with reduced details as context.

can be divided into a data preparation part and a rendering part. Starting with the data preparation, the raw data is abstracted multiple times, forming an abstraction hierarchy with different *abstraction levels* (ALs). The applied abstraction methods, for instance, aggregation, vary depending on the provided structure of the raw data. Examples are multiple graph nodes that are aggregated into a supernode [vLKS+11, Wat06] or time-series data that is aggregated along the granularities [SAAF18].

Similar to the visual mapping of the visualization pipeline, each abstraction level is mapped to a *level of detail* (LOD) that combines multiple visual encodings. The LODs are ordered in a visual hierarchy, from less detailed representations for highly abstracted data and vice versa. The actual rendering of the visual representation is equal to the visualization pipeline (see Section 1.1). The most common approach is that each LOD represents a particular aggregation level. Based on their tasks, users interact with the visualization and select the entities or attributes they are interested in. The interactions again trigger an update cycle of the view.

The interest in a particular item or data subset is quantified by a DOI function, which incorporates the raw data, the API, and the user input, resulting in a DOI value for each entity. The DOI value determines the user's interest in an entity and is normalized to the unit interval $[0, 1]$, with 0 for low interest and 1 for high interest [Hau06]. Mapping the DOI value to the aggregation hierarchy discretizes the DOI scale. Hence, entities with low DOI values correspond to higher ALs and, due to a low LOD, are visualized with fewer visual details. The visual representation is updated after the LOD is determined for every

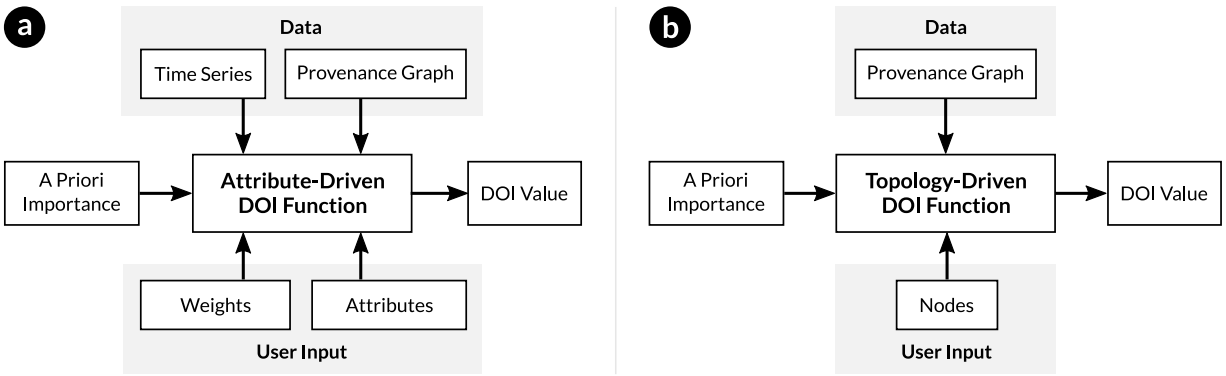


Figure 1.3.: Two approaches to calculate DOI values from given raw data, a priori importance, and user input. (a) The attribute-driven DOI function operates on time-series and provenance data. Users can select multiple attributes and weigh them based on user interest. (b) The topology-driven DOI function operates only on provenance data. In this case, users can select two or more nodes as input.

entity (or aggregate). Based on the updated view, users can, for instance, select another region of interest and continue their analysis [SSS+14].

In the course of this thesis, we developed and applied DOI functions for time-series and provenance data. Figure 1.3a shows the *attribute-driven DOI* function that combines multiple selected attributes into a single DOI value. The function can operate on multi-attribute time-series data or employ node attributes on provenance graphs. Users can select multiple attributes and weigh them based on their interest (Task B). The *topology-driven DOI* function, shown in Figure 1.3b, operates solely on provenance graphs and requires two or more selected nodes and can, for instance, be used to highlight paths (Tasks A and C). In both cases, the output of the DOI function is a DOI value for each output (as described above). In the case of provenance graphs, both DOI functions can be applied to address use cases where nodes with certain attribute values along paths should be highlighted.

1.3. Contributions

This thesis presents four interactive visualization approaches for highlighting interesting subsets in time-series and provenance data. We apply the previously introduced topology- and attribute-driven DOI functions from Section 1.2 and show how each function type supports the introduced user tasks (see Table 1.1). In addition, we demonstrate the practical applicability of our solutions by means of real-world case studies from cloud computing, finance, and biomedical research.

	CloudGazer	ThermalPlot	AVOCADO	KnowledgePearls
DOI Function	topology-driven	attribute-driven	attribute- & topology-driven	attribute-driven
A: Structure	✓	-	✓	-
B: Attributes	-	✓	✓	✓
C: Browsing	✓	-	✓	✓
D: Estimation	-	✓	-	-

Table 1.1.: Supported tasks and used DOI function of the presented approaches in this thesis.

Our first approach *CloudGazer* [SGKS15] introduces a solution for monitoring and optimizing a heterogeneous cloud-based network. It supports Tasks A and C by utilizing a topology-driven DOI function to extract a path of components through the network. *CloudGazer* works well for the analysis of bottlenecks across different network components, but lacks an overview that communicates the temporal development and current status of each component in the network. We address this shortcoming in the follow-up work *ThermalPlot* [SGAS16] and demonstrate this approach by means of a case study from the financial domain. *ThermalPlot* provides an overview for hundreds of items by utilizing an attribute-driven DOI function that reduces a combination of multiple attributes over time into a single point (Tasks B and D). Plotting them into a unique visualization space reveals the long- and short-term development for each item within a selected time span. We combine *ThermalPlot* and *CloudGazer* to describe how administrators can monitor the current status of a cloud-based network and investigate the impact of predicted incidents and their recommended solutions.

During the development of *CloudGazer* and *ThermalPlot* we discovered that users often tried to recall the chain of previous actions to understand the current status (of the network). Tracking these interactions and the corresponding analysis states results in large provenance graphs that makes it challenging for users to find states of interest. With *KnowledgePearls* [SGP+18], we present a solution for the efficient retrieval of analysis states. *KnowledgePearls* provides an interface for querying and exploring analysis states based on their similarity to a partial definition of a requested analysis state. The approach supports Tasks B and C by utilizing an attribute-driven DOI function. We demonstrate the value and utility of *KnowledgePearls* by integrating it into *Ordino* [SGS+19], a drug target discovery tool that allows users to flexibly rank and explore genes, cell lines, and tissue samples.

Another purpose of provenance data (besides recall) is replication [RESC15], which became particularly important in biomedical research after previously published findings could not be reproduced [Kai15, BI15, Buc15, HG13, BE12]. The discovery was a shock for the whole community, but triggered several efforts. We address this issue in *AVOCADO* [SLSG16] and present an interactive provenance graph visualization to review and ensure reproducibility. The approach supports Tasks A, B, and C by combining attribute-

and topology-driven DOI functions to expand only the paths of the graph with certain attributes. We integrated *AVOCADO* in the *Refinery Platform* [1], a data visualization and analysis system that manages the provenance data of biomedical workflows, along with their corresponding input and output files. A common output of workflow executions is multivariate tabular data. Depending on the workflow parameters, multiple versions of the same table exists. Understanding what exactly has changed between different version is challenging. In *TACO* [NSH+18], we present an interactive comparison tool to analyze the changes between multiple table versions over time.

1.3.1. Primary Publications

The following list of peer-reviewed publications forms the core of this thesis. Individual chapters are based on these publications. The publications, sorted by date, are:

CloudGazer: A Divide-and-Conquer Approach to Monitoring and Optimizing Cloud-Based Networks [SGKS15]: **Holger Stitz**, Samuel Gratzl, Michael Krieger, and Marc Streit. *Proceedings of IEEE Pacific Visualization Symposium (PacificVis '15), pp. 175–182.* Acceptance Rate: 30.4%.

Visualizing large dynamic graphs, such as cloud-based networks, with temporal attributes on nodes and edges is challenging due to the clutter introduced when visualizing the overall network. In *CloudGazer*, we increase the scalability by splitting the graph into semantic perspectives, which provide a simpler view of the network. The user can select a focus perspective while the remaining perspectives are presented as an overview. *CloudGazer* uses a topology-driven DOI function to extract adjacent nodes across perspectives. By this means, users can analyze the relationship and connection distance between different network components. Moreover, the *CloudGazer* prototype provides monitoring capabilities for cloud-based networks and supports the streaming of live data or the analysis of historical data.

In collaboration with the other co-authors of the paper, the author of this thesis designed the technique, implemented the prototype, and wrote the manuscript.

[1] `http://refinery-platform.org`

ThermalPlot: Visualizing Multi-Attribute Time-Series Data Using a Thermal Metaphor [SGAS16]: **Holger Stitz**, Samuel Gratzl, Wolfgang Aigner, and Marc Streit. *IEEE Transactions on Visualization and Computer Graphics, 22(12), pp. 2594–2607.* Impact Factor: 1.4.

Simultaneously analyzing trends for combinations of multiple attributes from large time-series data is an important task (Tasks B and D). In *ThermalPlot*, we address these tasks by summarizing complex combinations of multiple attributes over time using an attribute-driven DOI function. *ThermalPlot* provides a scalable overview of large item collections by encoding the items' DOI value in position and using multiple LODs. A use case is the analysis of companies from a stock index (e.g., S&P500, DAX). *ThermalPlot* facilitates the identification and comparison of companies that show a potentially interesting temporal development by combining short- and long-term value developments.

In collaboration with the other co-authors of the paper, the author of this thesis designed the technique and wrote the manuscript. In addition, he also implemented major parts of the prototype.

An earlier version of *ThermalPlot* was presented as a poster at the *IEEE Conference on Information Visualization* (InfoVis '15) and won the *Honorable Mention Poster Award*.

Further, the author of this thesis adapted ThermalPlot for the *KPMG Data Observatory*, an immersive multiscreen presentation setup located at the Imperial College London.

AVOCADO: Visualization of Workflow-Derived Data Provenance for Reproducible Biomedical Research [SLSG16]: **Holger Stitz**, Stefan Luger, Marc Streit, and Nils Gehlenborg. *Computer Graphics Forum (EuroVis '16), vol. 35, no. 3, pp. 481–490.* Acceptance Rate: 27.3%.

Provenance graphs from data-driven biomedical analyses contain data for dozens or hundreds of samples. To communicate and reproduce these multistep analysis workflows, it is crucial to visualize the provenance graph at different levels of aggregation. In *AVOCADO*, we reduce the graph complexity using hierarchical and motif-based aggregation and combine topology- and attribute-driven DOI functions to expand parts of the graph relevant to users. For example, users can combine different attributes to expand matching nodes into the highest LOD (while non-matching remain aggregated) or select a node to examine the path to the origin or later developments. Both methods can be combined to expand only paths with certain attributes.

In collaboration with the other co-authors of the paper, the author of this thesis designed the technique, supervised the implementation of the prototype, and wrote the manuscript.

An earlier version of *AVOCADO* was presented as a poster at the *IEEE Conference on Information Visualization* (InfoVis '15) and won the *Best Poster Award*.

KnowledgePearls: Provenance-Based Visualization Retrieval [SGP+18]: **Holger Stitz**, Samuel Gratzl, Harald Piringer, Thomas Zichner and Marc Streit. *IEEE Transactions on Visualization and Computer Graphics (VAST '18), 25(1), pp. 120–130, 2019.* Acceptance Rate: 25.6%

Tracking user interactions with a visual analysis system results in large provenance graphs that can be used as knowledge base to recall previous states of the analysis. With *KnowledgePearls*, we present a solution for the efficient retrieval of these analysis states using an attribute-driven DOI function. Users formulate search queries that are compared to all analysis states stored in the provenance graph. All matching states are potentially interesting to the user. However, states that match all search terms have a higher similarity and are consequently ranked higher. We employ the relationships of adjacent nodes and group them to reduce the visual clutter.

In collaboration with the other co-authors of the paper, the author of this thesis designed and implemented the prototype and wrote major parts of the manuscript.

An earlier version of *KnowledgePearls* was presented as poster at the *IEEE Conference on Visual Analytics Science and Technology* (VAST '17) and won the *Best Poster Award*.

1.3.2. Secondary Publications

In addition to the primary publications that form the core of this thesis, the author contributed to several related peer-reviewed papers. The following list contains an overview, including a short description of each paper, the author's contributions, and how each paper relates to this thesis.

TACO: Visualizing Changes in Tables Over Time [NSH+18]: Christina Niederer, **Holger Stitz**, Reem Hourieh, Florian Grassinger, Wolfgang Aigner, and Marc Streit. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '17), 24(1), pp. 677–686, 2018.* Acceptance Rate: 22.9%.

Multivariate tabular data can undergo changes in both structure and content, which results in multiple versions of the same table. For example, in biomedical analyses, as described in *AVOCADO* (see Chapter 5), tables can be the output of multistep analysis workflows and result in different versions when the parameters of these workflows are changed. Understanding what exactly has changed between versions in terms of additions/deletions, reordering, merges/splits, and content changes is challenging. We developed TACO, an interactive comparison tool that addresses these challenges and visualizes the differences between multiple tables at three LODs. At the highest level, we show the aggregated differences between multiple table versions over time. Users can drill down and visualize

the aggregated changes by selecting two table versions. A detailed pairwise comparison can be loaded on demand.

In collaboration with the other co-authors of the paper, the author of this thesis designed the technique and wrote the manuscript. In addition, he implemented major parts of the prototype.

Ordino: A Visual Cancer Analysis Tool for Ranking and Exploring Genes, Cell Lines and Tissue Samples [SGS+19]: Marc Streit, Samuel Gratzl, **Holger Stitz**, Andreas Wernitznig, Thomas Zichner, Christian Haslinger. *Oxford Bioinformatics, 2019.* Impact Factor: 5.481.

Data-driven pharmaceutical research is challenging due to both heterogeneity and size of the data. Often multiple attributes must be considered in combination to identify, for instance, genes that could serve as potential drug targets or biomarkers. In Ordino, users can interactively prioritize, filter, and explore tabular data in a first step. Once the user has selected one or multiple items in the ranked list, a collection of possible follow-up detail views for exploring the current selection is displayed. Ordino follows a Focus+Context approach, where the focus view is shown on the right, and the previous focus view is shown as context on the left. New views are pushed from the right to the list of open views. Users are able to close the view and horizontally scroll back to previous views at any time.

All user interactions are stored as provenance graphs in analysis sessions that can be shared with colleagues for presentation and further exploration [GLG+16]. By integrating *KnowledgePearls* into Ordino, as described in Section 6.6, analysts can search for similar analysis states in provenance graphs and, hence, increase the confidence in potential findings when independently identified by multiple experts.

The author contributed to the initial concept, design, and implementation of the platform and the manuscript.

1.4. Structure

This thesis is structured as follows:

Chapter 2 categorizes and discusses related work for Focus+Context techniques for graph and time-series data. Afterward, we survey the literature for the visualization and exploration of time-series and provenance data in more detail. Furthermore, we discuss works in the field of information retrieval for provenance graphs, which is relevant for the *KnowledgePearls* approach described in Chapter 6.

Chapter 3 introduces the *ThermalPlot* technique that summarizes complex combinations of multiple attributes over time using an attribute-driven DOI function and provides a scalable overview for hundreds of items using multiple LODs. This chapter is based on [SGAS16].

Chapter 4 describes *CloudGazer*, a solution for monitoring and optimizing cloud-based networks. In *CloudGazer* the dynamic graph is divided into semantic perspectives. Lost inter-perspective relationships are extracted using a topology-driven DOI function and reintroduced on demand as dynamically created inlays. In addition, we present a concept that combines *CloudGazer* with *ThermalPlot* to investigate performance predictions for the network. This chapter is based on [SGKS15].

Chapter 5 presents *AVOCADO*, an interactive visualization for large provenance graphs from the biomedical domain. The approach reduces the complexity of the graph using hierarchical and motif-based aggregation and expands regions of interest by combining attribute- and topology-driven DOI functions. This chapter is based on [SLSG16].

Chapter 6 introduces *KnowledgePearls*, a solution for the efficient retrieval of analysis states from large provenance graphs using an attribute-driven DOI function. This chapter is based on [SGP+18].

Chapter 7 concludes this thesis by summarizing the work, discussing the presented contributions, and indicating possible directions for future research.

2 | Related Work

Contents

This thesis builds on prior work on the visual analysis of time-series and provenance data. In this chapter, we survey the relevant work from each domain and discuss how this dissertation builds on it. We begin with Focus+Context techniques for graphs and time-series data. Then, we continue with multi-attribute time-series data and discuss different approaches to analyze the temporal dynamics of data. Next, we explain the exploration of dynamic graphs and provenance graphs. The last section introduces information retrieval as an alternative approach to find nodes of interest in large provenance graphs.

2.1. Focus+Context

Focus+Context techniques aim to reduce visual clutter by magnifying or highlighting selected regions in greater detail while preserving an overview with reduced details as context in the same view [CMS99, Hau06]. In the following, we discuss related Focus+Context techniques for static and dynamic graphs as well as time-series data.

2.1.1. Focus+Context for Graphs

One of the earliest Focus+Context techniques has been proposed by Furnas [Fur86] to visualize tree structures using an API and the distance of a selected node. Later techniques introduce space-filling tree layouts [CN02], increase the number of nodes [HC04], or leverage tree layouts to explore networks [LPB+06]. Van Ham and Perer [vHP09] extend Furnas' Focus+Context approach for exploring large graphs and propose a three-step model, as illustrated in Figure 2.1: (i) users search for a node of interest, (ii) the sys-

tem extracts an initial subgraph around that focus point, and (iii) users can expand the subgraph in different directions based on visual cues.

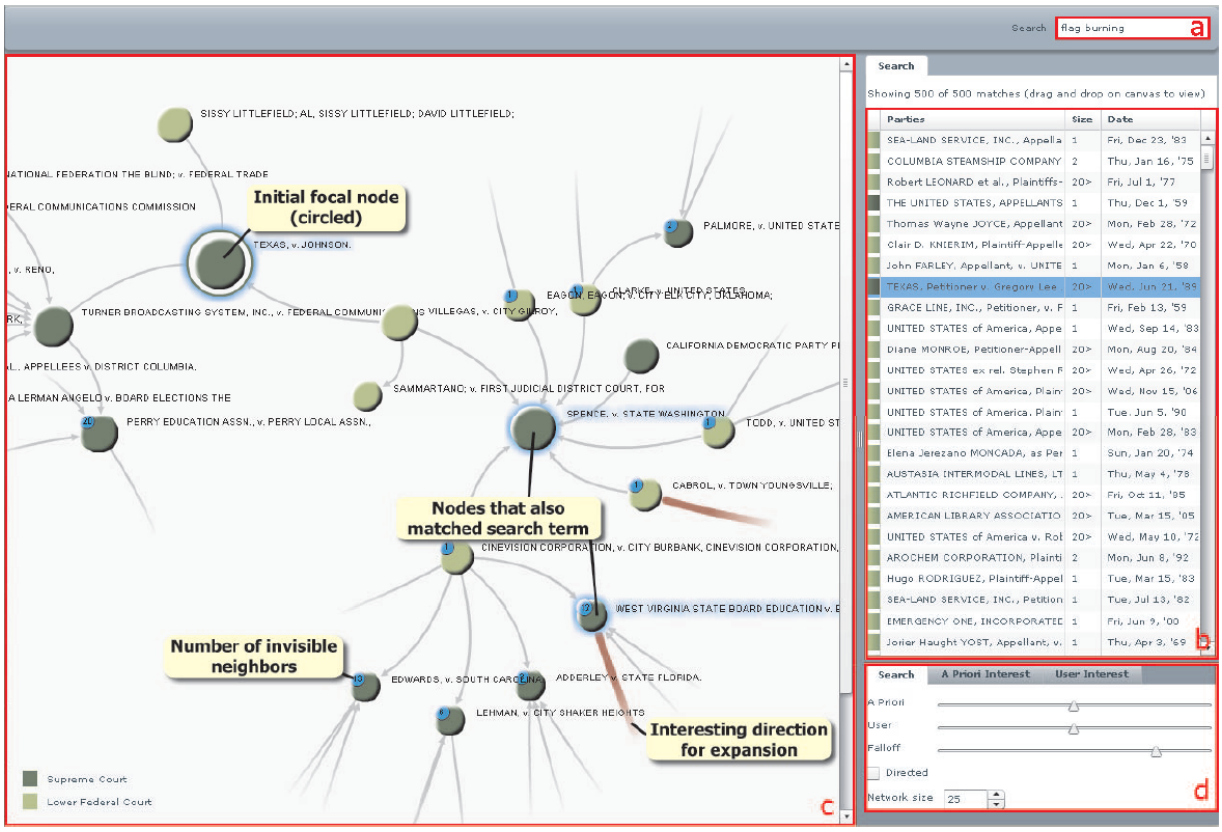


Figure 2.1.: Van Ham and Perer [vHP09] propose a DOI approach for the exploration of graphs. Users search for a term of interest (a) and drag a node from the search results (b) into the main screen (c). Users can modify the different components of the DOI function and the size of the subgraph (d). This Figure is taken from [vHP09].

Subsequent works extend the approach and consider the browsing history when retrieving nodes [CLWM11, PVH12, PKL+17], improve the visual cues for orientation and navigation [MSDK12, PVH12, GST13], or combine different DOI functions to filter subgraphs [VKB+15, KRD+15]. All these approaches have in common that they leverage DOI functions to extract subsets of the graph, which is similar to our topology-driven DOI function. However, existing approaches narrow the context to only a few nodes, and users must manually select additional subgraphs to get further context. In contrast, the approaches proposed as part of this thesis provide an overview of the entire graph, either by splitting it into multiple perspectives and visualizing them as thumbnails (see *CloudGazer* in Chapter 4), or by applying multi-level aggregations to provide a comprehensive context (see *AVOCADO* in Chapter 5).

The aforementioned approaches only apply to static graphs. Focus+Context approaches that apply to dynamic graphs are rare [BBDW14]. A notable exception is the approach by Abello et al. [AHSS14], where users can define flexible modular DOI functions to investigate the temporal evolution of a graph. However, their approach does not work well for the characteristics of provenance graphs, as these graphs often contain recurring patterns (in the case of workflow provenance) or many branches (in the case of interaction provenance graphs). In *AVOCADO* (see Chapter 5), we aggregate nodes that share the same time point hierarchically and apply motif-based aggregation to nodes from different time steps to preserve the structure of the graph. Users can then expand nodes of interest based on a selected time range.

2.1.2. Focus+Context for Time-Series Data

The analysis of time-series data with Focus+Context techniques can be divided into two groups: (i) distorting the visual representation using magnification lenses to highlight interesting parts or patterns of the time series itself, and (ii) arranging multiple time series according to their importance to support users in the task of finding items of interest.

Lens-based techniques are similar to Focus+Context approaches, as they provide additional details by altering the visualization. The difference is that the effect is locally constrained to the shape of the lens, while in Focus+Context approaches, the changes affect the visualization globally [TGK+14]. Examples for lenses that operate on time-series data are *SignalLens* [Kin10], *ChronoLens* [JCPB11], and *CloudLines* [KBK11]. For further approaches, we refer the reader to the extensive survey about lenses by Tominski et al. [TGK+14].

In this thesis, we focus on the second group, which addresses the comparison and highlighting of potentially interesting time series from a collection of time series. Hao et al. [HDKS05] propose a space-filling layout, similar to tree maps [JS91], for multiple time series, each with a single attribute. Items with the most interesting pattern (e.g., the highest sale across different regions) are placed at the top and given more visual space, compared to less interesting items that are visualized in a condensed fashion. In a follow-up work [HDKS07], the authors partition a time series in multiple bins, determine the interest for each bin, and visualize the entire time-series data using a multi-resolution layout, as shown in Figure 2.2. Accordingly, more visual space is given to interesting parts of the time series, resulting in a more compact overall layout.

The proposed approaches work well for the exploration of single attributes over time (e.g., CPU load). However, in many real-world scenarios, the combination of multiple attributes and many items must be considered. In *ThermalPlot* (see Chapter 3), we employ an attribute-based DOI function to combine multiple attributes over time and provide an overview of the temporal development for hundreds of items.

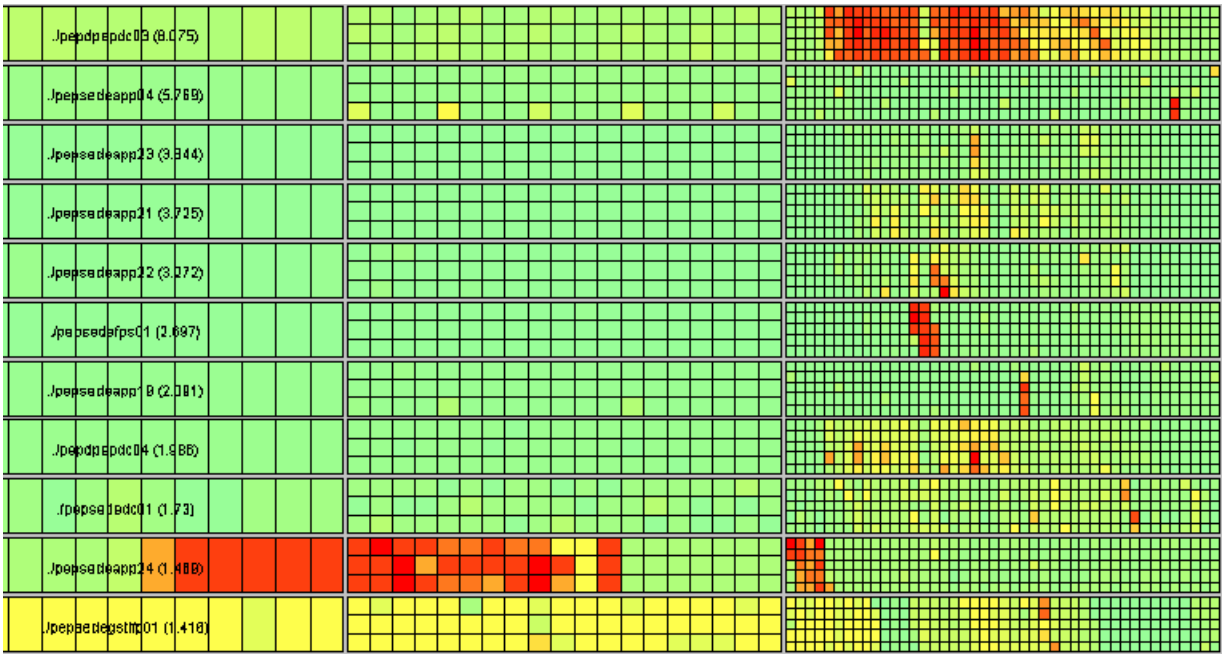


Figure 2.2.: Hao et al. [HDKS07] propose a multi-resolution approach to compare the continuous CPU utilization history (3 days = 864 values) of eight servers over time. The example shows three resolution levels dedicating more visual space to most recent values (to the left). The comparison reveals little correlation between the different servers. This Figure is taken from [HDKS07].

2.2. Multi-Attribute Time-Series Data

Due to the broad applicability of multi-attribute time-series data, a vast body of related work exists. For an extensive survey on the special characteristics of time in general, and a systematic discussion of available techniques, we refer the reader to the book on time-oriented data visualization [AMST11] and the corresponding online collection of available techniques[1]. Shurkhovetskyy et al. [SAAF18] discuss the abstraction of large time-series data, which is required for the visualization and analysis at different levels of detail (see Section 1.3).

Exploring complex real-world phenomena almost always requires taking into account a number of interrelated attributes along with their changes over time. To effectively address this goal, it is necessary to tackle two challenges: (1) the integration and comparison of multiple heterogeneous attributes for a collection of items, and (2) the extraction of temporal dynamics on multiple levels. In the following, we divide the related work along the lines of these two challenges and discuss existing solutions together with their respective strengths and weaknesses.

[1] `http://survey.timeviz.net`

2.2.1. Multi-Attribute Item Comparison

Addressing challenge (1)—integrating and comparing multiple heterogeneous attributes of an item collection—is difficult because the comparison should ideally work in both dimensions: across multiple attributes of a single item and across a single attribute of multiple items. Both can be addressed, for instance, by superimposing multiple curves in a line chart [GAW+11], by stacking multiple line or horizon charts [HKA09], or by using other pixel-based techniques [AKK96, KAK95]. However, these solutions often do not work for comparisons in both directions simultaneously. In *ThermalPlot*, we summarize multiple-attributes using a modular DOI function and visualize the aggregated items based on their DOI values.

2.2.2. Visual Analysis of Temporal Dynamics

To investigate the dynamics of time-dependent data, referred to as challenge (2), an analyst needs to take into account temporal patterns on both a global (coarse temporal granularity) and a local scope (fine temporal granularity). The literature differentiates between two possible kinds of visual mappings for encoding time-dependent data: static (mapping time to space) and dynamic (mapping time to time) [AMST11].

Mapping Time to Time

The main advantage of using animation to communicate temporal changes is its intuitiveness, as it is the way how we perceive changes in real world. Users might also find it more exciting and fun to use [RFF+08]. Examples for animated visualizations are the well-known *Gapminder Trendalyzer*[2] that uses bubble charts to plot demographic changes over time and animated scatter plots for stock market data [TK07].

While the use of animation can be a good design choice for presentation purposes, animation is known to be problematic for analysis tasks [RFF+08, KPS14]. Also, animation can be effective to encode smooth changes, however, users might get confused if the changes affect too many data items, if items do not move in synchrony, or if items reverse their tracks over time [RFF+08]. The reason for these limitations is the human's limited ability to follow, memorize, and compare information across time steps [Fis10]. The problem can be alleviated to a certain extent by allowing users to pause, replay, and adjust animation speed. However, due to the shortcomings, we chose to not rely on animation as a primary encoding principle to summarize temporal changes in large item collections.

[2] `https://www.gapminder.org`

Mapping Time to Space

The principle of mapping time to space utilizes position to encode change and temporal patterns. A wide array of techniques exist for investigating seasonal patterns and trends (e.g., *Cycle Plot* [Cle93] and *GROOVE* [LAB+09]). However, they do not scale well to multiple attributes, a large set of items, or long time-series [Gle18]. While heatmaps and pixel-based techniques scale better, it can become difficult for users to compare temporal patterns and trends across items or attributes [ACG14].

In contrast to animation, small multiples [Tuf83] and trajectories (aka traces) are better suited for analysis tasks performed on time-dependent data [RFF+08]. The small multiple representation can be a plot, a glyph [FFM+13], or any other visualization. While small multiples can be used to create an effective overview (e.g., *LiveRAC* [MMKN08]), they do not scale beyond a couple of dozen items or attributes. The sequence view in [STKF07] is an example for a large small multiple matrix showing line charts in 66 columns and 83 rows. However, such large matrices only work in combination with automated detection and guidance mechanisms that point users to relevant parts.

In *ThermalPlot* (see Chapter 3) we summarize combinations of multiple attributes over time using an attribute-based DOI function and map the resulting DOI value to an item's position, the most salient visual variable.

Trajectories

Trajectories are another option to encode change over time. Trajectories are visual traces that appear when continuously plotting the temporal development of items in 2D (e.g., [RAM+11, STKF07]) or 3D [VM04]. The emerging paths can then be used to compare patterns across multiple items. However, multiple overlapping trajectories can result in visual clutter, reducing their effectiveness [RAM+11]. Clustering algorithms can alleviate the problem by aggregating trajectories based on some kind of similarity metric [STKF07, vLBRS09]. Ziegler et al. [ZJGK10], for example, cluster companies that belong to the same industry sector and then present the trajectories for each company in a small multiple view that is grouped by cluster. However, this solution comes with extra cognitive load for comparing the items. In *ThermalPlot* (see Chapter 3), trajectories can be displayed on demand for selected items only in order to keep the visual clutter at a minimum.

DimpVis [KC14] is a technique where users directly interact with the trajectories for navigating in time. While the technique is very intuitive, it lacks direct support when items remain at one specific position for several time steps and it can get difficult to work with complex trajectories. Instead of allowing users to directly interact with the trajectories, we decided to let users select a single time step via line charts in a detail view. The corresponding parts of the trajectories are then highlighted accordingly.

In summary, we could not identify an existing technique that allows users to address both challenges for a large collection of multi-attribute items.

2.3. Provenance Graph Exploration

Visual analysis sessions can be recorded at various levels resulting in different types of provenance covering different aspects of the visual analytics process [KMS+08]. Ragan et al. [RESC15] propose five different types of provenance information: *data provenance* containing modifications applied to the dataset, *visualization provenance* containing the history of visualization states, and *interaction provenance* containing the history of user actions and commands with a system, *insight provenance* containing the analytical findings and hypothesis, and *rational provenance* containing the history of reasons and intentions. The approaches presented in this thesis operate on data, visualization, and interaction provenance. The remainder of this section discusses the related work in the context of data provenance graphs, which are common in the biomedical workflows. However, most of the discussed work is also applicable to other provenance types, as they share the same data structure.

Data provenance graphs are directed acyclic graphs (DAG), which may lead to the conclusion that much of the graph visualization literature [HMM00, BBDW14] might be relevant. Two characteristics, however, make data provenance graphs special: (1) they include a hierarchy by design, and (2) they have an inherent temporal aspect. We first summarize how node-link diagrams and matrix representations—the fundamental graph visualization techniques—are suitable for addressing the tasks introduced above. We then discuss different graph aggregation strategies and techniques suitable for visualizing dynamic graphs. We end our review with an overview of the state-of-the-art in provenance graph visualization.

2.3.1. Graph Representation

The two fundamental techniques for visualizing graphs are node-link diagrams and matrix representations. Which technique works better depends on graph type, graph size, and user tasks. Visualizing a graph as a matrix is suitable for attribute-based tasks performed on weighted edges, where each edge has an associated value [SS06]. However, for path-related tasks, such as following a path to address causality tasks (Task C), node-link diagrams are more effective [LPP+06]. Therefore, node-link representations are better suited to the characteristics of data provenance graphs and suited to fulfilling the user tasks defined in Section 1.3.

2.3.2. Graph Aggregation Strategies

The scalability of node-link diagrams is a well studied area of research with a large body of existing work. To improve the scalability for visualizing provenance graphs, we can utilize the hierarchy for aggregation [NJ04, EF10]—allowing users to explore the graph using drill-down and roll-up operations. In combination with semantic zooming [PF93], the information shown can be adjusted to various levels of detail while avoiding visual clutter. Further visualization techniques for group structures in graphs, as they appear through aggregation, have recently been surveyed [VBW15]. An alternative aggregation method is motif discovery and compression [MSOI+02, AS06], which reduces the visual complexity through topology-based aggregation while preserving the basic structure of the graph [MRSS+13]. Instead of introducing new aggregations, we decompose large graphs into multiple hierarchies in *CloudGazer* (see Chapter 4) and visualize them separately. Inlays shown on demand reintroduce the lost relationships between the hierarchies.

A different approach is to distort the visual space, as typically done in lens-based approaches [TAHS06]. Furnas [Fur86] visualized the nodes with different levels of detail, determining the DOI based on the selected node. The *DOITree* approach by Heer et al. [HC04] applies a multi-focal version of this DOI function to visualize tree structures more effectively. Van Ham and Perer [vHP09] extended the DOI approach to expand parts of a large static graph showing the context, preserving the overall graph structure around a selected node. Abello et al. [AHSS14] presented a DOI function that is divided into multiple components to investigate large dynamic networks. Vehlow et al. [VKB+15] used a combination of continuous and/or discrete DOI functions to filter dense biological networks and subsequently compared the extracted subnetworks. In Chapter 5 we propose *AVOCADO*, a combination of hierarchical and motif-based aggregation with a user-driven DOI to increase the scalability for exploration of large data provenance graphs.

2.3.3. Dynamic Graph Visualization

In visualization, temporal aspects of data are particularly challenging because of the unique characteristics of time, such as the presence of hierarchical levels of granularity with irregular divisions, the occurrence of cyclic patterns, or the fact that time cannot be perceived by humans directly [AMST11]. Researchers have studied the temporal aspect also in the context of graphs [KKC14, HSS15]. When dealing with data provenance graphs, the user wants to investigate the differences between two or more analyses executed at different time points. These differences can be visualized by mapping either time to time (*animation*) or time to position (*juxtaposition* and *superimposition*) [BBDW14]. Archambault [Arc09] used superimposition of different snapshots in combination with hierarchical aggregation of adjacent nodes and path-preserving coarsening. Similarly, a recent approach by van den Elzen et al. [vdEHBvW15] allows users to explore the evolution of networks by re-

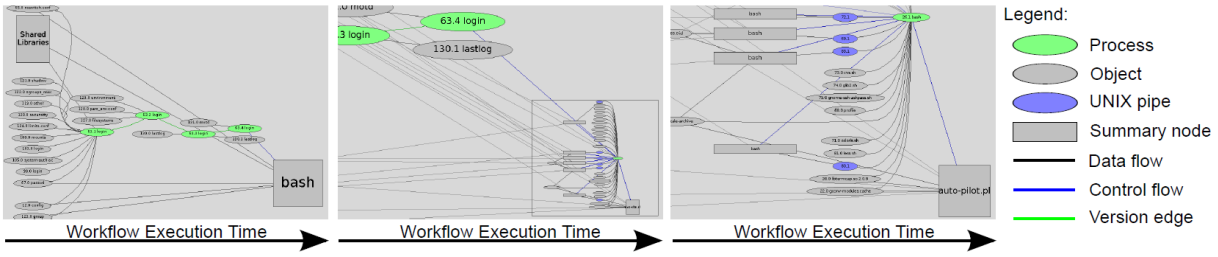


Figure 2.3.: *Provenance Map Orbiter* [MS11] uses aggregation techniques to compress the provenance graph. The graph layout, however, adapts poorly to dynamic aggregation, giving rise to additional edge crossings. This Figure is taken from [MS11].

ducing snapshots of the dynamic graph to points, forming a separate derived graph as an abstraction layer. These approaches visualize differences between large graphs very well. However, they are not suitable for the data provenance graph problem, as they do not support path-related causality tasks (Task C).

2.3.4. Provenance Graph Visualization

In recent years, workflow and provenance management systems (e.g., *VisTrails* [BCS+05]) have become more effective at capturing and storing provenance information. However, these systems provide no or only a basic visual representation of this information. For example, *Synapse* [OEY+13] tracks data provenance information to ensure reproducibility in cancer genomics and other biomedical research and visualizes the data provenance graph as a node-link diagram. However, due to heavy use of labels and icons, lack of visual glyph encoding, and missing aggregation techniques, this approach does not scale to large graphs. In contrast, *Provenance Map Orbiter* [MS11] (see Figure 2.3) uses aggregation techniques that compress the provenance graph to a high-level overview. Additionally, it uses semantic zoom and supports drill-down to show details on demand. The graph layout, however, adapts poorly to dynamic aggregation, giving rise to additional edge crossings, which hampers Task C.

Overall, none of the solutions discussed is able to address all of the tasks formulated in Section 1.3. The challenge is therefore to design an effective combination of existing techniques and strategies for visualizing large data provenance graphs with hundreds of nodes.

2.4. Information Retrieval

Information retrieval (IR) is an alternative to Focus+Context visualizations with graph aggregation and drill-down to show details on demand (see Section 2.3) that scales to large provenance graphs. Instead of selecting focus points manually and drill-down to investigate the details, users can express their interest in form of a search query, which is then used to retrieve matching nodes.

IR can be divided into two main parts: (1) building the search index from the input data, and (2) the retrieval process, which is targeted at finding items based on a search query. Depending on the data structure and the number of items, a broad variety of search capabilities and visual interfaces are available.

2.4.1. Index and Provenance

During the indexing phase, the IR system identifies and extracts features from the input data, and stores them for later retrieval. Features are, for instance, representative keywords describing a text document or tags and labels describing the content of an image or a video. Features of visualizations can be extracted in many ways, e.g., based on data metrics, image metrics, and parameters of the visualization pipeline. In general, indexing becomes easier and achieves better results for highly structured input data. Visualization grammars, as proposed by Wilkinson [Wil05] or systems such as *Vega* [SWH14, SRHH16]/ *Vega Lite* [SMWH17], or *Polaris/Tableau* [STH02a, STH02b, PCJ07], describe the visualization in a standardized and structured way. Hence, they are well suited to index and retrieval of visualizations. However, these grammars can represent only a snapshot of a visualization and offer limited interactions during the visual analysis, which form a valuable knowledge base for future exploration and recall [SvW08]. Storing changes and interactions over time creates an interaction history [KNS04, GZ09]. Capturing changes on additional levels results in various types of provenance information [RESC15] (e.g., data, visualization, and interaction provenance) that can also be used for retrieval purposes.

Provenance tracking, storage, and retrieval has also been an active research topic in the database community. Recent surveys by Herschel et al. [HDBL17] and Pérez et al. [PRSA18] summarize the state-of-the-art. Nevertheless, most of the works focus on efficient algorithms to store and retrieve information, while the visualization of and interaction with the search results play a tangential role.

Khan et al. [KKW+16] propose a framework that records users search activities in an enterprise file repository. Users can explore the evolving search provenance graph by setting a time range and exploring the collection of matching files in a detail view. Their approach demonstrates how provenance can be used to improve search queries. However,

the approach is not directly applicable to visualization retrieval, since files contain large amounts of unstructured text that require building and maintaining ontologies—which do not yet exist for visualization. Further, in *KnowledgePearls* we focus on utilizing the inherently incremental characteristics of visualization states.

Another exception in provenance retrieval is the well-studied field of data provenance graphs from biomedical workflows (see Chapter 5). Due to workflow modifications or multiple executions with different parameters or input datasets, this type of provenance graph can rapidly grow very complex. Index and retrieval of workflow provenance have been discussed in work by Frew et al. [FMS08] and Biton et al. [BCBDH08]. Workflow provenance graphs, however, are very different from visual exploration provenance graphs. Workflows are pipelines of tools and files forming a graph structure. While in workflow provenance graphs multiple tools or input files are modified between two pipeline executions, the properties of visualization states are changing incrementally.

2.4.2. Retrieval

Once the input data is available in the search index, users can start formulating search queries and sending them to the IR system for comparison with the index. We identified three strategies for formulating a search query:

1. **Query by Definition.** Queries can be created by selecting facets or classifications, by formulating a statement in system language, or via natural language.

2. **Query by Example.** Queries are implicitly defined by the user by selecting or creating an example. The query is then derived from the example.

3. **Query by Perception.** Users have a mental model of the query and interactively explore the data for possible results. Hence, the user knows which parts of the data and visualizations are of interest and how to proceed with the analysis.

Query by Definition

Writing a search query at the system level can be difficult for users. The query languages strongly depend on the underlying storage format (e.g., *XQuery*, *Prolog*, *SQL*, and *SPARQL*) and offer great versatility and flexibility [FSKS08, FMS08]. Furthermore, the syntax of these query languages is hard to memorize and tends to produce long statements. Easier access can be achieved by using custom perspectives (similar to database views) on the provenance data [BCBDH08] or carefully designed query builder and search interfaces that abstract the complexity of a query language.

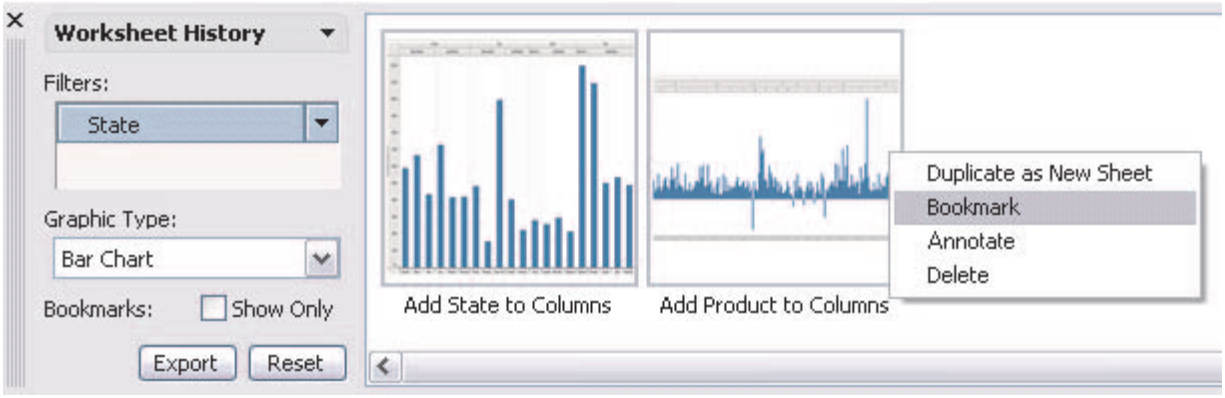


Figure 2.4.: Heer et. al. [HMSA08] provide a Boolean search interface for the provenance graph ("worksheet history"). This Figure is taken from [HMSA08].

A common interface for filtering a large collection of items are facets, which are generated from an object's metadata or text extraction and can form a hierarchy or a list of classifications. Users can select one child element of the hierarchy at a time and thus narrow down the scope (i.e., limit the number of search results) [SCM+06, CDF09]. With this approach, however, only one item can be selected at a time, and including other parts from a different sub-tree is impossible. In contrast, using facet classification, users can select multiple classes and thus construct a binary search query [Ahl96, SWA92]. A limitation is that numerical values must be binned to derive categories. Further, objects are considered only if they match exactly. Another example of a Boolean search interface was presented by Heer et. al. [HMSA08] for *Polaris/Tableau* [STH02a]. The selected chart types and data fields are stored as states in the provenance graph ("worksheet history") and can be employed by the user for filter operations (see Figure 2.4). These approaches, however, allow users to search for data fields only and the results are restricted to exact matches. In *KnowledgePearls* we provide related search terms that enable users to narrow down the scope, provide a fuzzy search for properties, and let users weight the search terms based on their interests [GLG+13, PSTW+17].

With recent advances in natural language processing, formulating queries in natural language is becoming increasingly popular. Users can speak or write queries using their native language to interact with the visualization [SS17] or search in datasets, such as integrated in *IBM Watson Analytics*[3]. The queries, however, often need to follow a predefined structure or contain commands that can be translated into system language.

[3] https://www.ibm.com/watson-analytics/

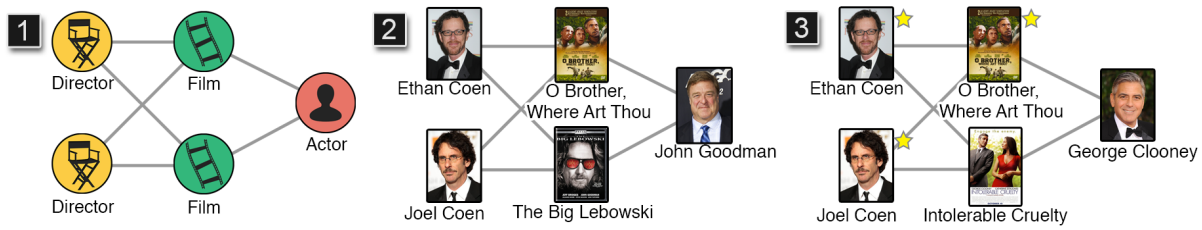


Figure 2.5.: In *Visage* [PHT+17] users can build broad queries using different node types (1). The first matching result (2) can be further refined by defining fixed nodes (3). This Figure is taken from [PHT+17].

Query by Example

In contrast to writing the definition of a query, users can create or re-use existing parts of an entity as search query. The eponymous high-level database management language *Query-by-Example* by Zloof [Zlo77] is one of the earliest approaches that allows users to query, update, and control the database with little knowledge of the query language. Numerous example-based language have emerged since then [OW93].

A notable example of provenance retrieval in the visualization domain is *VisTrails* [BCS+05], which encourages users to re-use existing workflows [SVK+08]. In order to find these workflows, users can interactively build parts of a workflow as search query and thus avoiding having to learn a new query language. Matching workflow versions are displayed along with the highlighted part. Similarly, users can interactively construct visual graph queries in *Visage* [PHT+17], as shown in Figure 2.5. Search results must match the structure of query graph and additional attributes attached to the nodes. Both approaches work well for a relatively small provenance graph and a limited number of search results. However, with a larger number of matching results, the presentation becomes cluttered.

Query by Perception

Both query by definition and query by example result in a query that serves as the input for retrieval. In contrast, query by perception relies on the user's visual system to identify contextually relevant states in a visualization of the provenance graph that encodes its properties. Examples are *AVOCADO* (see Chapter 5) and the taxonomy-based glyph design [MRSS+12], which both visualize workflows of biological experiments. *InProv* [BYB+13] utilizes time-based hierarchical grouping for filesystem provenance and lets users explore file changes in an interactive radial-based tree layout.

Shrinivasan and van Wijk [SvW08] propose a technique to capture visualization states and store them in an interaction history (i.e., provenance graph). Users can annotate and revisit states.

In the work on *CLUE*, Gratzl et al. [GLG+16] extend this approach by allowing users to assemble states of interest into a story that can then be used for presentation and recall.

As the explicit visualization of a large and quickly growing provenance graph is a limiting factor in terms of scalability, we decided to focus on a query by definition and query by example strategy in *KnowledgePearls*.

3 | ThermalPlot Attribute-based Focus+Context for Time-Series Data

Contents

Multi-attribute time-series data plays a vital role in many different domains, such as economics, sensor networks, and biology. An important task when making sense of such data is to provide users with an overview to identify items that show an interesting development over time, including both absolute and relative changes in multiple attributes simultaneously. However, this is not well supported by existing visualization techniques. To address this issue, we present *ThermalPlot*, a visualization technique that summarizes combinations of multiple attributes over time using an item's position, the most salient visual variable. More precisely, the x-position in the *ThermalPlot* is based on a user-defined DOI function that combines multiple attributes over time. The y-position is determined by the relative change in the DOI value (ΔDOI) within a user-specified time window. Animating this mapping via a moving time window gives rise to circular movements of items over time—as in thermal systems. To help the user to identify important items that match user-defined temporal patterns and to increase the technique's scalability, we adapt the level of detail of the items' representation based on the DOI value. Furthermore, we present an interactive exploration environment for multi-attribute time-series data that ties together a carefully chosen set of visualizations, designed to support analysts in interacting with the *ThermalPlot* technique. We demonstrate the effectiveness of our technique by means of two usage scenarios that address the visual analysis of economic development data and of stock market data.

3.1. Introduction

Understanding temporal developments of multi-attribute data is an essential task in many domains, such as economics, sensor networks, biology, or data journalism. Gaining new insights from such data can be challenging—even for a single multi-attribute item. However, the complexity increases significantly when scenarios comprise a collection of items, where each item comes with a set of multiple attributes that change over time. An important task when making sense of such data is to provide users with an overview for identifying items that show an interesting temporal development, including both absolute and relative changes of multiple attributes simultaneously. Two of the main challenges in this context are (I) integrating multiple heterogeneous attributes from a collection of items and make them comparable, and (II) showing multiple levels of temporal dynamics. Although a wide array of visualization techniques have been proposed for addressing both challenges, they often scale poorly to multiple attributes, a large set of items, or long time-series [AMST11]. In contrast to visual approaches, automated alternatives for summarizing multi-attribute data, such as Principle Component Analysis (PCA) and Multidimensional Scaling (MDS), often do not produce projections that can be interpreted intuitively. To the best of our knowledge, no single approach exists that effectively handles both challenge (I) and (II).

In this chapter, we aim to fill this gap by presenting the *ThermalPlot* visualization technique as our primary contribution. *ThermalPlot* provides an overview of a collection of items, allowing analysts to quickly identify items that show an interesting development over time. The technique encodes time-dependent changes in attributes into an item's position, which is known to be the strongest visual variable for encoding quantitative data [Mac86]. Changes can be derived effectively from the position in order to detect outliers, trends, and patterns. The position is based on a modular DOI function which combines multiple attributes with adjustable weight. As a secondary contribution, we introduce an interactive exploration environment for multi-attribute time-series data that integrates a series of support views which enrich interaction with the *ThermalPlot* technique.

We introduce the *ThermalPlot* technique and its implementation using a publicly available data set from the *Organization for Economic Co-operation and Development* (OECD)[1]. The data set contains an extensive collection of attributes for all OECD member countries. To illustrate our technique, we chose *long-term*, and *short-term interest rates* collected on a monthly basis between January 2000 and July 2015. We show how Latvia managed to tackle the financial crisis in 2009 and how it developed afterwards. In addition, we demonstrate scalability and effectiveness in a stock data use case where a private investor makes informed stock investment decisions using the *ThermalPlot* system.

[1] `https://stats.oecd.org/`, data set downloaded on 2015-08-21.

3.2. User Tasks

In visual data analysis, users frequently face open-ended, ill-defined tasks such as "find or detect something interesting". Particularly when dealing with multi-attribute data over time, such discovery tasks can become very cumbersome, as many attributes must be taken into account. To bring some clarity to such fuzzy analytical objectives, we identified five user tasks that must be supported and serve as a set of design requirements to be met.

T1: Monitor the development of multiple items in a certain time window. The user wants to get an overview of multiple items and monitor them simultaneously over a certain period of time.

T2: Select attributes and define their interestingness. Since not all data item attributes are equally interesting or important in specific use cases, the user needs to select one or multiple attributes and define how interesting each attribute is in relation to the other (DOI).

T3: Detect items that are (most) interesting. According to T2, the user wants to detect items that best fulfill the defined interestingness metric. Such items need to be made visually salient.

T4: Understand why the items are considered to be interesting. After discovering a set of interesting items, users need to understand why the system considered a specific item to be interesting.

T5: Monitor the development of a single item. Finally, it is not only important to know which attributes contributed to the interestingness of an item, but also to be able to analyze them in detail. This involves the need to investigate and compare the development of multiple attributes over time.

3.3. ThermalPlot Visualization Concept

The fundamental idea underlying *ThermalPlot* is simple but effective: we map a user-specified DOI value on the x-axis and the change in the DOI value (ΔDOI) on the y-axis, as illustrated in Figure 3.1. Plotting an attribute and its first derivative is not new (see for instance Figure 2 in [LZ10] that plots the *stock price* vs. *price change*). However, instead of making static plots for single attributes, we create interactive visualizations that summarize the temporal development of multiple attributes (addressing task T1).

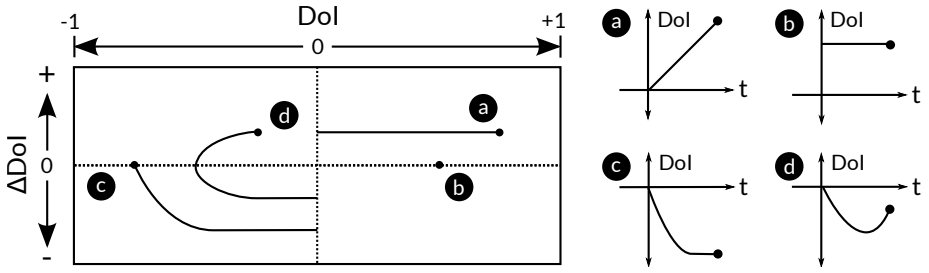


Figure 3.1.: The DOI value is mapped to the x-axis and the ΔDOI to the y-axis. DOI values that change over time result in distinctive positions and trajectories of items in the *ThermalPlot* space. The DOI values of the examples are (a) linearly increasing, (b) constant, (c) decreasing and then constant, and (d) decreasing first and then increasing.

In the *ThermalPlot* the DOI value is a weighted combination of one or multiple attributes over time, as explained in detail in the following section. Figure 3.2 shows a *ThermalPlot* visualization of the previously mentioned OECD data set. Depending on the usage scenario, the analyst defines a DOI function that results in high DOI values for one or multiple items of interest (addressing task T3). The ΔDOI is determined by the DOI change between the start (t_s) and end (t_e) of a user-defined time window (see Figure 3.2(b)). The items are then placed in the plot according to their DOI and ΔDOI values. This mapping results in a visualization where users can derive an aggregated summary of the items' developments over time from their positions in the plot. Latvia, for instance, was hit hard by the financial crisis in 2009, but showed a positive development in the *short-term interest rate* in the following years (compared to 2006). Thus, it is positioned in the upper left area (see Figure 3.2(a)). In contrast, Greece has a constant positive *short-term interest rate*, but is located in the lower left area because of the ongoing negative development of the higher weighted *long-term interest rate* attribute.

Another consequence of this mapping are the distinctive movements of items within the plot over time that are reminiscent of a thermal system. Items whose DOI increases move from left to right (see Figure 3.1(a)), while items with decreasing DOI move from right to left (see Figure 3.1(c)). The faster the DOI changes, the higher the items will rise. Consequently, negative changes in the DOI values result in downward movements of items. Mapping the ΔDOI values to the y-axis naturally separates items with a positive trend from those with a negative trend over the chosen time window. Together, this behavior results in circular movements of items. The magnitude of changes in the multi-attribute data determines the size of the circular patterns, resulting in macro and micro movements. Figure 3.1 illustrates four example movements through the *ThermalPlot* space, together with the corresponding development of the DOI value over time. Note that this does not necessarily mean that the items are constantly changing their positions in the plot, as the position is only updated when new data arrives in live streaming scenarios. Depending on the frequency of the updates, this might only happen monthly, as in our OECD use case, or daily, as in the stock market use case.

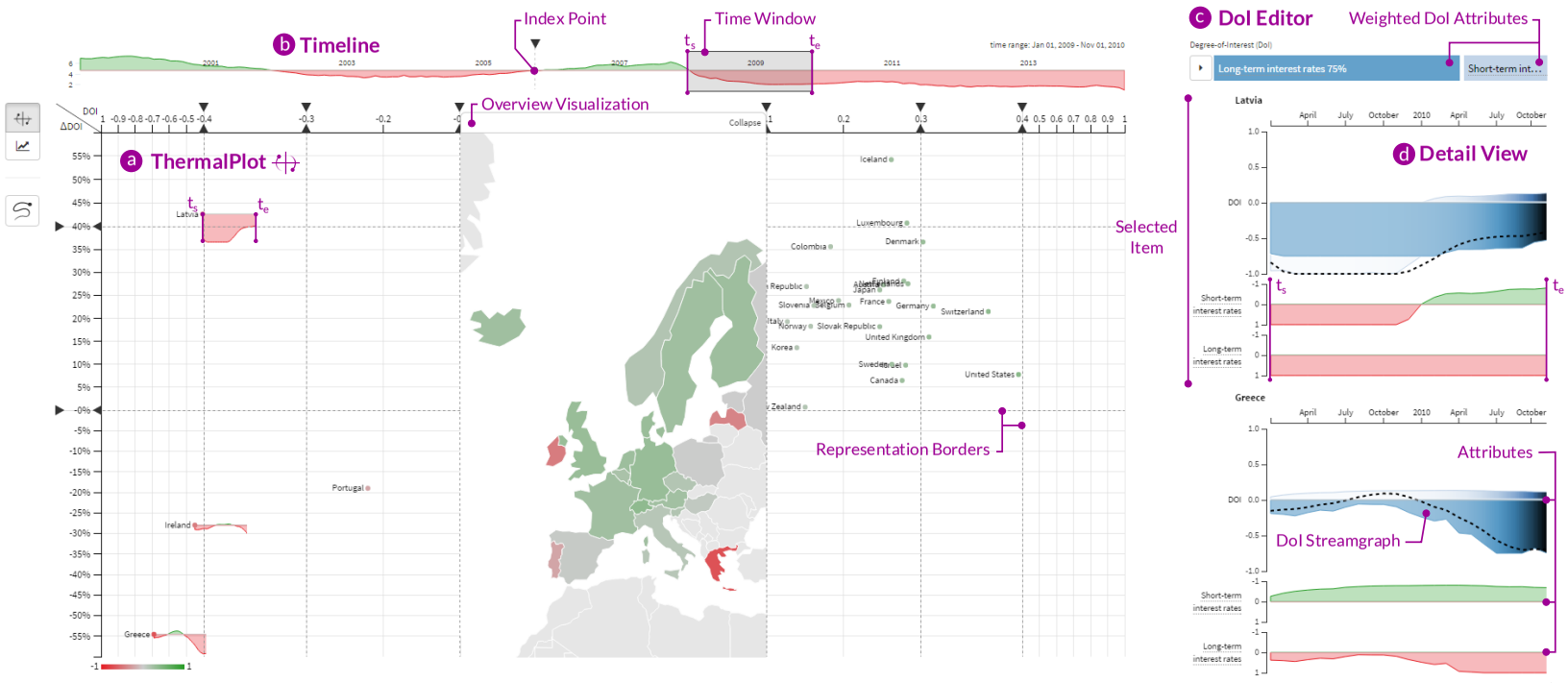


Figure 3.2.: The *ThermalPlot* technique (a) integrated in the exploration environment for multi-attribute time-series data showing the development of countries within a user-specified time window (b). Item positions in the plot are based on the selected index point and the weighted DOI attributes configured via the DOI editor (c). The detail view (d) shows the development of the composed DOI over time as streamgraphs and line charts of single attributes for selected items. Latvia, for instance, was hit hard by the financial crisis in 2009, but showed a positive development in the *short-term interest rate* in the following years (compared to 2006). Consequently, it appears in the upper left area. In contrast, Greece appears in the lower left area, even though the *short-term interest rate* attribute shows a constant positive development. However, it is surpassed by the higher weighted *long-term interest rate* attribute, which results in an overall negative development of the country.

The *ThermalPlot* concept is particularly powerful in two basic scenarios: (1) showing a static snapshot that summarizes the temporal development of items in a given time window. (2) presenting the current status of a live streaming data set, where item positions are slowly updated when new data comes in.

3.3.1. Modular Degree of Interest

Using DOI functions to adapt the visual representation is a well-known approach, and has been applied in many different ways and contexts, for example, to explore trees [Fur86], temporal data [CDRC08], large static graphs [vHP09], dynamic graphs [AHSS13], and to the visual analysis of small interconnected biological networks [LPK+13]. Intuitively, the value that results from a DOI function should reflect how interesting a particular item is to the analyst. In the *ThermalPlot* an item's position at a specific time point directly corresponds to its DOI value and therefore represents its interestingness. Before we discuss the additional roles of the DOI function in the *ThermalPlot* besides positioning the items, we introduce the method by which DOI values can be calculated for time-series data.

In his fundamental work on generalized fisheye views [Fur86], Furnas introduced the concept of DOI, where the function for calculating the DOI can be driven by various attributes. Which attributes define the importance of an item depends on the data set and task. The attributes that contribute to the DOI can be either static, such as the *founding year of the state*, or dynamically changing over time, such as the *interest rate* or the *gross domestic product (GDP)* of a country. *ThermalPlot* supports multi-attribute DOI functions, where for each time point individual attribute values are combined using a weighted sum, resulting in a single raw DOI value ($DoI_{raw}(t)$) (addressing task T2).

$$DoI_{raw}(t) = \sum_{i=1}^{n} w_i \times v_i(t) \mid \sum_{i=1}^{n} w_i = 1.$$

The weights applied can be defined freely by the user where n is the number of attributes that contribute to the raw DOI, w are the weights of the components, which sum up to one, and $v_i(t)$ is the attributes' value at time point t.

To incorporate trends and temporal effects at a specific time point t, we apply an exponential smoothing strategy [Gar06]. Exponential smoothing aggregates multiple previous time points by assigning decreasing weights over time. We use the *Holt-Winters exponential smoothing* method (also termed second-order exponential smoothing) [Win60], which is known to work well with data that contains trends or seasonal patterns.

$$\begin{aligned} DoI(t) = \alpha \times DoI_{raw}(t) + (1-\alpha) \times \\ (DoI_{raw}(t-1) + DoI_{trend}(t-1)) \, . \end{aligned}$$

The α $(0 \leq \alpha \leq 1)$ smoothing factor determines how many of the previous time steps influence the current value. An α value of 1 disables smoothing, while a value close to zero results in a strongly smoothed value. $DoI_{trend}(t)$ is an estimator for the trend of the time series, where β $(0 \leq \beta \leq 1)$ is an additional smoothing factor for the trend similarly:

$$DoI_{trend}(t) = \beta \times (DoI(t) - DoI(t-1)) + (1-\beta) \times DoI_{trend}(t-1).$$

Both parameters are configurable by the user. By default, we use $\alpha = 0.4$, $\beta = 0.4$ in the OECD use case.

An inherent problem of double exponential smoothing is finding an initial value for $DoI(0)$ and $DoI_{trend}(0)$. Depending on α and β, the influence of one specific time point $t-k$ on the current time point t can be computed. Thresholding the influence leads to a k which can be used as the starting point for computing the current time point t by setting $DoI(t-k) = DoI_{raw}(t-k)$ and $DoI_{trend}(t-k) = 0$. In our OECD scenario, for instance, we used $k = 12$, which corresponds to one year, as the granularity of k is defined in months. This results in a weight of time point $t-k$ of just 0.00087 at the current time point t when using an alpha value of 0.4.

For calculating the ΔDOI, the user needs to specify the time window (ranging from t_s to t_e with $\Delta t = t_e - t_s$). The ΔDOI is then defined as:

$$\Delta DoI(t) = DoI(t) - DoI(t - \Delta t).$$

In the case of $t = t_e$, this leads to $\Delta DoI(t_e) = DoI(t_e) - DoI(t_s)$, i.e., the DOI's change within the selected time window.

Normalization

To obtain correct DOI values that result in valid positions of items in the *ThermalPlot* space, it is essential that the values of a specific item's attribute that contributes to the DOI can be directly compared across items in a meaningful way (e.g., the *population of a country*). In contrast, *interest rates* of a country are determined by many factors and their absolute values cannot be meaningfully compared. We address this problem by letting the user define an index time point t_{index} that serves as a reference point [Ber10]. For the OECD data set, this reference point could be, for instance, a year before the financial crisis. Taking the change relative to the index point rather than absolute values enables comparison of the DOI values across items.

$$v_{rel}(t) = \frac{v(t) - v(t_{index})}{v(t_{index})}.$$

A related issue is that values across various attributes need to be in the same range before they can be incorporated into a combined DOI value. To address this problem, we allow users to specify a min/max value for each attribute, which we then use to normalize the values across attributes.

DOI Symmetry

DOI values are commonly defined in the range $[0, 1]$, where 0 means not interesting at all and 1 highly relevant. However, in scenarios that use an index point for normalization, the DOI values can be both positive and negative. In the OECD use case, for instance, an analyst could be interested in the biggest losers of the financial crisis in 2009 (negative overall DOI) that showed an upward trend in the following year (positive ΔDOI). To support such scenarios, we define the DOI in the range $[-1,+1]$, as illustrated in Figure 3.1. In these symmetric DOI cases, 0 still means not interesting. However, we let users define whether negative, positive, or both DOI values are of interest (see Section 3.3.3 for details).

3.3.2. Clutter Reduction Strategies

As in any scatterplot representation, a high density of marks in a certain region of the plot can result in occlusion problems and visual clutter. To increase the scalability of the *ThermalPlot* technique regarding the number of items, we apply a two-fold strategy that combines semantic zooming [FB95] with optional orthogonal stretching of scales [SSTR93].

Semantic zooming

We use semantic zooming to adapt the level of detail of an item to its DOI or ΔDOI value (addressing task T3). In our approach it is possible to define **representation borders**, that cause the representation to change when crossed. The borders can be defined either statically for a specific scenario or interactively by a user. Representation borders can be defined for both axes, resulting in a grid in which the level of detail can be specified for each grid cell (see Figure 3.3). Hence, every cell can also be seen as a rectangular semantic lens [TGK+14]. For instance, if a user is interested in items that are located in the sectors on the upper right as well as those contained in the first grid column on the left in the *ThermalPlot*, she can increase the level of detail for these particular regions of the plot.

What information the various levels of detail show depends again on the usage scenario. The design space ranges from a single pixel to multivariate data glyphs [War08], and even embedded full-fledged visualizations. In our prototype we support four levels of detail, as illustrated in Figure 3.3, that are defined incrementally. This means that the representation

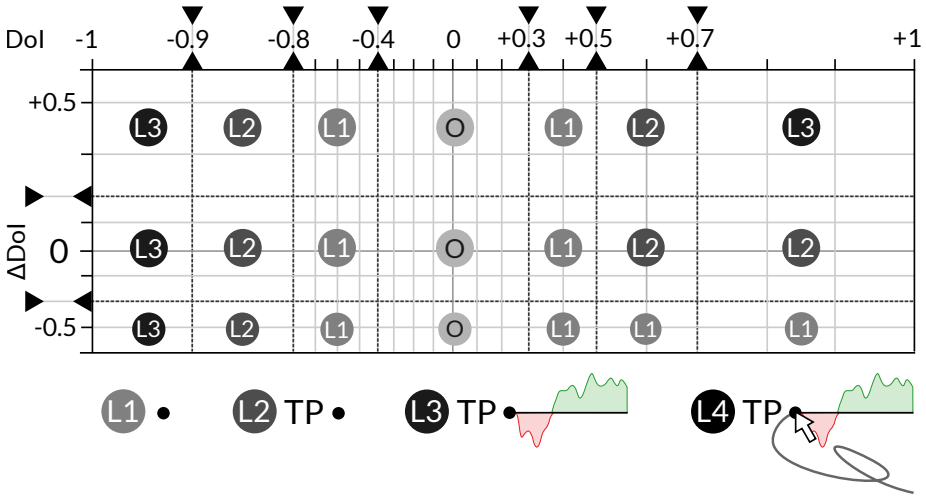


Figure 3.3.: DOI values between [-1,1] are mapped to the x-axis and ΔDOI values with a variable range to the y-axis. The visual space can be geometrically distorted by manipulating the DOI value associated with a representation border. The level of detail in each region of the plot can be configured individually. Examples of the pre-defined levels of detail L1 to L4 are shown at the bottom. Note that L4 is reserved for selections only and O indicates an embedded overview visualization (see Section 3.3.3).

of each level contains all visual elements from the previous level, plus new elements for encoding further details.

L1: Colored Mark. As a mark we chose a circle that can be colored by either a static attribute value or a temporally changing attribute that is aggregated to a single value.

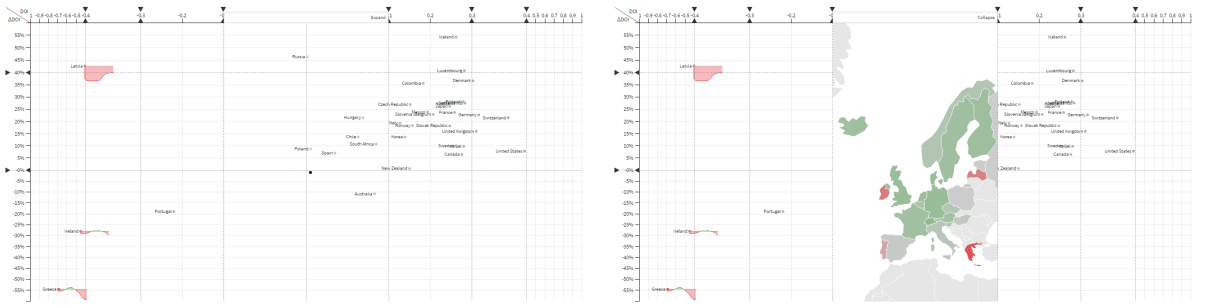
L2: L1 + Label. The L2 representation extends the mark from L1 by the item's textual label.

L3: L2 + Line Chart. In addition to the mark and label, we show a full line chart with the temporal development of a single attribute for the selected time window.

L4: L3 + Trajectory. In L4 we add a trajectory that describes the path an item takes through the *ThermalPlot* space over time.

Orthogonal Stretching

In addition to semantic zooming, we use orthogonal stretching [SSTR93] as a second measure to reduce the visual clutter. Which parts of the scale should be stretched or compressed can either be statically defined for a specific setup or dynamically set by the user. In theory, the representation borders are independent of the handles for the orthogonal



(a) Regular point-based representation

(b) Country map as embedded overview visualization

Figure 3.4.: Different options for visualizing items with a low or zero DOI value. In (a) the items scattered around the vertical center line are shown as points. (b) shows a country map embedded inside the *ThermalPlot* that gives a meaningful structure to items.

stretching. However, for the sake of simplicity in the interaction, we use the same interactive borders for controlling both. Users can move a representation border by dragging the small triangle that points away from the plot, while the triangle pointing towards the plot is the handle for distorting the space, as illustrated in Figure 3.3. Integrating a fisheye lens would also be an option. However, as the items' position plays a central role in our method, we decided to refrain from applying non-linear distortion techniques.

3.3.3. Integrated overview visualization

Items with a zero or low DOI value are by definition of no or low interest to the user. So instead of representing them by their default point-based representation, as shown in Figure 3.4a, we optionally use the space to visualize all items as marks inside an embedded overview visualization (see Figure 3.4b or Figure 3.10). The fixed layout of this visualization provides a meaningful structure that users can employ for quickly locating items. Depending on the DOI value range, the overview visualization can be either embedded in the center of *ThermalPlot* or attached on the left or right side. To flexibly tailor the *ThermalPlot* space to the task at hand, users can freely move and resize the overview representation via drag-and-drop.

What visualization technique fits best, depends on the domain problem and data. In our OECD example, the European countries are shown in map form (see Figure 3.4b). For stock data the companies can be visualized as a *Map of the Market* [Mar99] (see Figure 3.10). To provide further contextual information, static attributes associated with the items can be additionally encoded in the overview visualization. In our OECD use case, for instance, the color of each country depends on its current DOI value. By default,

we apply a diverging color scale where red is mapped to $DOI = -1$, gray to $DOI = 0$, and green to $DOI = +1$. Color-blind users can switch to suitable alternative mappings. As all items that are hidden by the overview visualization are of little or no interest, we desaturate them in the treemap, to make items with a high DOI value more salient.

As the overview contains all items with a DOI of zero or close to zero, items seamlessly enter the *ThermalPlot* space with an increased DOI value. The DOI threshold that defines up to which value items remain inside the overview visualization can be interactively specified by the user. Analysts can switch to the regular point-based representation by hiding the overview visualization (see Figure 3.4a). Animated transitions [HR07] allow users to visually follow items during the switch operation.

3.4. Interactive Exploration Environment for Multi-Attribute Time-Series Data

To put the *ThermalPlot* method to practical use, a couple of support visualizations are required. As illustrated in Figure 3.2, the overall multi-coordinated view setup consists of four linked components: (a) the *ThermalPlot* as the heart of our system, (b) the timeline, (c) the DOI editor, and (d) the detail view. The **timeline** showing the full time window for which data is available lets the user set an index point t_{index} and a time window $t_s - t_e$ that specifies the data which then serves as an input to *ThermalPlot*. The **DOI editor** is the interface for composing the DOI function by means of combining and weighting multiple attributes. Figure 3.5 illustrates the inputs and the data flow between the different components. The last component of the setup is a **detail view** that presents the development of multiple attributes over time for the current item selection.

To cover the requirements of a wide range of different usage scenarios, the defined set of views in the *ThermalPlot* setup can be extended with special-purpose views that are tailored to the domain problem. For instance, when dealing with large item collections, it can be useful to add an interface for browsing and filtering items by static attributes or weighted combinations thereof [GLG+13] (see Section 3.5).

3.4.1. Interaction with the ThermalPlot

In the prototype implementation, multiple ways of selecting items exist. Users can directly select single items in the plot by clicking on their respective representations. The items' representation in the overview visualization depends on the used visualization technique (e.g., countries on a map and blocks in a treemap), while the *ThermalPlot* always uses the point, which is available at all levels of detail. As an alternative, we provide a lasso

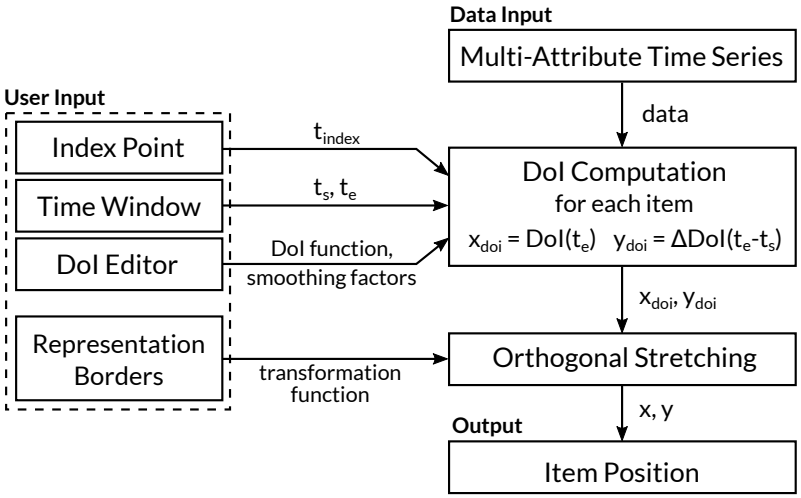


Figure 3.5.: Illustration of the data flow along with inputs and outputs. The multi-attribute time series and the user input are combined in the DOI and ΔDOI computation, which are then transformed by the representation borders. The final output is the items' x- and y-position in the *ThermalPlot* space.

selection for flexibly defining arbitrary regions of interest. Instead of completely removing non-selected items from the overview visualization and the *ThermalPlot* itself, we decrease their opacity to keep them as contextual information.

3.4.2. Timeline

The timeline serves two purposes. First, it provides the user with an overview of the time window for which the multi-attribute data is available (addressing task T1). Second, it is the interface for setting the index point (see Section 3.3) and the time window which defines the data upon which the DOI over time is calculated. Both the time window and the index point can be moved interactively using drag-and-drop. The size of the time window can also be changed dynamically. A change in one of these settings triggers an update of the *ThermalPlot* where the data is streamed from the server. During data transfer, we indicate the progress by a gradually filled up background of the time window widget in the *ThermalPlot* and the detail view shows the portion of the data that has already been transmitted.

For scenarios where the item collection can be represented by a surrogate attribute that summarizes all items, we show a line chart in the background of the timeline. To indicate that the position of the index point defines the global point of reference for the DOI calculation, we set the baseline of the line chart according to the value of the summary attribute at the corresponding time point. We then color all parts of the curve that are above the baseline in green and the parts below in red, as can be seen in Figure 3.2. For

stock market data, for instance, we plot the *Standard & Poor's 500* (*S&P 500*) index point over time. If no meaningful summary attribute is available, as in our OECD data set, we define a default attribute that is used for the timeline (e.g., the average *short-term interest rate*).

3.4.3. Detail View

The *ThermalPlot* technique summarizes effectively the development of an item collection based on the aggregated multi-attribute DOI. Additionally, it is essential that users understand what contributes to the DOI value of items over time (task T4) and are able to drill down to the actual time-series data of the different attributes that contribute to the aggregated DOI value (task T5). In the detail view, we therefore show a streamgraph [BW08] for every selected item encoding the DOI value and its weighted components over time together with the full line charts for a predefined set of attributes that are associated with the currently selected items (addressing tasks T4 and T5). Both chart types show the data within the chosen time window $t_s - t_e$. If data is missing for specific time points, we interpolate them linearly based on their nearest valid neighboring data points.

3.4.4. DOI Visualizations

We added three building blocks to the setup that are designed to allow the user to specify and understand the composed DOI function: the **DOI editor** as an interface for interactively configuring the DOI function; **DOI streamgraphs** for visualizing the contribution of each attribute for a specific item; and **trajectories** for showing the path an item takes through the *ThermalPlot* space.

DOI Editor

Users can interactively define and manipulate the multi-attribute DOI via an integrated DOI editor, as can be seen in Figure 3.6 (addressing task T2). The editor can operate at two different levels of detail. The overview, which is shown by default, presents the currently set weighting of individual components as a stacked bar. Using drag-and-drop, users can directly manipulate the weights of individual components. The weight of a component is redundantly encoded in the length of the bar and its saturation, and additionally shown in a text label. After switching to the detail mode, users can change the DOI formula by adding and removing attributes, setting their value range, and by optimizing parameters of the exponential smoothing (see Section 3.3.1). Additionally, the user can invert the meaning of an attribute, which is particularly useful when positive values have a negative meaning associated (e.g., the lower a country's *interest rate*, the better).

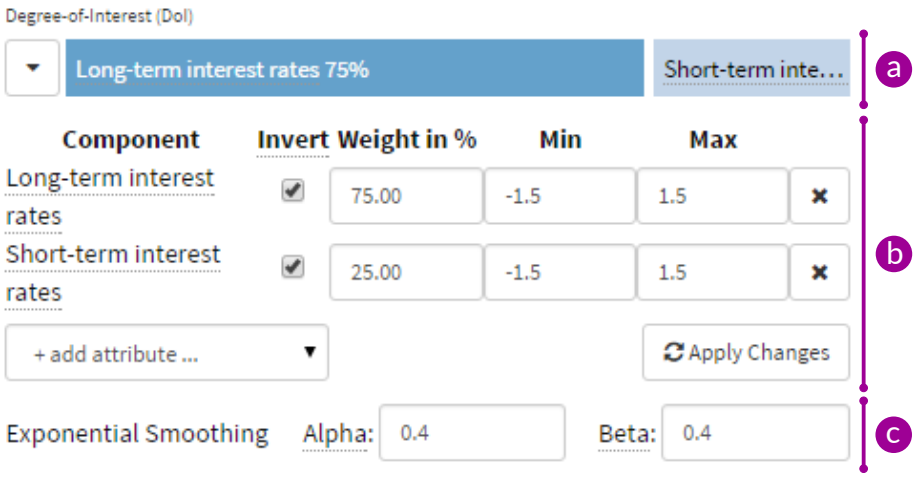


Figure 3.6.: Multi-attribute DOI editor with a stacked bar that encodes the weighting of individual components (a). Using the interface below, the user can add components, invert the semantic, and set the range for each attribute (b). Smoothing parameters are applied globally (c).

DOI Streamgraph

To support the user in understanding the effects the DOI function has on the items in the *ThermalPlot*, we add item-specific streamgraphs to the detail view (addressing task T4). The streamgraph visualizes how much each attribute contributes to the aggregated DOI value over time. Figure 3.7 shows an item's streamgraph for the DOI settings defined in Figure 3.6. In the streamgraph, every contributing attribute is represented as a stream whose height is proportional to the weighted attribute value $w_i \times v_i(t)$. See Section 3.3.1 for further details on the computation of multiple-attribute DOI values. In the example in Figure 3.7, the *short-term interest rate* attribute is always positive, while the *long-term interest rate* is mostly negative. In cases where multiple attributes contain positive and negative values, the individual streams can cross each other. The color of each stream corresponds to the color in the DOI editor bar.

Because of the layering principle, streamgraphs can by definition represent only positive values. However, to be able to encode both components with a negative and with a positive impact on the DOI, we decided to use a two-part representation: The parts above and below the baseline show the contributions of the attributes that exhibits positive and negative changes relative to the index point, respectively. Hence, the raw DOI value is the difference between the stream heights above and below the zero line.

The dotted line above the streamgraph representation indicates the smoothed DOI value for each time point. The smoothing effect is clearly visible when comparing the dotted line to the raw streamgraph values, which show small fluctuations of the attribute values.

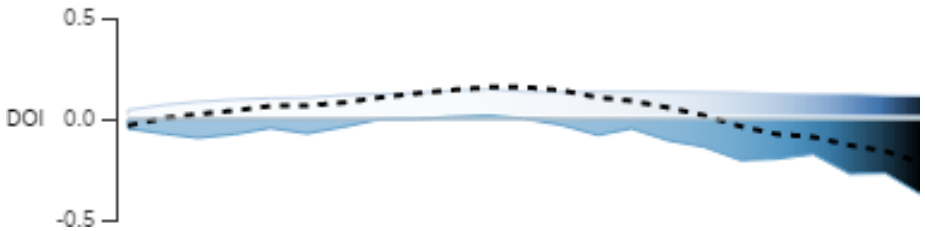


Figure 3.7.: Streamgraph visualizing the contribution of each DOI component at each time point within the time window. Streams below the zero line indicate negative contributions, while the dotted black line indicates the smoothed DOI value.

We additionally indicate how much individual time points contribute to the final smoothed DOI value by applying a horizontal color gradient: the darker the color, the higher the impact on the overall DOI.

In many real-world scenarios, missing data plays a role. An example is the closed stock market on holidays and weekends, where all associated attributes have no values for these days. Depending on the usage scenario, there are many ways of dealing with this problem. The missing values can be handled either in the data space, by applying forecasting or interpolation strategies, or in the view space, by clearly marking the missing data, for instance.

In *ThermalPlot* we apply a nearest neighbor interpolation to fill missing data values (see Section 3.4.3). However, it is essential that users are aware of the strategies applied and their consequences, as these can influence the results of multi-attribute DOI functions. To address this issue, we explicitly indicate interpolated values marked by a hatching pattern (see detail view in Figure 3.13).

Trajectories

With the highest level of detail (see **L4** in Section 3.3.2) we add trajectories to the item representation in the *ThermalPlot*. A trajectory represents the item's path within the selected time window. The example in Figure 3.8 matches the DOI streamgraph presented in Figure 3.7. The trajectories' opacity decreases over time (e.g., as in [RFF+08]), to allow correct interpretation of parts of trajectories in which the DOI remains relatively constant. In addition, the trajectories give analysts a static view of different thermal effects, such as loops and periods of rest, without the need for animation.

Although the simultaneous display of trajectories of multiple items can easily clutter the representation (see Figure 3.9), it facilitates spotting items that behave very differently from the rest. However, to keep the visual clutter at a minimum, trajectories are disabled by default and shown for items that have been actively selected only.

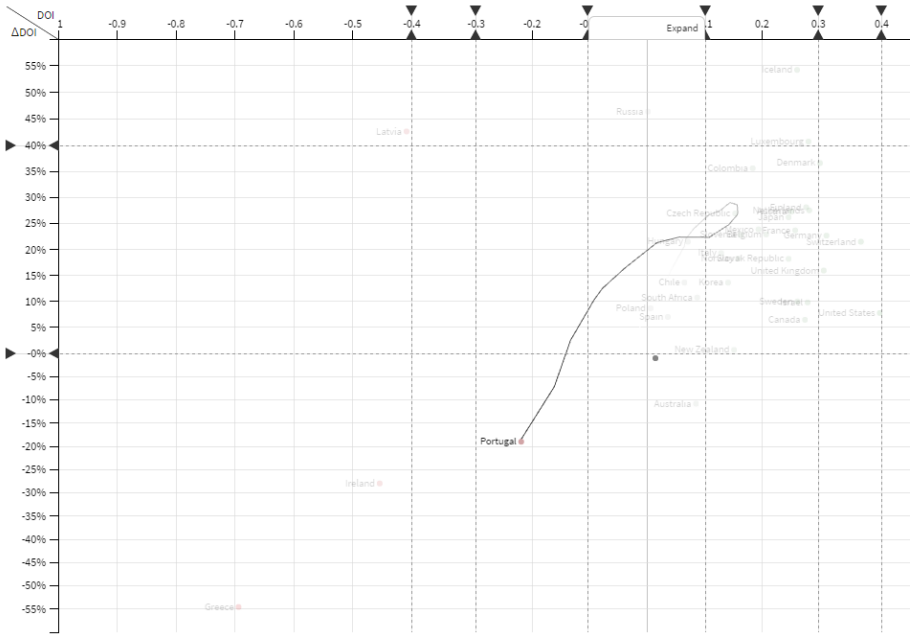


Figure 3.8.: Trajectory of a selected item in the *ThermalPlot* describing the path an item takes through the *ThermalPlot* space over time. The transparency along the trajectory encodes the item's temporal development.

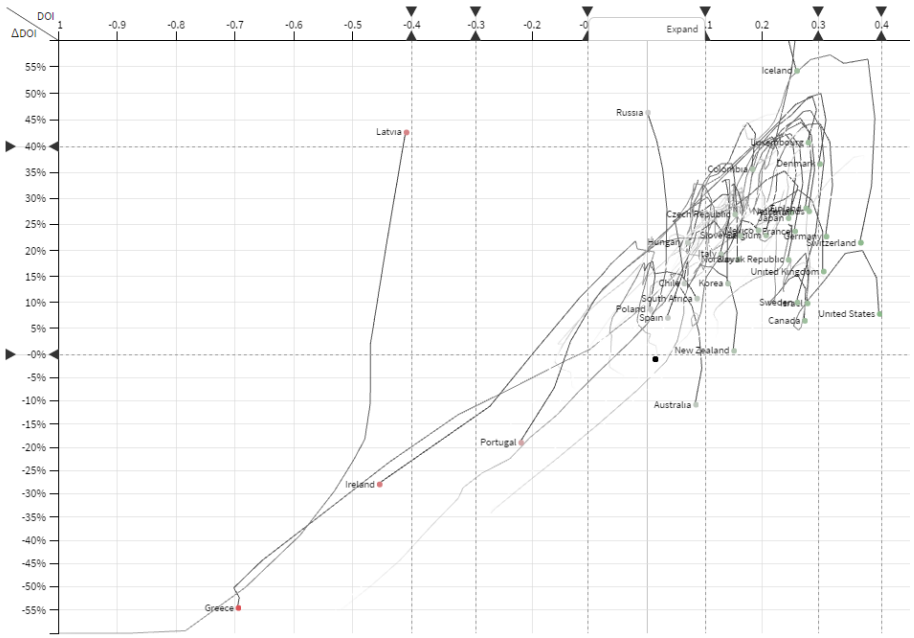


Figure 3.9.: Showing trajectories of multiple items simultaneously allows users to identify items that behave differently from the rest. However, as trajectories soon result in a cluttered representation, they are turned off by default and shown for selected items only.

3.4.5. Implementation

The *ThermalPlot* prototype has a client-server architecture. The server component is based on the recently published *CloudGazer* infrastructure (see Chapter 4), which supports streaming of time-dependent data for multiple items and attributes. By using different data connectors, we can access either live data or data already collected from a database. The web-client uses AngularJS[2] for the overall management. Visualizations are implemented in D3 [BOH11a][3]. An interactive version of the *ThermalPlot* environment and a demonstration video is available on the accompanying website[4].

3.5. Use Case

We demonstrate the value and utility of the *ThermalPlot* technique by applying it to a stock market data set that includes multiple attributes such as *trade volume*, *open/close price*, and *daily low/high* for all companies traded in the *Standard & Poor's 500* (*S&P 500*) index. We gathered the daily data via the *Yahoo Finance API*[5].

The use case is based on multiple analysis sessions with a consultant who invests parts of his private savings in stocks. Our domain expert is 32 years old and works for a company that specializes in economics and innovation policy consulting. The first phase of the analysis took place in mid-July 2015 using the *thinking aloud* method. The second phase was conducted in August 2015 using self-reporting with an analysis diary.

The expert usually checks his personal portfolio once or twice a week with the goal of making informed investment decisions. His current decision-making process is based on a combined investigation of the companies' recent developments on the stock market and quarterly published performance metrics (e.g., *dividend yield* and *earnings per share (EPS) growth*) that are available for all traded companies.

In his current workflow, he starts the analysis by investigating the static performance metrics from quarterly reports. As we have collaborated with him in a previous project, he already uses an interactive *LineUp* ranking visualization [GLG+13] that allows him to create a weighted combination of multiple static metrics to identify companies that could be underrated and thus interesting investment targets. He starts by going through the top-ranked companies and checks their recent development on the stock market by visiting

[2] `https://angularjs.org`; used Angular v1.2.6
[3] `https://d3js.org`; used D3 v3.5.6
[4] `https://thinkh.github.io/thermalplot/`
[5] `https://developer.yahoo.com/yql`

Figure 3.10.: *ThermalPlot* visualizing the recent development of all 500 companies from the S&P 500 index, according to the respective DOI computation. The treemap is used as an integrated overview visualization and shows all companies listed in the S&P 500 index. The annotations show the interpretation of our user for the different corner areas. Companies located in areas (a) and (c) are potential investment candidates.

online portals such as *Yahoo Finance*[6] or *finviz*[7]. However, collecting and matching this information from various sources is a tedious and time-consuming process.

ThermalPlot provides the expert with a big picture that summarizes the recent developments on the market (task T1). In order for *ThermalPlot* to show meaningful positions for the companies, the expert selects a time window and defines the DOI function. For the stock market data, the timeline shows the overall development of the S&P 500 index from the beginning of 2015 to the last trading day—in this case August 14, 2015. He sets the time window to include all data from the last two months of trading, which in turn triggers loading of the corresponding data from the server, as shown in Figure 3.10. To make the prices comparable across companies, he sets the index point to the beginning of 2015 (see Section 3.3.1). Consequently, all companies with a considerable positive development since the start of the year appear to the right of the treemap, whereas companies with a negative development are positioned to the left (see Figure 3.10). For the DOI computation he chooses a weighted combination of the dynamic *closing price* (75%) and

[6] `https://finance.yahoo.com`
[7] `https://finviz.com`

Figure 3.11.: Selecting the energy sector in the treemap highlights the stocks with the maximum detail level in the plot accordingly. At first glance, *TSO* and *VLO* are the obvious choice, but they might already be overrated according to our expert's own definition. Instead he investigates *WMB* that is located in his areas of interest (see Figure 3.10 (c)). According to the poor *LineUp* rank (to the right) and the loss in volume shown in the detail view (see Figure 3.13), he discards this company too.

the static attributes *EPS growth* (10%), which indicates a positive business development, *return on equity* (10%), which indicates competitiveness from the shareholders' perspective, and *dividend yield* (5%), which indicates the "shareholder friendliness" (see the upper right corner of Figure 3.11 and addressing task T2).

In the context of the stock market use case, the four corner areas have special meaning for the expert, as shown in Figure 3.10. The upper right area (b) contains strong companies with positive long- and short-term development. Consequently, this area would be the obvious investment choice, but comes with the risk of containing already overrated companies that could go down soon. Area (d) at the lower left contains companies with an ongoing negative development that would pose high-risk investment targets. Our expert is particularly interested in the remaining two areas: The lower right area (c) contains companies that developed well in the beginning of the selected time window, but show a recent negative trend. The expert assumes that this could be only a minor short-term effect for a number of stocks and that their stock prices could rise again soon. The last area, on the upper left (a), contains companies with a negative development in the beginning but a recent positive trend. This could be caused by recent good news or reports, indicating a turnaround.

As an overview visualization, a centered treemap that covers the area defined by DOI values ranging from -0.2 to 0.2 is used. Companies in the treemap are grouped by industry

Figure 3.12.: The expert selects the companies *LYB* and *ANTM* that are located in area (c) in Figure 3.10. He presumes that the stock price for companies in this area could raise again soon. He discards *ANTM*, due to the poor *LineUp* rank (> 100) and would rather invest in *LYB*, that can be found under the top 20 companies.

sector according to the *Global Industry Classification Standard* (GICS) taxonomy[8]. The blocks representing the companies in the treemap are colored according to their DOI (see Section 3.3.3).

The expert starts the analysis by looking at the distribution of companies in the different areas of *ThermalPlot*. While the upper left area (a) is empty, the lower right area (c) contains three companies that he wants to investigate further: *WMB*, *LYB*, and *ANTM*. After selecting *WMB*, he sees that the company is part of the energy sector. Consulting the treemap, the majority of companies from the energy sector have performed poorly since the start of the year. As he closely follows the news and stock reports, this confirms what he already knows—there was a lot of pressure on the energy sector in the first half of the year. To take a closer look, he selects all companies in the energy sector by clicking the label of the sector in the treemap, as shown in Figure 3.11. While the companies in the lower left area are of no interest to the investor, he looks at *WMB*, which is a clear outlier located at the upper border of the lower right area (c), and investigates its performance by inspecting the combined streamgraph (task T4) and the individual attributes in the detail view (see Figure 3.13 and addressing task T5). He recognizes an odd peak in the *volume* line chart in late June. After a quick Internet search, he is able to attribute the peak to a dividend payout announcement. After finally checking the company's rank in the LineUp visualization, where it is ranked in the lower half (> 250 of 500), he decides to look for other investment options. Still focusing on the energy sector, the two outliers in

[8] `https://www.msci.com/gics`

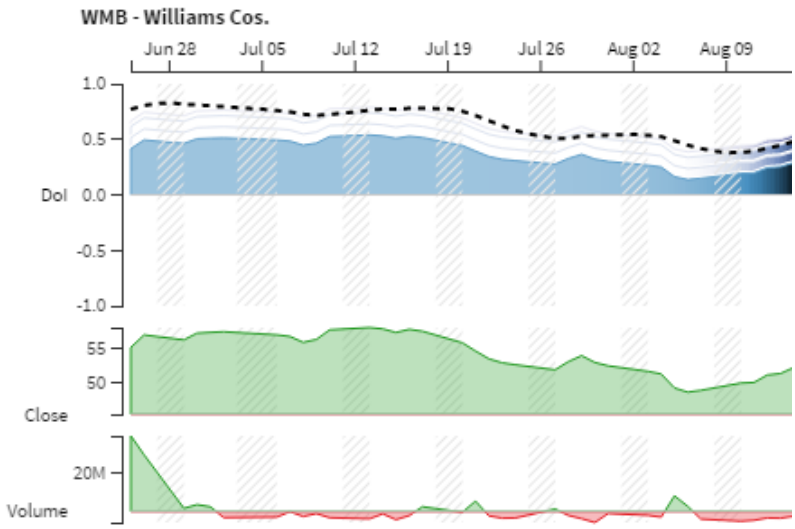


Figure 3.13.: The detail view shows the DOI streamgraph and line chart for the different attributes for the selected company *WMB*. The hatching pattern in the streamgraph and line charts encode missing values caused by weekends and holidays.

the upper right area (b) grab his attention: *TSO* and *VLO*. However, as companies in this area may already be overrated according to his own definition, he decides to not invest in the energy sector and clear the selection. He continues by checking the development of *LYB* and *ANTM*, which are the other two companies from his original selection (see Figure 3.12). A further look in *LineUp* reveals that *ANTM* does not rank highly (> 100) and does not meet his static performance metric criterion. However, *LYB* is among the top 20 companies according to the ranking and therefore seems to be a potentially lucrative investment. Using *ThermalPlot*, the expert was able to effectively identify one stock worth investing in from a large collection.

Informal User Feedback

The fact that the expert is now using *ThermalPlot* on a regular basis demonstrates that he deems the tool as a valuable addition to his current stock investment decision making process. He also mentioned that the new analysis workflow is much more elegant than his former approach and that the technique does an excellent job in summarizing the current developments on the market.

However, he also noted that, as a post-analysis follow-up step, he still needs to check external sources and recent news to collect more evidence for backing up the investment decision he made based on *ThermalPlot*. Consequently, *ThermalPlot* cannot serve as a comprehensive tool for decision-making, but shows its value for monitoring the market and quickly identifying potential investment candidates. He further mentioned that it

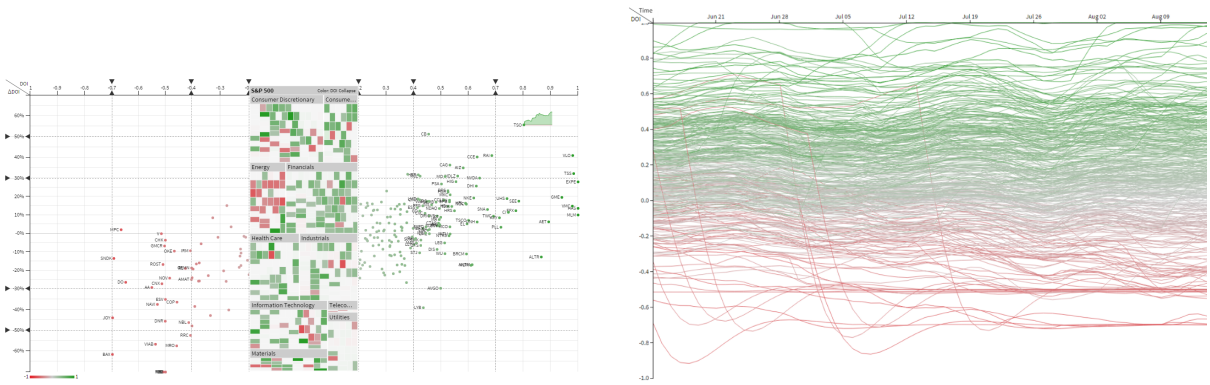
would make sense to extend the time window to include the last two years, as short-term fluctuations are not relevant to a private investor who is rather interested in long-term investments. He also noted that the trajectories are hard to interpret.

In the feedback sessions we discovered that it is difficult for the user to configure the DOI and understand the implications of the weighting and the exponential smoothing. To address these issues, an advanced editor for constructing the DOI function and further techniques for combining multi-attribute time-series data must be developed.

3.6. Discussion

Scalability One of the most critical factors when it comes to judging the value of a novel visualization technique is its scalability. In this regard, the *ThermalPlot* suffers from the same occlusion problems as scatterplots. This can be particularly problematic for items that are shown in a higher level of detail, containing labels, line charts, and trajectories. We address the problem by giving users the possibility to change the level of detail combined with the option to spread and compress the visual space, as discussed in Section 3.3.2. The embedded and linked overview visualization also supports quick identification and selection of items. However, ultimately it is the user's responsibility to resolve visual clutter in densely populated areas of the plot. As future work, it seems promising to integrate strategies that automatically adapt the scales and representation borders. This way, the system could adapt to the current situation without active interaction by the user. This could be particularly valuable for live streaming scenarios, where items move through the *ThermalPlot* space, to provide users with an overview that enables them to quickly grasp the overall status of the items and how they behave over time. To reduce the visual clutter in certain regions by adapting the layout and level of detail, it would be necessary to quantify the overlap of items in the respective regions of the plot. Although this adaptive behavior has potential to significantly reduce visual clutter, it is prone to get distracting or even confusing to users.

Still, the proposed measures would not prevent the overlap of items that have the exact same position, i.e., the same DOI and ΔDOI values. A possible countermeasure is to aggregate a group of individual items to a single surrogate item. A disadvantage is that users could then assume that all aggregated items have the exact same temporal development. However, due to the multi-attribute DOI function, the contribution that attributes have to the final DOI can vary over time. The same argument holds for trajectories, which could be considerably different due to the fact that only limited previous time points are taken into account for computing the final DOI (see Section 3.3.1 on exponential smoothing). To address these issues, advanced aggregating techniques for time-series data need to be developed. The integration of adaptive behavior and aggregation techniques are both interesting topics for future research, albeit beyond the scope of this chapter.



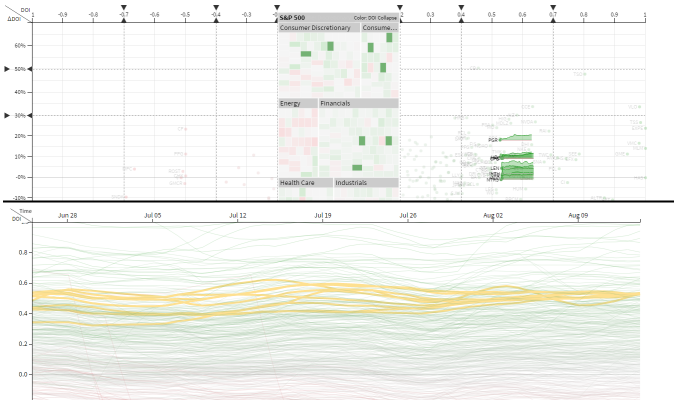
(a) *ThermalPlot* with DOI mapped to x-axis and ΔDOI to y-axis.

(b) *Line chart* with time mapped to x-axis and DOI mapped to y-axis.

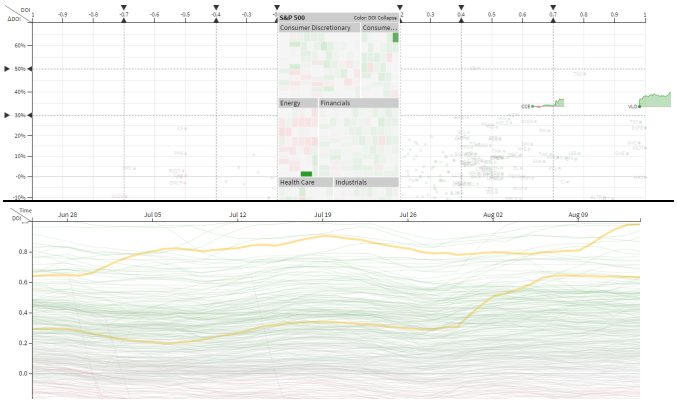
Figure 3.14.: *ThermalPlot* provides a better overview for a large number of items, in this case companies from the S&P 500 index. Assertions about the trend of a certain item in the line chart is not possible. Only negative outliers (red), caused by stock splits, can be clearly identified and found in *ThermalPlot* at DOI $= -0.5$ (bottom).

ThermalPlot vs. Line Chart One of the most common visualization techniques for time-series data is a regular line chart with time mapped to the x-axis and values (e.g., DOI values) mapped to the y-axis (see Figure 3.14b). Line charts scale to only a small number of items and a small time window for data with high variance [JME10]. Following the trend of one particular line in a large collection is only possible using interaction, i.e., by selecting an item or following a line with the cursor. In contrast, *ThermalPlot* provides an overview that scales to many items. The performance of an item can be identified by its position (see Figure 3.14a). However, this technique comes with the disadvantage that time is encoded implicitly.

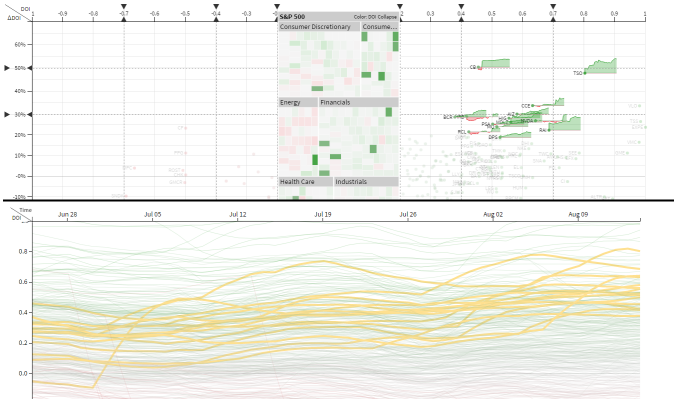
In order to select the same items in *ThermalPlot* and a line chart, the user needs to apply different interactions. Selecting all items within a specific DOI value range in the line chart corresponds to selecting a vertical slice in *ThermalPlot* (see Figure 3.15a). Selecting all items with a specific ΔDOI value requires to set an angular brush [HLD02] in a line chart, whereas in *ThermalPlot* the selection is a horizontal slice of the plot (see Figure 3.15b). Selecting items within a specific DOI and ΔDOI range requires a combination of region select and angular brush in a line chart, whereas in *ThermalPlot* it can be achieved by making a rectangular selection (see Figure 3.15c). A lasso selection in *ThermalPlot* is even more flexible, as it allows the user to select items within an arbitrary region. To achieve the same selection in a line chart, however, a more complex series of interactions is required.



(a) Selecting a **vertical slice** in *ThermalPlot* corresponds to a horizontal region selection in a line chart.



(b) Selecting a **horizontal slice** in *ThermalPlot* corresponds to an angular brush in a line chart.



(c) Selecting an **arbitrary region** in *ThermalPlot* corresponds to a combination of a region select and an angular brush in a line chart.

Figure 3.15.: Different selection shapes in *ThermalPlot* and the corresponding results in a line chart.

Trajectories Trajectories provide an easy and static way for showing an item's position for all time points within the selected time window. In addition, they allow revealing periodic patterns, such as circles or recurring peaks (see Section 3.4.4). However, these patterns can be distorted in cases in which the optional orthogonal stretching is applied. Depending on the chosen configuration, circular patterns can look elliptical, for example. In *ThermalPlot*, we make the user aware of the distortion by adding grid lines to the background of the plot.

Granularity In the visual analysis of time-dependent data, the level of granularity plays an important role [AMST11]. In our stock market scenario, for example, we work with daily *closing prices*. However, the granularity could also be increased to one value per hour, minute, or second. For automated high-frequency trading programs, the level of granularity must be even higher. Granularity is a relevant factor to *ThermalPlot* because it determines the visual patterns and trends a user will see in the *ThermalPlot*. For instance, the granularity level has a large impact on the path of the trajectories. While a low sampling rate is sufficient for seeing macro patterns (e.g., visible as large loops in the trajectories), a higher frequency is required for micro patterns. Further, the smoothing algorithm and its parametrization have a large impact on the results. The appropriate level of granularity and the smoothing approach depend on the specific analysis task.

Animation By default, the selected time window in *ThermalPlot* is fixed to the user selection. However, the streaming capabilities of the implementation (see Section 3.4.5) also enable real-time data updates. In this case, the time window is shifted automatically to include the latest time point. Newly incoming streaming data triggers the re-computation of the user-selected DOI function (see Section 3.3.1), and cause the items' positions to be updated accordingly. However, if the time window is very small, covering only a few time steps, items can change their positions rapidly—making it hard to follow the position changes. Therefore, by choosing a reasonably large time window, it's the user's responsibility to control the change behavior.

Technical Considerations Besides the visual scalability of the technique, we should also discuss briefly technical constraints comprising the performance in data loading, streaming, and caching. Naturally, the larger the user-selected time window, the more data needs to be transferred from the server to the client. When selecting a time window $[t_s, t_e]$ with $\Delta t = t_e - t_s$, the actual required data time window is $[t_s - \Delta t - k, t_e]$. The additional history is needed for computing the ΔDOI value for the start point of the selected time window t_s (see Section 3.3.1 for details on the ΔDOI computation).

We indicate the data loading progress by gradually filling up the time window representation from left to right. In addition, the opacity of the *ThermalPlot* is substantially decreased and the thermal space is overlaid with an animated progress icon. Even when data loading is in progress, the user can follow the narrative resulting from the movements of the items within the selected time window in the *ThermalPlot*.

3.7. Summary

We have presented *ThermalPlot*, a scalable visualization technique for exploring multi-attribute time-series data. We use the position—the strongest visual variable—to encode item importance according to the DOI value in the horizontal direction and according to the change in DOI value (ΔDOI) in the vertical direction. This mapping allows users to see effectively the development of attributes over time at a glance. We introduce several scalability concepts and support views, including a timeline, a DOI editor, and a detail view. We have introduced the *ThermalPlot* technique using two data sets with different scale and complexity. We evaluated the technique by means of a stock market use case and expert feedback.

In the presented exploration environment, items are part of a larger collection without relationships between them. In real-world scenarios, however, items often influence each other and therefore cannot be treated independently. In the financial market, for instance, a crash of a single company can have a negative impact on a large group of stakeholders, such as suppliers, customers, and shareholders. Other examples are the exploration of biological pathways, where cellular effects can influence reactions further downstream [LPK+13], and the monitoring of IT networks, where a problem in the infrastructure can propagate to other parts of the network. In Chapter 4 we investigate ways to conceptually integrate item relationships into the *ThermalPlot* environment.

4 | CloudGazer Topology-based Focus+Context for Large Dynamic Networks

Contents

With the rise of virtualization and cloud-based networks of various scales and degrees of complexity, new approaches to managing such infrastructures are required. In these networks, relationships among components can be of arbitrary cardinality (1:1, 1:n, n:m), making it challenging for administrators to investigate which components influence others. In this chapter we present *CloudGazer*, a scalable visualization system that allows users to monitor and optimize cloud-based networks effectively to reduce energy consumption and to increase the quality of service. *CloudGazer* is a multiple coordinated view system that visualizes either static or live status information about the components of a perspective. Instead of visualizing the overall network, we divide the graph into semantic perspectives that provide a much simpler view of the network. We reintroduce the lost inter-perspective relationships for selected parts of the focus perspective using a topology-driven DOI function. The extracted path of components is added as dynamic inlays. We demonstrate the effectiveness of *CloudGazer* in two usage scenarios: The first is based on a real-world network of our domain partners where static performance parameters are used to find an optimal design. In the second scenario we use the VAST 2013 Challenge dataset to demonstrate how the system can be employed with live streaming data.

4.1. Introduction

The availability of modern cloud computing technology has led to a surge in building more dynamic, fast growing, and continually changing systems. Cloud-based networks are built from various physical components, such as servers and storage devices, that host applications and provide resources that can be used flexibly for different purposes. To make optimal use of the hardware, applications run on virtual machines (VMs) that are, in turn, hosted on servers. However, the assignment between components is neither exclusive nor static. Multiple application instances can run on the same VM, and multiple applications of the same type can run on multiple VMs. Moreover, a physical server can host several VMs. To optimize the quality of service and minimize energy consumption, these assignments are changed regularly depending on the load of individual VMs or other circumstances in the network.

The work of cloud data center administrators comprises many different tasks, ranging from designing the network to active monitoring and optimizing the infrastructure for reduced energy consumption and a high quality of service. State-of-the-art network monitoring systems are often of limited use for these tasks, as they provide only an overview of the status of isolated components, such as CPU load, memory load, and available bandwidth. However, the crucial knowledge about how components influence each other is missing. An alternative approach is to present the overall network infrastructure as a graph. Figure 4.1 visualizes an example network in which physical components are shown in blue, virtual machines (VMs) in green, and applications in red. As can be seen, the graph can become cluttered quickly—even for small networks.

In cloud-based networks we can differentiate between two basic types of relationship: (1) **direct relationships** between components of the same type, representing physical connections or logical groupings of components (e.g., a grouping of VMs or applications by customer); and (2) **mapping relationships**, representing the assignment of one component to another (a VM running on a server). In Figure 4.2a and 4.2b, direct relationships are indicated by solid lines and mapping relationships by dashed lines.

Instead of letting users work with the overall graph that mixes both relationship types, we split the network into perspectives according to component type: physical, virtual, and application perspective, as demonstrated in Figure 4.2. The resulting perspectives are much smaller and easier to manage, and also match better the mental model of the administrators. This subdivision strategy for coping with the complexity of graphs has already been applied successfully in many different domains. The large biological pathway network, for instance, is subdivided into small semantic sub-pathways [LPK+13].

However, subdividing the network comes at the cost of losing mapping relationships, which are crucial, for instance, to avoiding side effects during optimization that result from changes in the network. For example, migrating a VM to another server can optimize

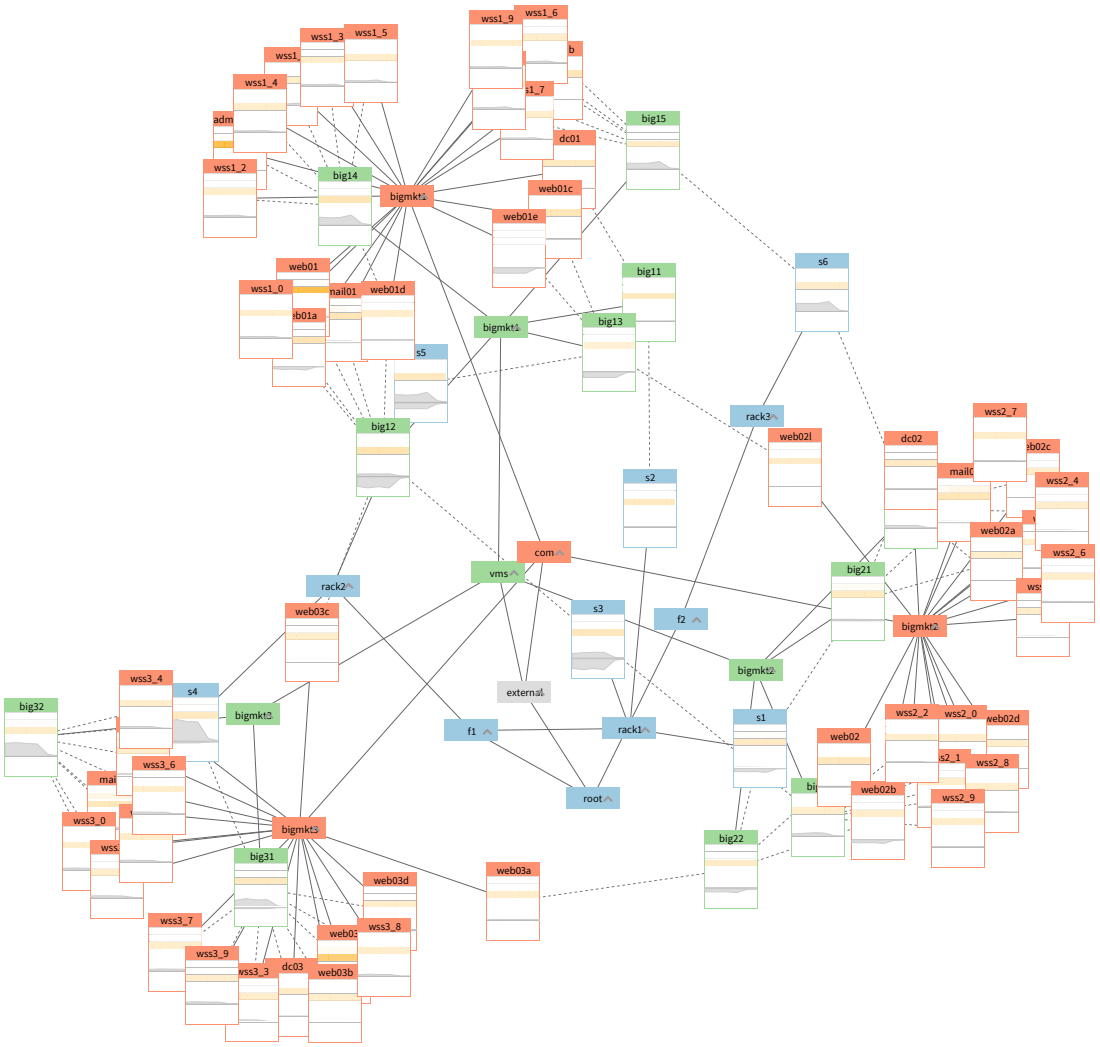
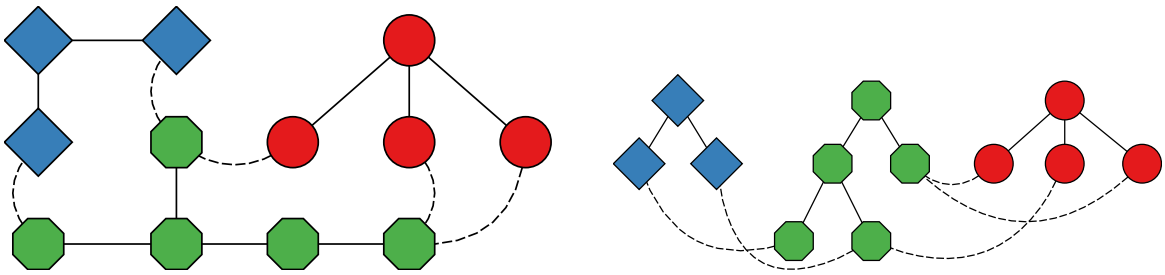


Figure 4.1.: Graph of a cloud-based network with 67 nodes. Blue, green, and red nodes encode physical components, VMs, and applications respectively. Solid links denote relationships between components of the same type, such as logical groupings of VMs and applications by customer, and dashed links indicate mapping relationships where one type of component is assigned to a component of a different type.



(a) Overall graph (b) Component-specific perspectives

Figure 4.2.: Division of the network into component-specific perspectives. Solid lines represent direct relationships between components, while dashed lines indicate mapping relationships. The graph in (a) is split into the three perspectives shown in (b).

one application's communication, but may hamper the performance of other applications hosted on the same VM.

The primary contribution of this chapter is *CloudGazer*, a visualization system for **analyzing**, **monitoring**, and **optimizing** complex distributed systems. *CloudGazer* lets users work with separate perspectives while reintroducing lost inter-perspective relationships on demand. As a secondary contribution we present the *Hierarchical Grid* layout, which further increases the scalability of our solution in terms of the number of components.

4.2. Domain Background and Goals

Modern networks comprise different types of components that all work together: physical servers, virtual machines (VMs) hosted on servers, and applications running on the VMs. This design results in a graph where relationships among components can be of arbitrary cardinality (1:1, 1:n, n:m). In the following section, we introduce different service models offered by providers, followed by a discussion of the domain goals we aim to solve.

4.2.1. Cloud Computing Stack

Before cloud computing became popular, customers were able to rent a whole physical server located in some data center. However, with improved virtualization approaches, the rise of cloud computing and platforms such as *VMWare VSphere*[1], *OpenStack*[2], and

[1] `https://www.vmware.com/products/vsphere/`
[2] `https://www.openstack.org/`

OpenNebula[3] the situation has changed fundamentally. According to the established NIST definition [MG11], cloud computing can be categorized into three service models: *Infrastructure-as-a-Service (IaaS)*, *Platform-as-a-Service (PaaS)*, and *Software-as-a-Service (SaaS)*.

IaaS providers sell VMs to their customers, who can freely install their preferred operating system, host services, and manage their own software-defined network. Using this strategy, IaaS providers are able to increase their overall data center workload by hosting multiple VMs on a single physical server. This has the advantage that customers are not directly affected by hardware problems. Most of the major cloud operators today (*Amazon Web Services*[4], *Microsoft Azure*[5], *Google Cloud Platform*[6], and *IBM Cloud*[7]) provide IaaS for their customers. PaaS providers go one step further and provide only platforms on which customers can run their applications. Examples are classic web-hosting providers and also Microsoft Azure, *Google App Engine*[8], and *IBM BlueMix*[9], which all provide a platform to host websites or web-applications. The last type of cloud provider offers specific applications or software to the customers, which is called SaaS. Customers of such providers do not have any administrative rights and are restricted to using only specific services. A prominent example is the Customer Relationship Management service provided by *Salesforce*[10].

Depending on the cloud computing model, administrators encounter various challenges when managing their networks. While in SaaS scenarios they have full control of every aspect of the network, when renting out servers they can influence only how the underlying physical network is organized. In all other cloud models, administrators can manipulate the assignment of components. In the IaaS case, for instance, they can reassign VMs to servers. The more control customers have, the more they want to monitor, manage, and optimize the network, for example, by moving applications between different rented VMs. However, depending on the assignments between VMs and physical servers, moving an application may decrease the performance of other applications.

In general, customers would benefit from knowing, for instance, the assignments of their VMs to servers. However, for privacy reasons they are often only allowed to see coarse high-level information. Similarly, administrators of IaaS or PaaS providers would benefit from knowing details about applications run by their customers, to optimize the assignment for quality of service and energy consumption.

[3] `https://www.opennebula.org/`
[4] `https://aws.amazon.com/`
[5] `https://azure.microsoft.com/`
[6] `https://cloud.google.com/`
[7] `https://www.ibm.com/cloud/`
[8] `https://cloud.google.com/appengine/`
[9] `https://console.bluemix.net/`
[10] `https://www.salesforce.com/`

4.2.2. Goals

Over several months of close cooperation with our project partners, we analyzed the process of managing and optimizing cloud-based networks. There are commercial products in this field such as *VMWare's Distributed Resource Scheduler*[11], which try to optimize assignments of VMs to servers by analyzing their behavior automatically. However, such systems are of limited use for complex, heterogeneous cloud-based networks. They apply somewhat simplistic models and rules for optimization and work best in cases where all VMs are clones, as in a group of web servers. In heterogeneous networks, a deep understanding of the semantics and communication between the components from all three perspectives (physical, virtual, and application) is important to monitor and optimize the network effectively. Consequently, a visualization solution that targets these problems should allow administrators to:

G1: Monitor the status of the network by visually inspecting static performance information about the components (e.g., CPU power, available memory) and/or live performance and traffic data.

G2: Discover bottlenecks by analyzing the infrastructure's design in the context of the monitoring information.

G3: Optimize the network interactively. Depending on the requirements and purpose of the network, different optimization criteria exist. For example, if the internal communication needs to be minimized, the administrator's goal is to reduce the length of communication routes between components. If the task is to optimize the balance of resources, administrators should be able to change the mapping between components, e.g., the assignment of VMs to physical servers or assignment of applications to VMs.

4.3. Requirements

Below we present a list of requirements that an effective cloud monitoring and optimization solution must fulfill. We elicited the requirements in interviews and feedback sessions with cloud computing experts, one of whom is co-author of this chapter.

R1: Encode topology of cloud infrastructure. The visual representation of the cloud-based network needs to show relationships between components and encode different types of components.

[11] `https://www.vmware.com/products/vsphere/`

R2: Encode static or dynamic attributes. This includes static performance attributes such as installed main memory, hard disk capacity, and CPU specification. In the case of dynamically changing data, the visualization must encode attributes such as the current CPU load or main memory load factor. In addition to attributes of single components, the communication flow and connections between components needs to be represented effectively without cluttering the visualization.

R3: Enable time navigation. The user must be able to select interactively the time interval for which the streaming data is encoded in the network visualization. The selected time span should be either bound to the current time point or fixed to a static snapshot of the network.

R4: Support interactive changes of mapping relationships. It should be possible for users to optimize the cloud-based network by manipulating the mapping relationships between components.

R5: Scalability. The visualization needs to scale to a large number of components, many attributes, and a high traffic load.

R6: Encode topological evolution. An effective solution should enable users to explore, compare, and analyze changes within the structure and assignments in the network over time.

R7: Support privacy preservation. Administrators who are in charge of specific subparts of the network may not have the clearance to see all parts, but must be offered a privacy-preserving view, in order to minimize side effects when optimizing their part of the network.

4.4. Domain Related Work

CloudGazer is designed to address the three domain-specific goals formulated in Section 4.2.2. In this section we start with a discussion of commercial tools that target similar goals, followed by a consideration of related work in network traffic visualization. Finally, we summarize contextually relevant approaches to visually comparing and relating multiple hierarchies.

4.4.1. Cloud Computing Software

The majority of commercial tools follow a classic dashboard approach that enables users to monitor the current state of cloud-based networks (cf. goal G1). Dashboards are

mash-ups of simple graphs, statistical plots, and tables that present the network topology together with traffic and performance parameters over time. In most tools the dashboard is designed to provide a high-level overview of the network, from which users can drill down to lower-level information, such as single transaction events. Examples of such monitoring tools are *OPSView Virtualization Monitoring*[12] and *Dynatrace*[13]. Depending on the tool, information is presented at various levels of detail. The most condensed status of a network or components within the network are traffic-light representations. An inherent problem in many tools is that switching to more detailed information about one component or part of the network often results in a loss of context.

In general, dashboard solutions—if well designed—are well suited to addressing monitoring tasks (goal G1). However, discovering potential bottlenecks (goal G2) is difficult, since individual dashboard elements are often isolated from each other, which hampers the detection of relationships and anti-patterns that could cause problems in the near future. Most of the tools do not focus on integrated ways of optimizing and fixing problems (G3). One exception is *Cirba Control Console*[14], which gives hints on how to optimize the cloud-based network in order to prevent future problems and to increase cost-effectiveness. The hints are based on scores that are computed for all physical and virtual machines. Although the tool supports the task of optimizing the network based on static data effectively, it cannot cope with live streaming data.

Tools such as OPSView Virtualization Monitoring and Compuware APM for Enterprise Tiers present the overall structure of the cloud-based network in a single graph or tree representation. Although this works for small networks, it results in scalability issues with a growing number of components. Larger graphs get cluttered easily, making it hard for administrators to interpret and relate different components and their relationships in the context of live streaming data.

In summary, none of the available tools addresses all three goals effectively in a single solution. We therefore believe that the presented solution could have a significant impact on the design of next-generation cloud computing tools.

4.4.2. Network Traffic Visualization

The problem of visualizing computer networks has been and remains an active research topic in the visualization community. Most of this work focuses on the task of monitoring and analyzing traffic visually, for instance, to detect and react to attacks. Examples are the work by Fisher et. al that shows connections on top of a treemap which encodes the subnets of the network [FMK+08], and the *LiveRAC* system [MMKN08], which uses a space-filling

[12] `https://www.opsview.com/solutions/virtualization-monitoring`
[13] `https://www.dynatrace.com/solutions/`
[14] `https://www.cirba.com/`

layout for visualizing the status of nodes at multiple levels of detail over time. LiveRAC is particularly interesting, as it scales well to visualizing data associated with thousands of nodes. In LiveRAC and many other systems, the topology is secondary and often not even shown in the visualization. However, from a domain-independent visualization point of view, it boils down to the challenge of presenting topological information of a graph or hierarchy together with node and edge attributes that potentially change over time. Existing solutions usually focus either on the topology aspect (e.g., [NSS07]) or on the evolution of nodes and attributes over time (e.g., [MMKN08]). A notable exception is *enRoute* [PLS+13], where users are able to select a path in a biological network, which is then presented together with associated experimental data. Another exception is the work by Saraiya et al. [SLN05] that shows heatmaps and line charts as small glyph nodes embedded in a graph visualization. However, the solution becomes cluttered quickly if applied to graphs with more than a few dozen nodes. In *CloudGazer*, we strive to incorporate both aspects—topology and additional data attributes—while addressing the scalability issue by splitting the network into multiple perspectives.

4.4.3. Hierarchy Matching

As cloud-based networks are graphs, the vast body of work on graph visualization is applicable [vLKS+11]. Due to the fact that the workflows and the associated data change over time, also the state-of-the-art in the sub-field of dynamic graph visualization is relevant in this context [BBDW14, KKC14]. However, instead of visualizing the overall graph, we cope with the complexity of the cloud-based network by subdividing it into multiple hierarchies. This strategy requires reintroducing the lost relationships between the hierarchies. A vast body of related work exists on matching and comparing two or multiple hierarchies. A recent survey by Graham et al. [GK10] identified seven fundamental approaches to this task: i) drawing edges between spatially separate hierarchies; ii) highlighting related nodes; iii) animating between the hierarchy representations; iv) using matrix representations; v) agglomerating nodes that have multiple parents for display in the same representation; vi) 3D representation of interlinked hierarchies; vii) atomic view that shows only parts of the hierarchies on demand.

The first two approaches are options that we discuss in further detail in the remainder of this section. All other approaches are not applicable. Animation (iii) and atomic views (vii) are not viable options, as administrators need to see the status of all perspectives concurrently to be able to monitor them (see G1). Matrix representations (iv), such as the *RelEx* system [SFMB12], can match only two trees and are therefore not applicable in this context. Agglomerating nodes (v) can also be ruled out, as it would increase the complexity of the perspectives again. 3D representations (vi) suffer from occlusion and perspective distortion.

The first approach of drawing edges between hierarchies is well suited to identifying structural changes. *TreeJuxtaposer* [MGT+03], for instance, supports pairwise tree comparison. The work by Robertson et al. [RCC05] follows a similar idea for mapping two schema trees. Holten and Wijk [HW08] visualize two trees as space-filling icicle plots that face each other, where items between plots are connected by hierarchically bundled edges. Another example is *Code Flow* [TA08], which visualizes drifts, merges, and insertions between different versions of source code. The conceptual difference from the previous examples is that Code Flow visualizes the evolution between multiple states of the tree by showing each state in a parallel coordinate fashion. Although this extends the approach to multiple trees, it has the same limitations as parallel coordinates: only relationships between adjacent trees can be seen.

All these papers are good examples of how to address the comparison task effectively. In *CloudGazer*, however, we not only have direct 1:1 relationships between trees but inter-hierarchy relationships of varying cardinality (1:*n* and *n*:*m*), which makes it hard for users to understand the complex relationships between the trees. Thus, we use dynamically created inlays to address this problem (see Section 4.5.4).

The second approach identified in the survey uses coloring and highlighting to visualize relationships across trees. Bremm et al. [BvLH+11] proposed an interactive visual comparison of multiple trees where users need to select one tree as reference in order to see how it relates to others. All compared trees are presented as small views rendered next to each other. While this allows users to identify topological differences between trees, it is error-prone and slow, as it requires users to manually match the relationships by visually comparing them—a task that is known to be cognitively demanding. In *CloudGazer* we utilize interactive thumbnails to provide an overview of the perspectives. However, for communicating inter-perspective relationships, we rely on inlays that contain all relationships relevant to the current selection.

4.5. CloudGazer Visualization Approach

Even small-scale networks with a few dozen components can become hard to understand, as demonstrated in Figure 4.1. To address this issue, we apply a divide-and-conquer strategy where the overall graph of the network is broken up into component-specific perspectives (see Figure 4.2). Deriving perspectives from the overall graph is a one-time authoring step that can be performed automatically.

In *CloudGazer* we arrange the perspectives in a multiple coordinated view setup with each perspective shown as a separate view. The user can interactively choose a focus perspective that is presented in detail, while other perspectives are shown as interactive thumbnails. As illustrated in Figure 4.3, *CloudGazer* consists of *blocks* encoding component-specific

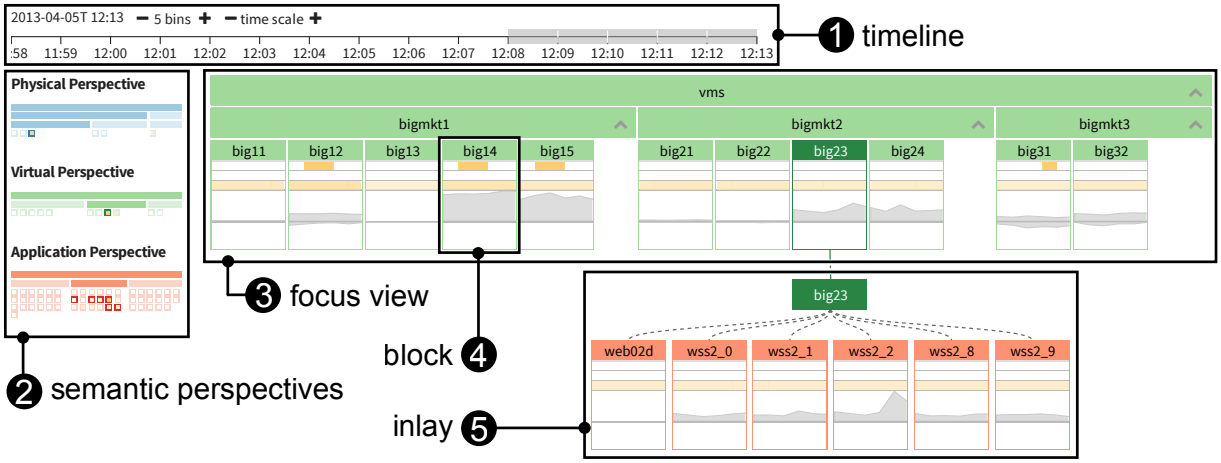


Figure 4.3.: Building blocks of *CloudGazer*. (1) Timeline for temporal navigation. (2) Semantic perspectives shown as interactive thumbnails. (3) Focus view presenting one perspective in greater detail. (4) Blocks visualizing a single component with its associated data. (5) Inlay showing relationships of selected nodes to other perspectives.

attributes (see Section 4.5.1), *interactive thumbnails* showing a high-level version of all perspectives, a *focus view* visualizing one perspective in detail (Section 4.5.2), *inlays* embedded in the focus view showing relationships of the focus perspective to others (Section 4.5.4), and a *timeline* for temporal navigation (Section 4.5.5).

By selecting components of interest, the user can investigate mapping relationships across perspectives effectively. We insert these relationships into foreign perspectives as inlays. To make the association between nodes and perspectives clear, we assign the same color to all nodes that belong to a particular perspective. In the following sections we discuss the building blocks of the *CloudGazer* system.

4.5.1. Blocks

Blocks are the basic visual unit of *CloudGazer* that facilitate monitoring the state of a single component, as illustrated in Figure 4.4. Depending on the usage scenario, a block encodes static performance information of components or live status and traffic information (see R2).

Stacked Bars We represent static component attributes as stacked bars normalized by their global maximum value (see Figure 4.4). The light gray bar encodes a value associated with the component, while the dark bar represents the sum of attribute values from other perspectives that are assigned to this component. For example, if the mapped attribute

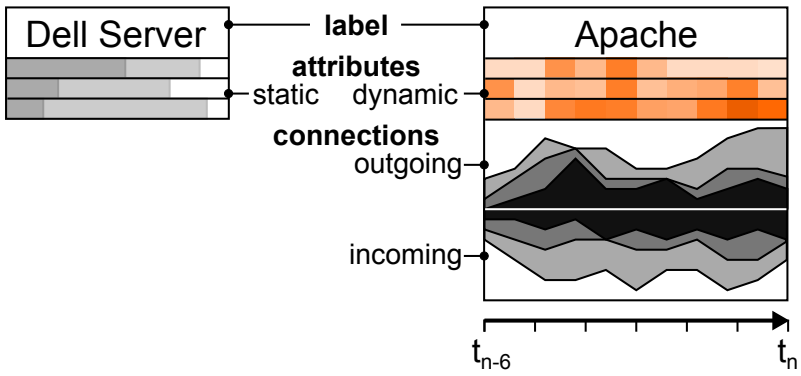


Figure 4.4.: Blocks represent the status of a single component. Static blocks (left) encode component attributes using stacked bars. Each bar corresponds to one attribute. Dynamic blocks (right) visualize live streaming data using heatmaps and streamgraphs. Each row of the heatmap encodes data from different attributes over time. In the case of live streaming data, new data is pushed into the heatmap from the right. The streamgraph in the lower part of the representation encodes incoming and outgoing connections. The height of the first inner layer corresponds to the number of connections with directly linked components. With each layer the distance in the hierarchy increases, as indicated by a decreasing brightness.

is memory, the length of the bar encodes the main memory installed on the server. While the dark gray bar is the memory allocated by all VMs hosted on this server, the light gray bar encodes the remaining free memory. To make the length of the bars comparable across components, we use the white portion of the bar to indicate a difference to the maximum value across all components on this perspective level.

Heatmap We use heatmaps to encode component attributes that change over time (see Figure 4.4). Each column represents a time step, and each row is associated with a different attribute. Possible attributes are CPU load, main memory load factor, and hard drive disk usage. New live traffic data is added as the last column on the right, pushing previous time steps one column to the left. The number of columns can be changed interactively by the user. We use a white (0) to orange (1) color scale.

Streamgraph We use streamgraphs to encode the communication with other components, as illustrated in Figure 4.4. Instead of explicitly visualizing the communication between individual components by changing the edge encoding, we group connections according to the number of intermediate hops in the perspective hierarchy and show the groups as layers of the streamgraph. Parent and child components, for instance, have a distance of one, while siblings and grandparents/grandchildren have a distance of two. All

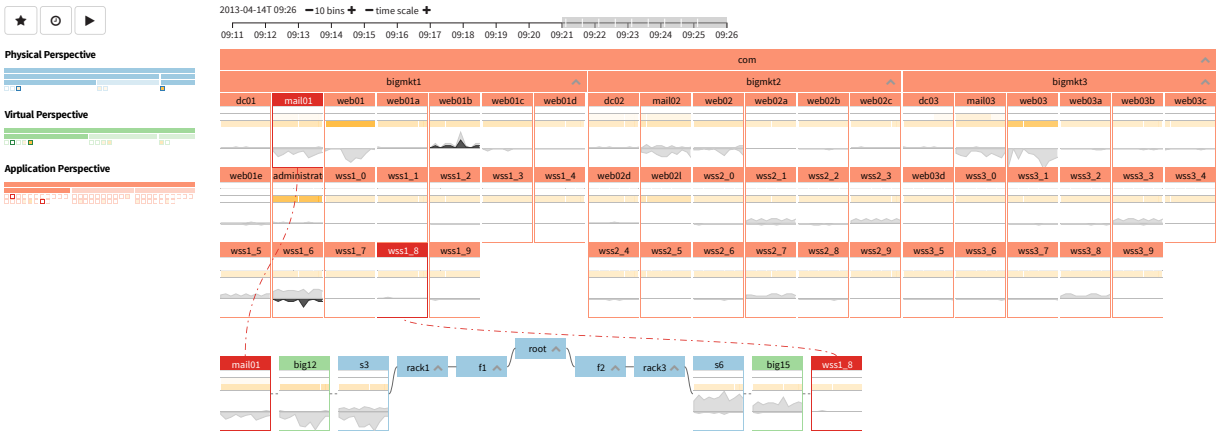
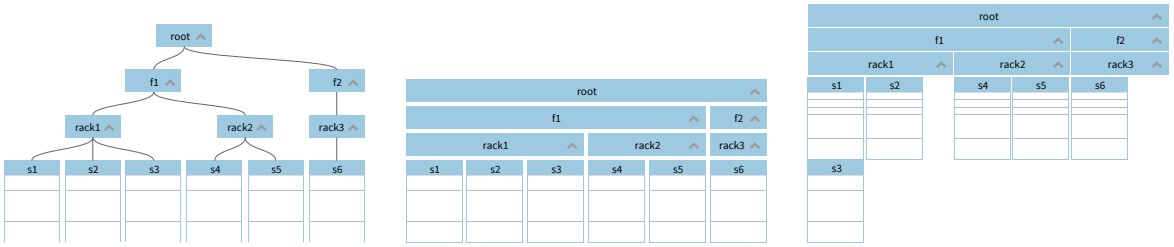


Figure 4.5.: Screenshot of *CloudGazer* showing the application perspective in the focus view (right) and the virtual and physical perspectives as interactive thumbnails (left). The user has selected the blocks representing '*mail01*' and '*wss1_8*' to inspect their relationship across the semantic perspectives. By looking at the dynamically created inlay (bottom), it becomes obvious that the virtual machine '*big15*' has a high load even though '*wss1_8*' has only few connections. The user concludes that another application on '*big15*' must cause the problem.

external communication with components outside the cloud-based network is summarized as the outermost layer. All groups are stacked according to their distance and normalized by their current global maximum value. In addition, we differentiate between incoming and outgoing connections by separating them into two charts, as show in Figure 4.3. Streams pointing upwards and downwards represent outgoing and incoming connections, respectively. Note that, depending on the user's preferences, it is possible to switch from streamgraphs to a stacked bar chart representation. Due to our design decision to encode communication in the block rather than the edges, point-to-point connections are lost. To alleviate this problem, a user can select a block in order to filter the data of all other blocks to contain only the communication with that selected.

As the streamgraph is the most salient part of the block representation, it should encode the attribute that is most relevant to solving the analysis task. In our usage scenarios, streamgraphs represent the number of connections, while the evolution of all other attributes is visualized in the heatmap. However, the mapping of attributes to the streamgraph and rows of the heatmap can be tailored to the usage scenario. Multiple stacked streamgraphs are also possible.

Note that we use the color of blocks contained in the interactive thumbnail perspectives to encode a single attribute. In the examples shown in the chapter, the color represents an aggregated value of the number of connections.



(a) Node-link layout (b) Icicle plot layout (c) Hierarchical Grid layout

Figure 4.6.: Hierarchy represented by the different layout approaches available in *CloudGazer*.

4.5.2. Focus View

The focus view is the central visualization in *CloudGazer* and presents in detail the currently selected perspective, as shown in Figure 4.3. The visualization of the focus perspective is linked with the thumbnails of all perspectives shown on the left of the interface. When the user selects a block in the focus view, all blocks in foreign perspectives that share a mapping relationship are highlighted in the interactive thumbnails, as shown in Figure 4.8.

CloudGazer supports various layouts for arranging the blocks effectively (addressing requirement R1). Widely used tree layouts such as node-link and icicle plots [KL83] have the disadvantage that they grow in size rapidly with an increasing number of leaf nodes. To alleviate the problem, we propose the *Hierarchical Grid* layout, which is explained in more detail in the following section. To account for the size of the perspective and the task at hand, users can switch freely between layouts.

Even with a space-efficient layout and dividing the network into multiple smaller perspectives, the most crucial issue of the focus view is its scalability to larger numbers of blocks without sacrificing the ability to track and monitor individual components (see also requirement R5). In *CloudGazer* we take a series of measures to address this issue:

Collapse & Expand By clicking on nodes, users can collapse all child nodes to a single node. For example, in the application perspective all web applications can be collapsed on demand. The collapsed proxy node then shows aggregated information from all hidden child nodes.

Hierarchical Zooming To quickly focus on certain sub-parts of the hierarchy, users can double-click on a node to turn into the new root of the displayed hierarchy. This kind of navigation is particularly useful for large hierarchies.

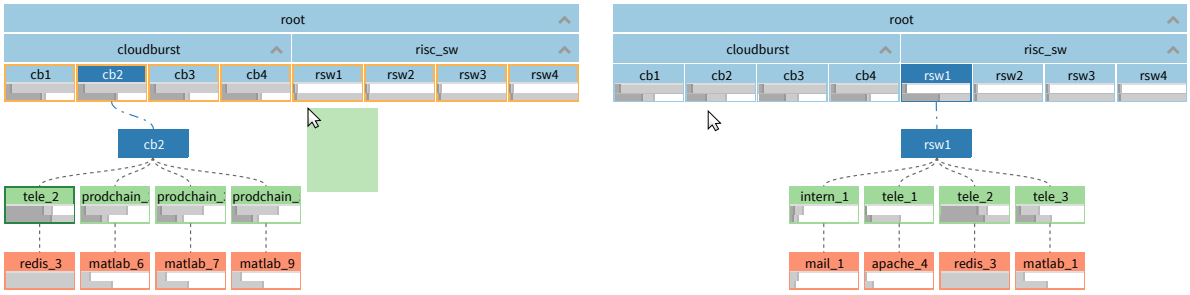
Activity-Based Shrinking In real-world scenarios not every component will be active. Depending on the current load of the network, some components might have little or no communication at all. These components are less relevant to administrators. Thus, they can be visualized in a simplified and more compact form. For these cases, older time steps can be removed, to make the blocks thinner, or blocks can even be hidden. *CloudGazer* optionally provides automatic adaptation of component width according to its current activity, calculated by the attributes' variance over time. The thinner a block, the less of its attribute's history is shown, which ensures that blocks remain comparable. In addition, the blocks can be automatically ordered by activity, such that active ones are moved to the front, the downside of which is a constantly changing layout.

4.5.3. Perspective Layouts

An effective layout for arranging hierarchically structured perspectives is an important success factor for a cloud-based network visualization. For smaller hierarchies we provide a regular node-link tree layout (see Figure 4.6a). To increase the scalability in terms of leaf nodes (R5), we introduce the *Hierarchical Grid* layout.

The Hierarchical Grid layout is a modified version of an icicle plot [KL83]. As icicle plots are space filling, the node widths on each level of the hierarchy are determined by dividing the available width by the number of nodes. Figure 4.6b shows a small hierarchy represented as an icicle plot. However, with an increasing number of nodes, the node width can become too small. As we visualize traffic data and additional attributes inside the nodes, a reduced node width also reduces the space for visualizing data. In classic icicle plot implementations, users can alleviate the problem by zooming into a part of the hierarchy by promoting a node to become the new root. In the Hierarchical Grid layout we arrange the leaf nodes in a grid-like structure. While this causes the representation to grow downwards, the node width for the leaves is kept constant, as can be seen in Figure 4.6c. A constant node width is important to make the data shown in the streamgraphs and heatmaps comparable across blocks. Figure 4.8 shows the layout applied to an application perspective with 54 blocks. The design decision to keep the node width static in a space-filling layout can result in empty space between branches of the hierarchy. However, we consider this to be a minor esthetic issue that does not have a negative impact on functionality.

Both icicle plots and the Hierarchical Grid layout express the hierarchy implicitly through the position of the nodes. If nodes from different levels touch each other, they share a relationship. As traffic information is encoded in the nodes themselves and not on the edges, icicle plots and the Hierarchical Grid layout are more space efficient than explicit node-link diagrams. Due to the compactness of the Hierarchical Grid and the fact that it can also be interpreted when rendered smaller, we use this layout for the interactive thumbnails of the perspectives, as shown on the left in Figure 4.8.



(a) Before optimization (b) After optimization

Figure 4.7.: Server perspective with the selected server '*cb2*' and the corresponding inlay with related VMs and applications. The two stacked bars encode the components' main memory and disk space (dark gray = used, light gray = free, white = empty space to make bars comparable across components, i.e., only present if available memory is different between components on the same hierarchy level). To reduce the connection distances between the VMs '*tele_1*', '*tele_2*', and '*tele_3*', the administrator reassigns the VM '*tele_2*' to server '*rsw1*' via drag-and-drop.

Note that in this chapter we focus on hierarchically structured perspectives. However, the presented visualization concept is independent of structure and layout of the perspectives and could also be applied to general graphs or other specialized topologies.

4.5.4. Inlays

A downside of splitting the overall network into multiple perspectives is the loss of visual representation of the mapping relationships between components belonging to different perspectives. *CloudGazer* addresses this problem by letting the administrator select blocks for which the lost context will be reintroduced using *inlays*. Inlays are dynamically created graphs that are assembled according to the current block selection. When the user selects a single component in one perspective, all related components from other perspectives are added to the inlay graph. If the selected block is a server, for instance, the inlay contains all VMs hosted by this server and all applications running on the VMs (see Figure 4.7). If the user selects a second block, the inlay algorithm looks for a path that connects the selected components in the other perspectives (R1). If a path can be found, all components along the path will be part of the inlay, as can be seen in Figure 4.8.

In the focus view, which shows the selected perspective, we add the inlay at the bottom. If the inlay shows the path between two selected blocks across multiple perspectives, the left-most and right-most blocks are duplicates of the originally selected block. We use dotted lines to connect the original blocks with the duplicates in the inlay. Animated transitions [HR07] help users to track visual state changes when adding the inlays.

By inspecting components and relationships in inlays, administrators can discover bottlenecks in the cloud infrastructure (addressing G2). In addition, *CloudGazer* enables users to optimize the network proactively by changing mapping relationships (G3). Using drag-and-drop makes it possible to reassign components across perspectives (fulfilling R4). If blocks visualize live traffic data, the impact of the changes can be observed immediately.

4.5.5. Timeline

The interactive timeline provides for temporal navigation and for choosing the time span that shows up in blocks (R3). The length of the time span directly influences the block width. The time span is discretized into multiple bins, as indicated within the selected time span shown in the top of Figure 4.3. The number of discrete steps can be changed interactively and is used within blocks to bin attribute values.

4.5.6. Implementation

CloudGazer is an HTML5 web application that uses D3 [BOH11a] for visualization and the AngularJS[15] web framework to mash up elements. The server part is written in Python using the Tornado framework[16] to provide live traffic data. The interaction with the prototype system is demonstrated in an accompanying video.

4.6. Usage Scenarios

We demonstrate the effectiveness of *CloudGazer* in two scenarios. The first discusses how the prototype system can be used to optimize statically a cloud-based network of one of our partners by optimizing assignments between components (G3). The second uses simulated data to demonstrate how *CloudGazer* can be used to monitor dynamic networks (G1) and discover bottlenecks (G2).

4.6.1. Optimizing a Cloud-Based Network

Our project partner *RISC Software GmbH* specializes in administrating various cloud infrastructures for different customers and research projects. They maintain an IBM CloudBurst with four physical servers. Each CloudBurst server has 72 GB memory, is connected

[15] `https://angularjs.org/`
[16] `http://tornadoweb.org/`

to a 40 TB storage array, and has a 10 GBit connection. The VMs have 512 MB to 32 GB of memory assigned, which can be flexibly allocated. In addition, they maintain a second rack with four servers, each with a configuration of 128 GB memory, 2 x 1 TB internal storage, and a 10 GBit connection.

A research project on traffic engineering is collecting and processing telematics data. Three applications are required for this purpose: a database, a computation server, and a web server, which initially run on independent VMs (*tele_1*, *tele_2*, *tele_3*) and on separate CloudBurst servers (*cb1*, *cb2*, *cb3*). However, the data transfer between the database and the computation server results in high internal network load. The administrator decides to merge the three applications on one physical server in order to reduce communication distances. Figure 4.7 shows the server perspective in focus with related VMs and applications running on the selected server *cb2* as inlay. The visualization in *CloudGazer* shows that the CloudBurst servers have insufficient memory capacity to merge all VMs on one single server. The administrator explores the servers of the other rack and discovers that server *rsw1* has enough available memory for hosting all project-related applications. Using drag-and-drop, he assigns the VM of each CloudBurst server to server *rsw1*.

4.6.2. Monitoring Dynamic Cloud-Based Network

In the second usage scenario we demonstrate the *CloudGazer* system with simulated data from the 2013 Big Marketing VAST Mini-Challenge [CGW13]. The dataset consists of NetFlow data along with additional server attributes, including CPU load and memory usage, collected over a period of two weeks. We interpret the given network infrastructure as the application perspective. Based on this dataset, we generated a virtual and physical perspective. Since the data are relatively sparse, we aggregated them such that one second in the visualization corresponds to 60 seconds in the dataset. Further, we combined all workstations of each Big Market section in ten characteristic workstations running on terminal servers. Together with cloud computing experts we created the following use case to demonstrate how *CloudGazer* supports administrators monitoring a network based on live data:

The administrator of the Big Market network is responsible for handling customer requests concerning problems with the cloud infrastructure. A customer reports a problem accessing her e-mail and other applications. The administrator starts to investigate the issue by looking at internal logs. He finds out that the customer is logged in as *wss1_8* and decides to look at the status of the application in *CloudGazer*'s application perspective (see Figure 4.8). Each block in the focus view shows the overall status of an application, including CPU load and disk usage as heatmaps and incoming/outgoing connections as streamgraphs. However, neither *wss1_8* nor the mail server *mail01* are under heavy load according to the block visualizations. There is some external traffic on *wss1_8*, but this seems to be regular traffic caused by the customer's web usage. The administrator suspects

that not the applications themselves are the problem but the VMs they are running on, and in particular their physical relationship. By selecting both blocks *mail01* and *wss1_8* in the focus view, an inlay is added showing with which VMs and servers the applications are associated. He realizes that *big15*, which hosts the customer's workstation, is under heavy load, as shown in Figure 4.8. The administrator clicks on the VM block, which makes *CloudGazer* switch to the virtual perspective that shows an inlay of all applications hosted on the selected VM. The administrator realizes that one application, *wss1_6*, is consuming most of the VM's resources. Using drag-and-drop, he moves this application to an idle VM. This solves the customer's problem, since the VM can now provide more resources to the workstation, and the overall network is again more balanced.

4.7. Performance Predictions

With our *CloudGazer* approach users can monitor and optimize cloud-based infrastructures based on live streaming data. In order to find possible optimizations users must discover potential performance bottlenecks beforehand. However, with the approach presented in Section 4.5 users can only address potential performance bottlenecks after they occur. In order to increase the reliability and efficiency of cloud infrastructures, the goal is to prevent future bottlenecks before they happen. In this section we present a visualization concept that enables administrators to browse recorded historical data, monitor live performance data, and explore predicted performance data. The ultimate goal of our visualization concept is to let administrators visually investigate predicted performance bottlenecks (events) and compare alternative recommendations for preventing these potential problems.

Our visualization concept is targeted at administrators who should be able to monitor cloud infrastructures and optimize them by visually evaluating a list of predicted events and recommendations. Figure 4.8 shows a design sketch that combines *CloudGazer*, *ThermalPlot* (see Chapter 3), and LineUp [GLG+13] with a new prediction view that is tailored to the specific domain problem. The concept consists of a series of linked views that are described in the following.

ThermalPlot View The *ThermalPlot* visualization provides an overview of the cloud infrastructure by positioning the components in a space that maps the criticality of components to the x-axis and the positive and negative change of the criticality to the y-axis (see Figure 4.8b and Chapter 3). Consequently, the higher the DOI value, the more critical a component is, and the further on the right it will appear in the *ThermalPlot* space.

How critical the status of a component is, is calculated by a configurable DOI function (see Figure 4.8a) that is computed as a weighted sum of multiple performance attributes,

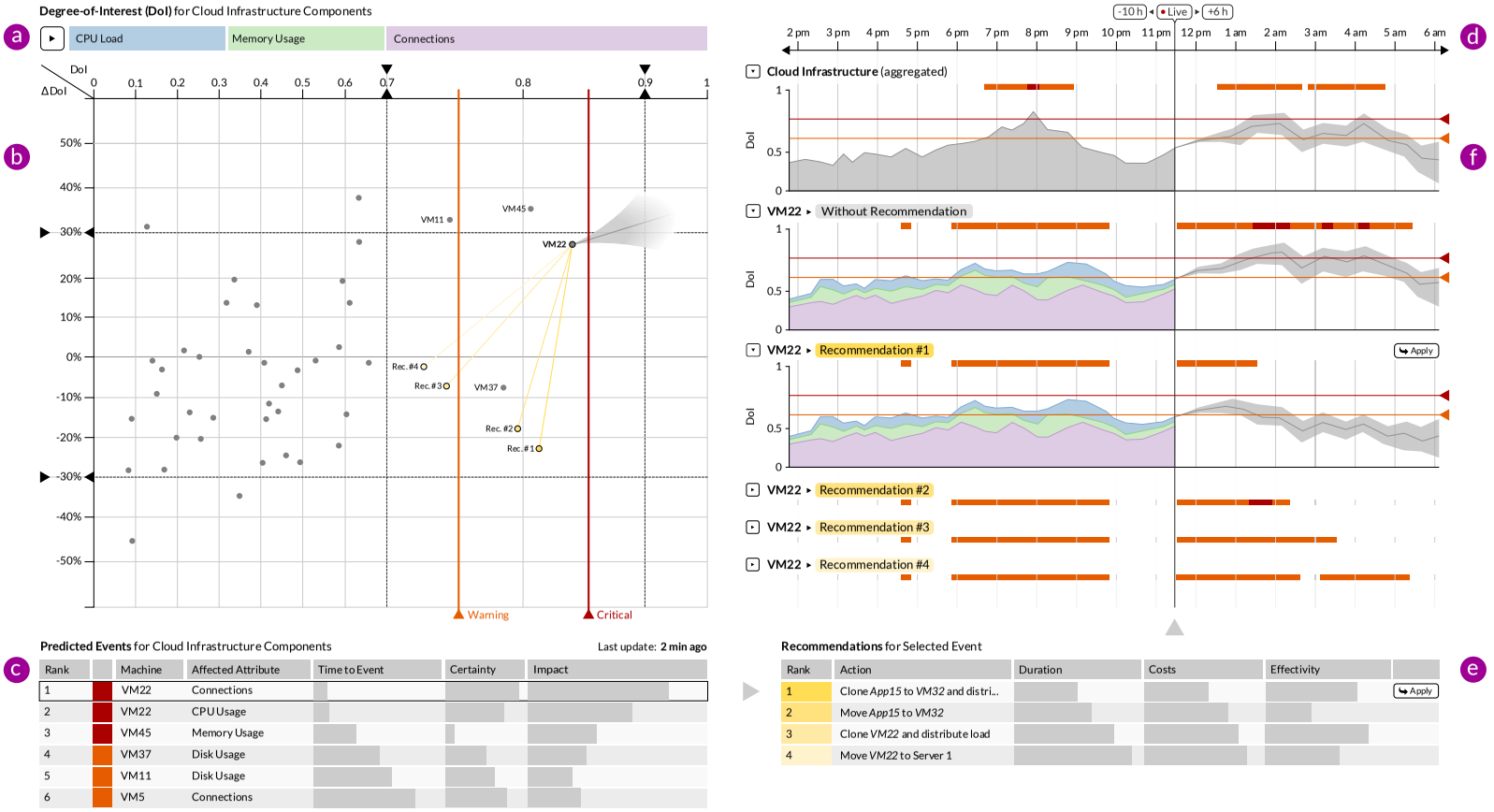


Figure 4.8.: Design sketch of the proposed visualization concept. (a) The administrator can configure the degree of interest that determines how critical the status of a component currently is. (b) The *ThermalPlot* shows the current as well as predicted positions of components in the DOI space. (c) Ranked list of predicted performance bottleneck events. (d) Selected time span with past and future. (e) Ranked list of possible countermeasures for avoiding the predicted event. (f) Detail view visualizing the past and predicted items' performance using *peak bars* (collapsed state) and streamgraphs (expanded state).

such as CPU load, RAM usage, and the number of currently opened connections. Which performance attributes are taken into account by the DOI computation depends on the use case. Additionally, we show two vertical threshold lines that discretize the criticality of components into two states: 'warning' and 'critical'. The administrator can freely configure the thresholds of the states by changing the vertical position of the lines via drag and drop.

Predicted Events Ranking In the lower part of the visualization we show an interactive ranking containing a list of predicted critical events (see Figure 4.8c and [GLG+13]). Events are the result of a prediction model that uses the historical and live data collection to discover potential bottlenecks. Each event consists of a cloud infrastructure component and a specific attribute that is expected to be critical, together with the estimated time span until the event occurs (the shorter the closer), the certainty of the prediction (the longer the more likely), and a value that quantifies the impact the event has on the perspective or infrastructure (the longer the bigger). An example for an event would be a rapid increase in the number of connections that a component must handle, which also results in a higher CPU load. Selected events are visualized in the *ThermalPlot* space with the predicted position and a funnel that encodes the certainty (see Figure 4.8b).

Timeline The timeline (see Figure 4.8d) shows the selected time range (e.g., 10 hours into the past and 6 hours into the future) for the DOI computation (past to live) and prediction (future). Adjusting the selected time range triggers a re-computation of all DOI values and events.

Recommendations Ranking Selecting an event from the event ranking (see Figure 4.8c) brings up a separate ranking with recommendations that are supposed to prevent the event from happening (see Figure 4.8e). Each recommendation consists of a description for the action and a multi bar chart representing the predicted duration that is needed to apply this recommendation (the shorter the faster), the cost estimation (e.g., bandwidth; the longer the more expensive), and an effectiveness estimation to reduce the event's impact (the longer the more effective). Recommendations are also shown (with their respective color coding) in the *ThermalPlot* space (see Figure 4.8b) and in the detail view.

Detail View The detail view (see Figure 4.8f) shows a selected component from the *ThermalPlot* or the event ranking view, together with all recommendations to potentially solve the issue. Each item can be represented as *peak bar* (collapsed state) or as a detailed streamgraph (expanded state). The peak bar visualizes DOI values that are above a certain threshold in the corresponding threshold color. The predicted performance changes that are expected for the different recommendations are also visualized using peak bars.

When expanding the representation, we reveal the more detailed DOI streamgraph that visualizes the full time-series data within the selected time span. Available data in the past is visualized as a stacked DOI streamgraph that represents how much each attribute contributes to the aggregated DOI value over time. For predicted future performance values we show the predicted maximum, expected, and minimum DOI value. This encoding helps the administrator to judge how uncertain predictions are and what the predicted worst, average, and best case is.

We only show the aggregated uncertainty for the overall combined DOI value. However, streamgraph representations cannot encode the individual uncertainty for each attribute. As a next step, we plan to evaluate different visual encodings that address this limitation.

Furthermore, our current concept does not visualize relationships between components in the infrastructure. However, this knowledge can be critical for the administrator to better judge side-effect that might be introduced by recommended infrastructure changes. In future work we will explore ways of how to include this topological information into our concept.

4.8. Discussion and Limitations

Scalability Scalability is the most critical concern when developing monitoring visualization techniques for large-scale cloud infrastructures. The strategy of splitting the network into semantic perspectives alleviates the problem but does not conclusively solve it. To further increase the scalability of *CloudGazer* so it can cope with large cloud-based networks, we take several measures, the most important of which is an optimized layout (Section 4.5.3) and specially designed interaction techniques (Section 4.5.2).

While the strategy of breaking up a cloud-based network into smaller semantic perspectives increases its scalability to larger networks, it also introduces problems concerning loss of relationship representations. In *CloudGazer* we address this issue by adding inlays that show relevant portions of related perspectives on demand.

Communication Encoding Communication between components is mainly 1:1. A naive approach is to visualize all connections using a node-link diagram and encode the amount of traffic by changing the width or color of edges. However, in discussions with our project partners we found that the communication distance, i.e., the number of intermediate hops, is more relevant than the actual communication endpoint. Therefore, we use streamgraphs to encode communication distances for each component (see Section 4.5.1). When a user selects a component, we filter all other streamgraphs to show only the communication with that component, allowing users to inspect the 1:1 connections on demand.

Optimization Costs *CloudGazer* allows administrators to optimize their networks interactively by manipulating the assignments of components across different perspectives. For example, administrators can move one VM to another server in order to optimize the communication distances or average server load. However, moving a VM to another server entails costs, such as the transfer time from one server to another or a possible short outage of the VM. In the current version of *CloudGazer* these costs are not considered, even though they can influence the usefulness of optimizations in terms of cost/benefit ratio.

Evolution Monitoring a network consists both of tracking the status of its components and of monitoring for changes in its topology due to reconfiguration in response to changed requirements or to optimization measures. Thus, perspectives and the mappings between them also change over time. Finding ways to let administrators track changes and evaluate their consequences is an interesting research question (R6), which we plan to address in the future.

Privacy Preservation In large-scale networks, multiple administrators work on different parts or aspects of the same network. Some users may have limited clearance to see certain parts of the network. However, to avoid side effects during concurrent optimizations, it is beneficial to give them an overview of the whole network. Privacy-preserving visualization techniques need to be applied to provide abstract overviews without showing details that are not allowed to be seen by certain users. A simple approach is to hide labels. However, even without labels individual components might be identifiable due to characteristic communication or attribute patterns. Consequently, more sophisticated privacy-preserving visualization techniques must be integrated [DK11]. *CloudGazer* does not yet include such measures. Therefore, requirement R6 remains open for future work. In particular, the problem of finding the right balance between costs and benefits of privacy-preserving network visualizations is an interesting topic.

4.9. Summary

We have presented *CloudGazer*, a flexible visualization solution for monitoring and optimizing cloud-based networks. Following the divide-and-conquer principle, we first divide the overall network into smaller semantic perspectives that are easier to understand and handle. In a second step, we reintroduce the lost inter-perspective relationships for selected parts of the focus perspective using a topology-driven DOI function. The extracted path of components is added as dynamic inlays.

5 | AVOCADO Topology- and Attribute-based Focus+Context for Provenance Data

Contents

A major challenge in data-driven biomedical research lies in the collection and representation of data provenance information to ensure that findings are reproducibile. In order to communicate and reproduce multi-step analysis workflows executed on datasets that contain data for dozens or hundreds of samples, it is crucial to be able to visualize the provenance graph at different levels of aggregation. Most existing approaches are based on node-link diagrams, which do not scale to the complexity of typical data provenance graphs. In our proposed approach, we reduce the complexity of the graph using hierarchical and motif-based aggregation. Based on user action and graph attributes, a modular DOI function is applied to expand parts of the graph that are relevant to the user. This interest-driven adaptive approach to provenance visualization allows users to review and communicate complex multi-step analyses, which can be based on hundreds of files that are processed by numerous workflows. We have integrated our approach into the *Refinery Platform* [1], an analysis platform that captures extensive data provenance information, and demonstrate its effectiveness by means of a biomedical usage scenario.

[1] `http://refinery-platform.org`

5.1. Introduction

Recent advances in biomedical research have enabled the rapid acquisition of data from biomedical samples for clinical and pre-clinical studies. The bioinformatics workflows employed to analyze such data incorporate many distinct steps and tools that often result in long workflows. Moreover, the complexity increases with repeated workflow execution and the number of samples processed. In the long term, the analyses associated with a biomedical study become hard to maintain, compare, and reproduce. To address this issue, all parameter modifications and workflow executions need to be captured as provenance information. Workflows may also be modified or executed multiple times with different parameters, or may use a different dataset as input, which results in a complex data provenance graph. Most existing visualization approaches are based on node-link diagrams, which usually do not scale well to large provenance graphs of dozens to hundreds of nodes. Hence, a major challenge is to visualize such graphs effectively in order to allow analysts to understand the dependencies between different files in a dataset.

In recent work by Ragan et al. [RESC15], provenance was characterized by the supported provenance *type* and *purpose*. According to their organizational framework for provenance, our approach operates on *data provenance* containing all executed workflows together with their parametrization as well as their in- and output files. The purpose of our visualization is to *recall* the analysis history, thus enabling analysts to better understand complex analyses, and to *present* the information to both colleagues who are involved in a particular project and others that are not part of the team, such as the general public. *Replication* and *action recovery* (undo/redo), however, are not direct goals of our visualization, as this is typically handled by the tools that capture and manage data provenance information.

The primary contribution of this paper is AVOCADO (Adaptive Visualization of Comprehensive Analytical Data Origins), an interactive provenance graph visualization approach that visually aggregates the data provenance graph by exploiting the inherent topological structure of the graph. Based on the aggregation, we then expand relevant parts of the graph interactively using a multi-attribute DOI function. As a secondary contribution, we integrated the visualization approach into the *Refinery Platform* [2], which captures, manages, and allows users to operate on data provenance information from bioinformatics workflows. We demonstrate the effectiveness of our visualization by means of a usage scenario.

[2] `http://refinery-platform.org`

5.2. Background

In the past decade, biomedical research has transitioned from being a primarily hypothesis-driven to a data-driven endeavor. This has been the result of the availability of large amounts of heterogeneous data, for instance from genomics studies, electronic health records, imaging data. To analyze such data, complex bioinformatics workflows are employed that usually operate on files that are run through a series of specialized tools. Particularly in the analysis of genomic data (e.g., from studies that aim to identify driving mutations in cancer or to pinpoint genetic variants that cause rare diseases) several multi-step workflows are commonly employed for quality control, preprocessing and data normalization, identification of statistically significant differences between cases and controls, identification of correlations, and other higher-level analyses (e.g., to identify changes in gene expression levels associated with a particular genomic mutation).

The failure to reproduce the results of a large number of such studies has raised major concerns in the biomedical research community and triggered several efforts to address this issue [BE12, HG13, BI15, Buc15]. The reasons why attempts to reproduce such studies fail are diverse and range from inadequate statistical power to publication of incomplete or incorrect study protocols and to unavailability of raw experimental data [Kai15]. In particular, the failure to record and share data provenance information for published data frequently renders results of computational analyses irreproducible. Even when such information is published, it can be extremely challenging to reproduce a study [GBLZ+15]. In recent years a number of bioinformatics data analysis systems have been developed that aim to track data provenance automatically and comprehensively (e.g., *Kepler* [ABJF06], *Taverna* [WHe13], *Galaxy* [GNT10], *VisTrails* [BCS+05]). However, they lack adequate visualization tools to review and communicate this provenance information, which severely limits its value for reproducibility purposes.

A comprehensive effort to facilitate collaborative and reproducible biomedical research is the open source *Refinery Platform*, which integrates data management and analysis. Refinery handles data at the file level and facilitates the execution of workflows on one or multiple input files in the *Galaxy* bioinformatics workbench [3]. For each of these analyses, Refinery automatically tracks comprehensive data provenance, including workflow templates applied, workflow parameters, tool versions, input files and the user executing the analysis. Every *analysis* consists of one or more *analysis input groups*, which correspond to the execution of a Galaxy *workflow* on a set of input files (see Figure 5.1). The raw data sets are imported into Refinery as Investigation-Study-Assay (ISA-Tab) files [RSBM+10], which provide metadata and information about the raw data generated. For example, if a user selects 10 files to be processed by a workflow that takes one input file and produces one output file per input file, then the corresponding analysis would have 10 inputs and 10 outputs and would consist of 10 analysis input groups. Every analysis uses only exactly

[3] `https://galaxyproject.org/`

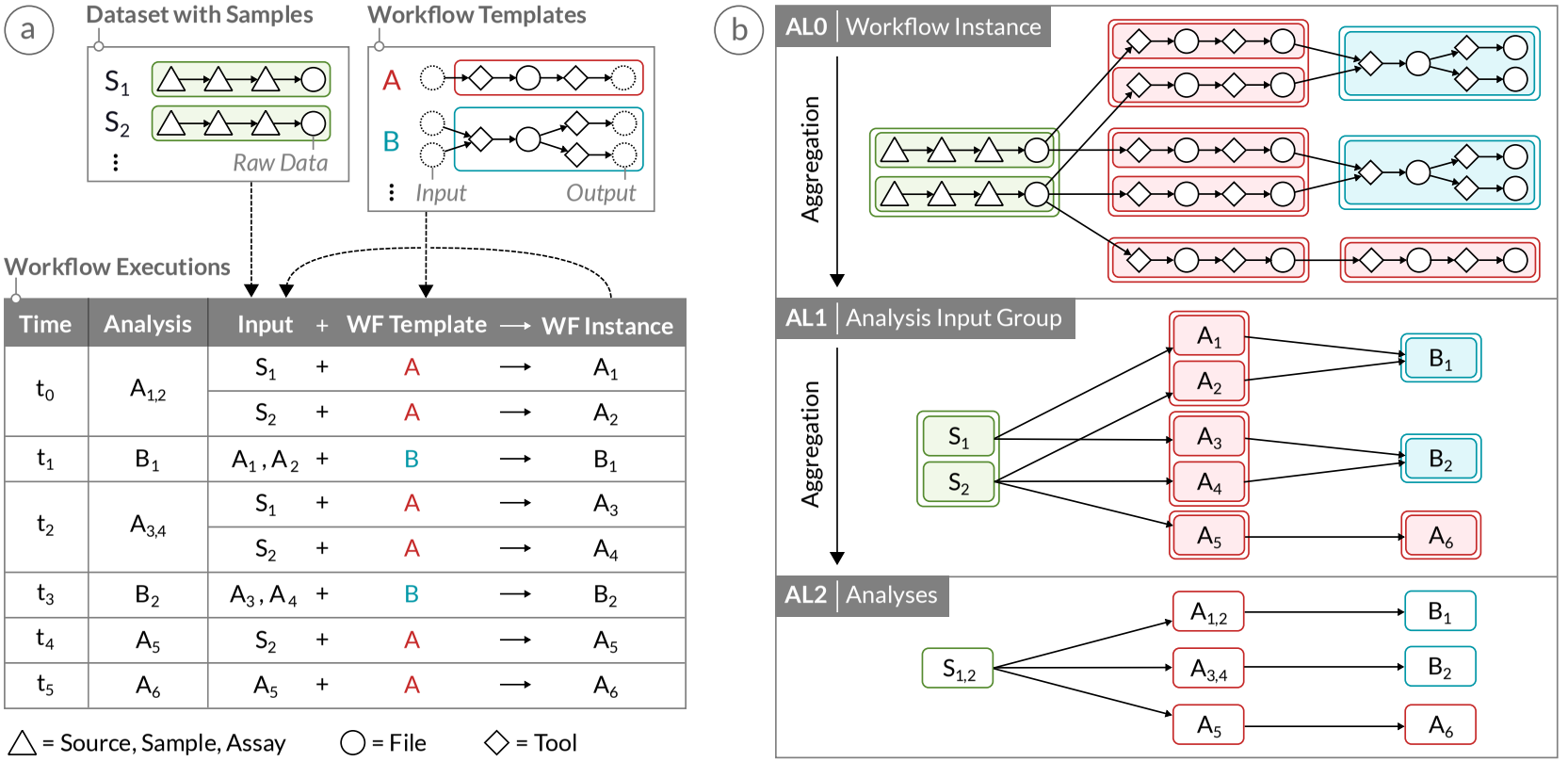


Figure 5.1.: A workflow provenance graph consists of multiple workflow executions over time. (a) Each execution requires one or more input files and applies a workflow template, containing a sequence of tools and files. (b) The provenance graph can be visualized at multiple aggregation levels (AL) exploiting the hierarchical structure of the graph. AL0 shows the connected *workflow instances* that are first aggregated to *analysis input groups* at AL1 and further aggregated to *analyses* at AL2. The aggregation methods are explained in Section 5.4.1.

one workflow template. Along with the metadata attributes that users can assign to files in Refinery, the data provenance information represents a richly annotated graph that contains all information necessary to reproduce the findings of a study performed with the help of the system.

Data provenance graphs such as those generated by Refinery and similar tools are directed acyclic graphs (DAGs). They contain different types of node: primarily file nodes and tool nodes. The latter represent the software that is used to process the files in a particular workflow. As illustrated in Figure 5.1, tools and files in workflow templates, analysis input groups, and analyses form a hierarchy that can be exploited to aggregate parts of the graph, as discussed in Section 5.4. Furthermore, data provenance graphs are often very broad, because each raw input file is run through the same set of workflows with either the same or different tool parameter settings. The length of the path from a raw input file to a highly processed output file can include a dozen steps or more, making the graph not only broad but often also deep.

5.3. User Tasks

In close collaboration with bioinformatics experts and with the input from biomedical researchers, we refined Lee's et al. task taxonomy for graph visualization [LPP+06] to identify a series of tasks that need to be supported by an effective data provenance graph visualization. The experts with whom we worked (one is a co-author of this contribution) are leading the development of the Refinery Platform and related projects. One of the authors, a visualization expert, worked very closely with the Refinery Platform team for more than 12 months.

T1: High-Level Overview Users want to start the exploration by inspecting an aggregated version of the data provenance graph, to gain an overview of which workflows were run and how often, in which configuration, and at which point in time. While further details should be hidden and summarized, an indication of the actual graph's depth and breadth is desired.

T2: Attribute Encoding Analyses are annotated with a series of attributes such as date and time of execution and in- and output files. This information should be accessible through the provenance visualization.

T3: Drill-Down on Demand Due to space constraints, the graph should be presented in the most reduced form possible, but users still need to be able to view information at the finest level of detail. The visualization should enable drill-down operations into sub-graphs that are of current interest, while the rest of the graph should be kept in a compact representation as context.

T4: Investigate Differences in Aggregates The precondition for aggregating analyses is a common workflow template. However, tool parameters, input files, and the number of analysis input groups might vary, for instance, when an analysis was re-run with additional inputs or parameters. The data provenance visualization needs to provide users with the means to identify and investigate such differences.

T5: Investigate Causality A crucial task in the exploration of data provenance graphs is to enable users to investigate the chain of files and transformations that contributed to a certain analysis result.

T6: Search and Filter Users should be able to focus on certain aspects of the graph, for example, a specific workflow type, provenance data changes, or execution time range, by triggering filter and search actions.

Although these tasks were developed with the input of experts who are working in the biomedical domain, we expect that these tasks are also relevant to other application domains in which similar analysis pipelines are employed for data-driven research.

5.4. The AVOCADO Visualization Concept

In AVOCADO, we reduce the complexity of the data provenance graph through a combination of graph aggregation and expansion strategies. Parts of the aggregated graph that are relevant to the user are expanded on demand by applying a modular DOI function. This interest-driven adaptive approach allows us to handle complex multi-step analyses that can be based on hundreds of files processed by multiple workflows (see Section 5.2). Figure 5.2 illustrates our visualization using a data provenance graph with 13 analyses containing 927 nodes from different workflows.

5.4.1. Graph Aggregation Strategies

The first part of our approach reduces the data provenance graph through a combination of hierarchical and motif-based aggregation chosen to provide a meaningful overview that preserves the overall graph structure (T1).

Hierarchical Aggregation Hierarchical aggregation reduces the size of a dataset by grouping related data items into aggregates based on the result of a recursive clustering operation. In our case, we make use of the inherent hierarchy contained in the data provenance graph. We aggregate *workflow instances* (AL0) into *analysis input groups* (AL1) and further into *analyses* (AL2) (see Figure 5.1(b)). Analyses contain all analysis input groups that share the same workflow execution time. In AVOCADO, we render all

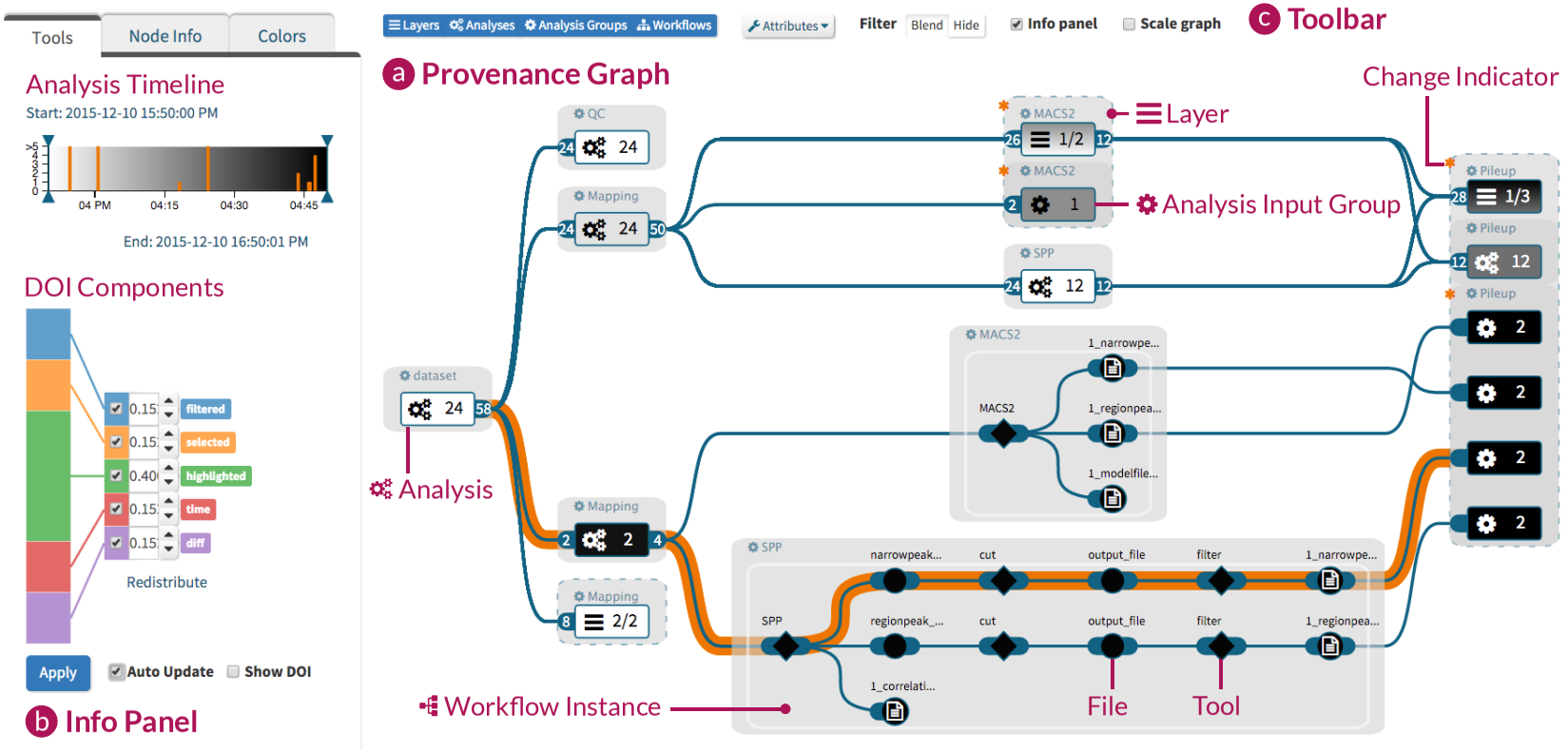


Figure 5.2.: The provenance graph (a) is aggregated and filtered based on the selected analysis execution time and the weighted DOI components (b). In the top center of the graph, two horizontally aligned workflows show a compound layer node, where the top node represents the layer itself, while two workflows are extracted because their specific DOI values exceeded a predefined threshold. The toolbar (c) allows users to switch between node-type-specific views (layer, analysis, analysis input group, workflow instance) and to change the attribute mapping onto nodes.

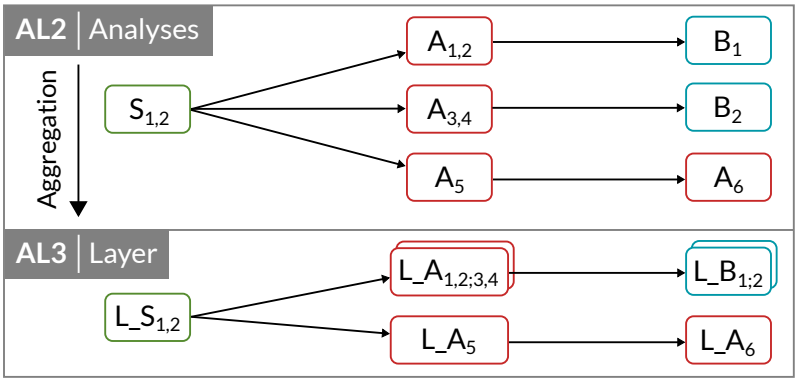


Figure 5.3.: AL2 corresponds to AL2 in Figure 5.1(b). The additional aggregation level AL3 groups similar analyses into layers using motif-based aggregation.

aggregation levels from top (AL2) to bottom (AL0), with the result that analyses, analysis input groups and workflow instances are stacked on top of each other. Each aggregation level is traversed in a breadth-first approach, placing the nodes in a column-based layout.

Motif-Based Aggregation A motif compresses a graph or parts of it while preserving the basic graph structure. Traditional motif discovery algorithms search for all permutations of a fixed number of nodes. In AVOCADO, motifs visualize the overall structure of the study. We aggregate analyses that use the same workflow template into a combined *layer* node. Although the analyses use the same workflow template, they may vary in the number of incoming and outgoing files and the number of contained analysis input groups. We describe these variations—called *layer delta*—in the analyses by computing the similarity of all analyses within one layer based on the number of incoming and outgoing files and the number of contained analysis input groups. To make them comparable, we calculate the difference of each analysis to the analysis with the earliest execution time (within the same layer). Figure 5.3 illustrates the result of the motif-based aggregation from analysis nodes (AL2) to the layer aggregation level (AL3), which yields an even higher compression.

5.4.2. User-Interest-Driven Expansion

Based on a modular DOI, we expand regions that are of particular interest to the user. Our interest-driven expansion corresponds to an unbalanced drill-down [EF10] and enables the user to investigate nodes at lower aggregation levels, while keeping the overall graph as context to analyze the driving changes in the development of a study (e.g., workflow executions, recurring executions, and changes).

Each node has a DOI value assigned that reflects the current level of interest and controls its aggregation. We identified and implemented five components that contribute to the DOI

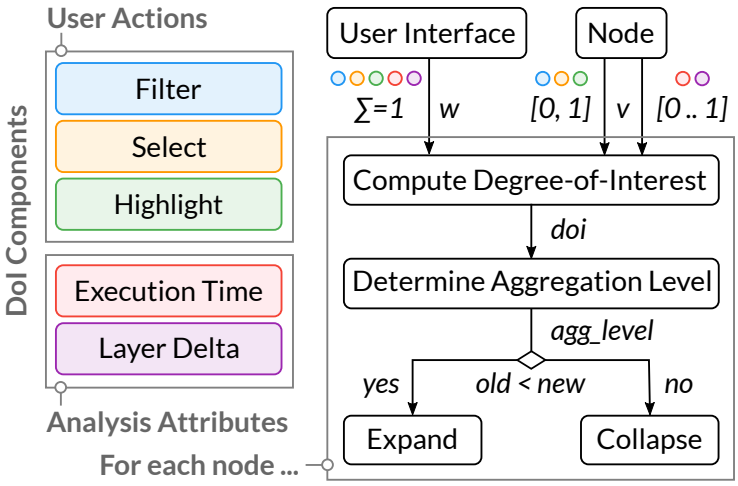


Figure 5.4.: The DOI value for each node incorporates the user-driven weight w and value v for all DOI components. Based on this value, the aggregation level of the node is selected, which determines the compression or expansion of the node.

of a node, grouped into user actions and analysis attributes. The components *filter* (part of the facet-browsing interface, see Section 5.4.4), *highlight* (when hovering over a node), and *selected* (when clicking a node) are driven by the user, whereas *workflow execution time* and *layer delta* are based on node attributes (see Figure 5.4). Each component consists of a weight, configured in a user interface, and the value provided by the node itself. The modular DOI function integrates all DOI components into a single DOI value using the following equation:

$$doi(node) = \sum_{i=1}^{n} w_i \times v_i \;\Big|\; \sum_{i=1}^{n} w_i = 1.$$

The DOI value for a given node is equal to the sum of all component weights times the component values. The applied weights can be defined freely by the user, where n is the total number of DOI components, w are the weights of the components, the sum of which is one, and v_i is the attribute value. The values for the user actions are binary, which means that the corresponding DOI component value is set to 1 (active = highlighted, selected, or passing a filter) or 0 (inactive). For the values of analysis attributes, we use a continuous and normalized scale ranging from 0 to 1.

In the next step, the DOI value drives the selection of the aggregation level of the node (see Figure 5.4). We partition the DOI range into increasing aggregation levels. When nodes switch aggregation levels due to user interaction, we use animated expand and collapse transitions, preserving the mental map of the user. Note that we can also *extract* analyses (AL2) with a high DOI value from a layer node (AL3) *without expanding* them, as shown in the layer, far right in Figure 5.2. With this extraction technique, analysis paths that

are currently not of interest to the user can be hidden, while attached to the layer remain visible.

We visualize all DOI components as a stacked bar chart (see Figure 5.2(b)), which allows the user to manipulate the weight of each component interactively (see Chapter 3). When the weights are adjusted, the other components adapt their weights proportionally to ensure a sum of 1. Moreover, we allow the user to work with only a single DOI component by activating or deactivating them individually. Applying a modified DOI function results in an update of the provenance graph representation.

5.4.3. Visual Encoding

The data provenance graph contains a series of node attributes that need to be encoded effectively (T2). Depending on the aggregation level, we create a different node glyph.

Node Type Figure 5.5(a) illustrates the different node types. Nodes in AL0 are visualized as primitive shapes, for instance, diamonds for tools, squares for raw input data, and circles for files. We visually group the workflow using a bounding box with a semi-transparent background that corresponds to the nodes of higher aggregation levels. Nodes at the level between AL1 and AL3 are drawn as rectangles including a unique icon representing the aggregation level (e.g., a cogwheel for analysis input groups) and the number of aggregated child nodes. In the case of layers (AL3), we add the total number of child nodes to indicate possible expansions. For better distinction between analyses (AL2) and layers (AL3), we add a dashed outline to the layer bounding box to indicate the aggregation.

Age We vary the brightness of the nodes to encode execution date and time of analyses at all aggregation levels (see Figure 5.6), addressing T4. White represents the earliest and black the most recent analysis execution time. As layers contain analyses that were created at different time steps, we use a black-to-white gradient to encode the distribution of execution times within the layer node.

Change We present the layer delta as defined in Section 5.4.1 to address T4 by adding an asterisk—as shown in Figure 5.2(a)—to the layer node (AL3) in which the similarity value is greater than zero.

Attribute Information By applying semantic zooming, we reduce the number of text labels based on the space available. For the aggregated nodes of AL1 to AL3, we show the workflow name next to the bounding box, together with the number of child nodes and the number of incoming and outgoing edges. For workflow instances in AL0, the displayed attribute value (e.g., tissue, factor, or file name) is selectable by the user.

5.4.4. Interaction Techniques

In AVOCADO, the user can navigate between the aggregation levels—drill-down (T3) and roll-up (T1)—for each node individually or use the buttons in the toolbar (see Figure 5.1(b)) that directly set all nodes to the selected aggregation level. In addition, the user can search or filter nodes and highlight specific paths to understand causality. We use animated transitions for all operations to provide visual continuity and maintain the mental map of the user.

Filtering We provide two user interfaces for filtering nodes by attribute value and time (T6). A facet-browsing interface limits the number of nodes on the basis of the attributes that they contain, such as tissue, drug, or cell type. In addition analyses can be filtered by analysis execution time using an interactive timeline, as illustrated in Figure 5.6. The date and time is mapped to the x-axis of the timeline, while the analysis input group count is mapped to the y-axis. The background corresponds to the time gradient (see Section 5.4.3). The user can move filtering sliders to adjust the time range. Hovering over an analysis highlights the corresponding node in the provenance graph view.

All filter operations result in more available screen space for regions with high interest. Nodes that are affected by a filter can be either shown with reduced opacity or hidden, that is, removed from the graph, as illustrated in Figure 5.6(b) and (c). Hide operations require a recomputation of the whole layout. We apply animations to transition to the new layout.

Path Highlighting Enabling the user to investigate the steps that led to an analysis result is an important task (T5). For each graph node we provide the selection of the predecessors (i.e., the path leading to the selected node), and the successors (i.e., all nodes derived from the selection). The path is highlighted by changing the color of the edges (see Figure 5.2). Additionally, the user can apply the DOI function to expand all nodes along the highlighted path, while keeping the remaining nodes aggregated.

Note that the primary purpose of our interactive AVOCADO visualization is to enable analysts to *recall* already executed multi-step analysis workflows. Although, the presentation of findings is not a primary goal of AVOCADO, it is supported indirectly via static screenshots that can be shared with others.

5.5. Implementation

AVOCADO is implemented in JavaScript, jQuery, and D3 [BOH11b] and handles initialization, layout computation, motif-based compression, and rendering. We use the *Dagre* [4] JavaScript library to compute layered graph layouts for data provenance graphs with hundreds of nodes. *Dagre* implements the 2-layer crossing minimization [JM97] to reduce the number of edge crossings. Additionally, we compute the order of analysis input groups (AL1) using a barycentric heuristic. We also employ horizontal coordinate assignment [BK02] to balance the speed of computation with layout aesthetics. The layout adapts dynamically to user actions (e.g., filtering, collapse/expand) and DOI changes on a per-node basis. Changing the DOI of any node causes the weighted sum to be recomputed, which results in automatic adjustment of the aggregation level of the corresponding analysis. Our motif-based compression algorithm uses a single breadth-first traversal to discover motifs in the topology-sorted graph, as explained in Section 5.4.1. Analyses are added in a repeated traversal to layers on the basis of their preceding motif sequence and the motif itself. With this additional aggregation constraint, we avoid layering analyses based solely on their workflow template without considering provenance in preceding layers. We then calculate and normalize the numeric change metrics of every layer. The assembled data provenance graph is stored in a hierarchical data model where all aggregation levels inherit from a generic node object. Finally, the graph and the filter components (i.e., timeline and DOI view) are rendered in SVG using D3 [BOH11b].

We have integrated the data provenance graph visualization, multiple support views, and the user interface into the dataset browser of Refinery. Our visualization acquires datasets and their provenance graph via an internal RESTful API in JSON format.

5.6. Usage Scenario

We demonstrate the functionality and effectiveness of the AVOCADO approach by describing how it can be applied in a typical scenario in which an analyst needs to recall, review, and interpret data provenance for a study that was conducted by a collaborator (see Supplementary Video).

5.6.1. Data

In our example, we used data provenance information from a *simulated* epigenomics study. Epigenomics is the study of naturally occuring biochemical modifications of the genome

[4] `https://github.com/dagrejs/dagre/`

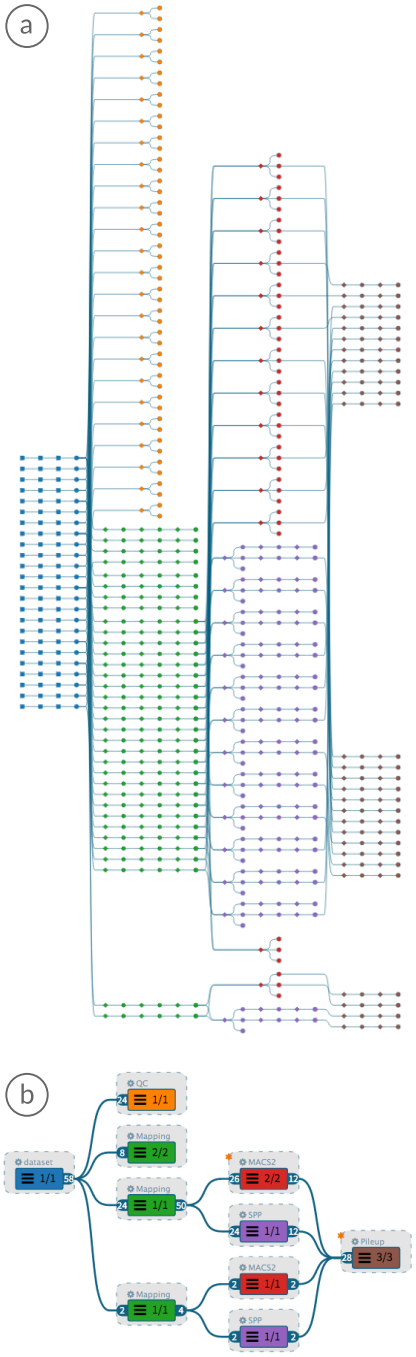


Figure 5.5.: Overview of the data provenance graph of the analysis. Colors represent workflow instances *QC* (orange), *Mapping* (green), *MACS2* (red), SPP (purple), and *Pileup* (brown). (a) Fully expanded graph showing tool and file nodes. (b) Graph aggregated to level AL3.

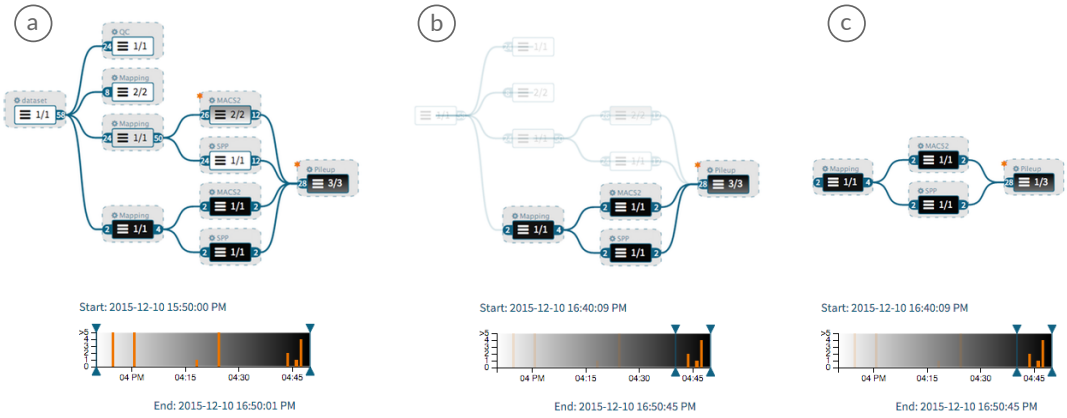


Figure 5.6.: Attribute mapping and time-based filtering. Darker nodes were created more recently.
(a) data provenance graph aggregated to level AL3 with node color representing time. No filtering applied. (b) Time filter applied in *blend* mode. (c) Time filter applied in *hide* mode.

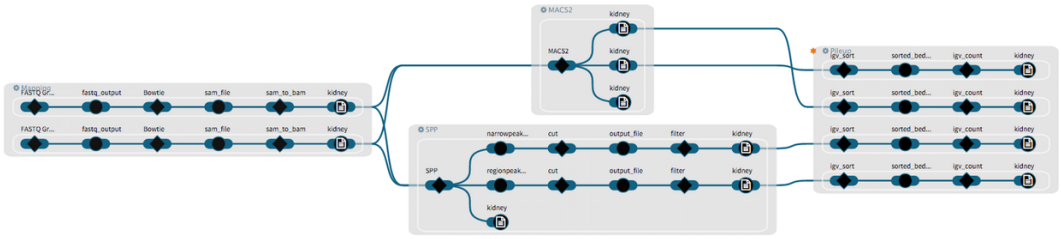


Figure 5.7.: The filtered subgraph from Figure 5.6(c) is expanded to the workflow instance level (AL0) with the *tissue* attribute mapped to the output nodes ("kidney").

that influence gene regulation and many other essential cellular processes. The prevalent technology for studying these modifications is called ChIP-seq (Chromatin Immuno-Precipitation Sequencing) [Par09], which is based on high-throughput sequencing. In our example, several workflows were applied to raw ChIP-seq data in order to identify differences in the distribution of two modifications called *H3K27ac* and *H3K4me3* along the mouse genome in two different tissues (*kidney* and *liver*) after treatment with one of three different drugs (*Alpha*, *Beta*, *Mock*). This resulted in a total of 12 combinations of experimental factors (2 modifications $\times$ 2 tissues $\times$ 3 treatments), and for each combination data from two replicates was obtained, thus yielding a total of 24 input files for this study.

We applied five different state-of-the-art workflows in this study (see Supplementary Figures A.4-A.8), implementing a typical ChIP-seq analysis approach as described by Park [Par09]. The specific workflows are (1) *QC*: a quality control workflow to evaluate the quality of the input data. The output is a report. (2) *Mapping*: a workflow that maps the sequencing reads in the input data to the genome sequence. (3) *MACS2* and (4) *SPP*: are peak calling tools that identify locations within the genome to which a larger number of sequencing reads have been mapped than elsewhere in the genome. (5) *Pileup*: This is a workflow that prepares the data for visualization in a genome browser tool.

Due to the computational effort and cost that would be required to execute these workflows on real data, we modified the workflows as follows: (1) We replaced each tool in the workflows with corresponding dummy tools that only passed on any input files that they received. (2) Instead of real data we used small text files. All tools and workflows [5] (see Figure 5.7) and the metadata and data files used to run the analyses in Refinery [6] are available on Github.

Although these analyses were simulated at the tool level, *the simulation had no effect on the size and properties of the data provenance*, which allows us to demonstrate the capabilities of AVOCADO using the resulting graph.

5.6.2. Data Provenance Exploration

First (see Supplementary Video), the analyst wants to gain an understanding of the structure of the study (T1). While the fully expanded data provenance graph (Figure 5.5(a)) only hints a hint at the structure of the study, AVOCADO provides a more compact and task-appropriate representation at the highest aggregation level (AL3) (Figure 5.5(b)). Based on the colors of the aggregate nodes and their labels, the analyst can quickly review which workflows were used and how often. Furthermore, the aggregation of all redundancies in the graph used in level AL3 still provides information about the order in which workflows were chained together. Next, the analyst wants to understand in which order

[5] https://github.com/refinery-platform/galaxy-tools/tree/develop/simulation/
[6] https://github.com/refinery-platform/sample-data/tree/develop/avocado_usage-scenario/

the collaborator ran the analyses, and switches the color mapping to the time encoding that maps analysis execution time stamps (T2) to a unidirectional linear color map (see Figure 5.6(a)). In this view, the analyst discovers that a part of the analysis was conducted very recently, while the rest was performed much earlier. The analyst then applies the timeline filter (T6, see Figure 5.6(b)) to limit the graph visualization to the most recent analyses, and switches filtering into "hide" mode (see Figure 5.6(c)). The analyst then drills down into the most recent set of analyses to the workflow instance level (T3, see Figure 5.7). Using the attribute mapper, the analyst maps the tissue attribute onto the output files (T2) and observes that the most recent analyses were conducted only on *kidney* samples. By hovering over the output nodes of the *MACS2* and *SPP* workflows, the analyst reviews additional attributes of these nodes and finds that both the narrow peak and the region peak files of these tools were compressed with the *Pileup* workflow. The analyst returns to the AL3 level and decides to investigate a layer node that has a change indicator (T4) and contains analyses that were executed at different time points (see Figure A.1(a)). After drilling down into the analyses (see Figure A.1(b) and (c)), the investigator finds that the collaborator re-ran the *MACS2* workflow on a pair of files but that the results were not processed any further. Following this observation, the analyst decides to investigate, which raw data is associated with these files in order to understand why they were not processed. By switching the DOI function to always expanding selected paths to the highest level of detail, the analyst selects the inputs of the *MACS2* workflow and traces them back to the raw data (T5, see Figure A.2(a) and (b)). Once the raw data files have been identified, the analyst then applies the reverse functionality and traces all results generated from one of the raw data files. The analyst observes that this file was processed by both the *MACS2* and *SPP* workflows and that results of both tools were also processed with the *Pileup* workflow (see Figure A.3).

This usage scenario illustrates how AVOCADO can be applied in a typical analysis session to address the tasks we defined together with domain experts and based on the literature (see Section 5.3). In order to evaluate the AVOCADO approach for a wider spectrum of scenarios and to refine it so it can handle also more extreme cases, a comprehensive user study will have to be conducted, as outlined in Section 5.8.

5.7. Discussion

Graph Layout The choice of layout algorithm is a crucial factor in the perception of a graph visualization. We use a grid-based approach as described in Section 5.4. As in a spreadsheet, the largest node (e.g., an expanded workflow) defines the width for the remaining cells in the same column, which results in long edges to nodes in adjacent columns. Other layout approaches might create a more compact layout (e.g., [YDG+15]), but must also consider the characteristics of the data provenance graph (see Section 5.2) and the interactive requirements to ensure fluid transitions when changing hierarchy levels.

Since edge crossings decrease the readability of the graph, they are to be minimized. We achieve this by re-ordering the nodes in a post-processing step after creating the initial layout. A further aspect is layout stability. In AVOCADO, we initially compute a stable layout, as explained in Section 5.5. However, interactions such as drill-down/roll-up and DOI expansion change the size of one or multiple nodes and therefore trigger a re-computation of the whole layout. This is neither resource- nor time-efficient. In future work, this will be improved by recomputing only the parts of the graph that have changed together with their neighboring context to make space in the layout.

Scalability The AVOCADO approach relies on the hierarchical structure of the provenance graph to achieve compression across the three aggregation levels AL1 through AL3. In order to scale, AVOCADO requires each level to have fewer nodes than the level below, which is generally the case for biomedical workflows. To each aggregation level the same scalability considerations as for general node link diagrams apply. The current implementation performs well on typical biomedical workflows that have around 1,000 nodes at AL0, but technical limitations lead to decreased response times when the provenance graph grows beyond this size.

Degree of Interest Function The effect of a multi-component DOI value can be hard to interpret. In practice, we observe that users tend to distribute the weights equally or select only a single component at a time. A set of predefined configurations for different tasks (auto-expand selected paths, fully expand clicked nodes, etc.) would simplify the DOI interface but requires further user testing.

5.8. Summary

We have presented AVOCADO—an approach to visualizing workflow-derived data provenance graphs. We visualize the multi-attribute time-dependent graph as a node-link diagram. To reduce the size of the graph, we apply a combination of hierarchical and motif-based graph aggregation. We interactively expand parts of the graph using a modular DOI function that is based on graph attributes and user-driven actions. By means of a usage scenario we have demonstrated how our technique can help analysts to gain a deeper understanding of complex multi-step analyses.

6 | KnowledgePearls Using Focus+Context in Provenance-Based Visualization Retrieval

Contents

Recording user interactions and visualization states of a visual analysis sessions results in a provenance graph that can be used as knowledge base with large potential for recalling previous results and guiding users in future analyses. However, without extensive manual creation of meta information and annotations by the users, search and retrieval of analysis states can become tedious.

We present *KnowledgePearls*, a solution for efficient retrieval of these analysis states using an attribute-driven DOI function. As a core component, we describe a visual interface for querying and exploring analysis states based on their similarity to a partial definition of a requested analysis state. Depending on the use case, this definition may be provided explicitly by the user by formulating a search query or inferred from given reference states. We explain our approach using the example of efficient retrieval of demographic analyses by Hans Rosling and discuss our implementation for a fast look-up of previous states. Our approach is independent of the underlying visualization framework. We discuss the applicability for visualizations which are based on the declarative grammar *Vega* and we use a Vega-based implementation of *Gapminder* as guiding example. We additionally present a

biomedical case study to illustrate how *KnowledgePearls* facilitates the exploration process by recalling states from earlier analyses.

6.1. Introduction

Visual exploration is a time-consuming and complex process. A single analysis session can consist of hundreds of individual steps. Over the past two years, we developed—together with our collaboration partners at a pharmaceutical company—a data-driven visual analysis system for identifying and prioritizing drug targets. All interactions within a session are tracked in a provenance graph. Capturing sessions in provenance graphs not only ensures reproducibility of findings, but also provides a growing knowledge base of the overall analysis state of the explored datasets. Further, visual exploration is usually not performed by a single domain expert alone but by a team in which each expert analyzes the dataset individually. To avoid repetitive analyses that do not lead to new findings and to increase the confidence in potential findings when identified by multiple experts independently, effective recall of captured provenance information becomes crucial. In addition to retrieval purposes, the growing knowledge base can be used to guide future analysis sessions by identifying unexplored areas of the exploration landscape. In this chapter, we present a novel search and retrieval concept for visualization states stored in a provenance graph.

A visualization state is a snapshot of properties, including data properties, visual properties, and interaction properties as attribute-value pairs, at a certain juncture during analysis. The composition of properties depends on application, visualization types used, and interactions supported. Changing one or multiple properties creates a new visualization state that is pushed into the provenance graph.

Ragan et al. [RESC15] characterize provenance information by *type* and *purpose*. According to their organizational framework, our approach operates on *interaction provenance*, which contains the history of user interactions with the system, and the tightly coupled *visualization provenance*, which comprises the history of visualization states. *KnowledgePearls* is designed to allow users to *recall* states visited in the analysis history. In addition, we see the purpose of our work extending to user guidance.

Our primary contribution is a solution for efficient retrieval of visual analysis states that are structured as provenance graphs of automatically recorded user interactions and visualizations. As a secondary contribution, we propose guidelines and requirements for existing visualization systems to support provenance tracking and retrieval and explain the integration of *KnowledgePearls* into a dashboard defined in the *Vega* visualization grammar [SRHH16].

At a glance, *KnowledgePearls* allows users to specify a search query either by definition (i.e., data attributes, selected items) or by example (i.e., from an active visualization state). All query aspects can be weighted based on user interest. To provide more meaningful search results, matching subsequent visualization states are grouped into state sequences (resembling a pearl necklace). We argue that this approach has the potential to minimize redundant analyses and to accelerate the process of analyzing the data.

We introduce *KnowledgePearls* and its integration using a *Gapminder*-inspired prototype implemented in *Vega* as guiding example. The loaded provenance graph contains all interactions Hans Rosling performed with the original *Gapminder* software in three different presentations [1][2][3]. Additionally, we demonstrate the scalability and effectiveness of our solution in a case study carried out by collaborators working in the field of cancer drug discovery.

6.2. Design Objectives

Based on literature reviews, interviews with our domain experts, and our own experience with provenance information in the context of visual analysis, we elicited a set of design objectives that an effective provenance-based retrieval approach should support.

O1 Support Heterogeneous Properties. A visualization state can contain properties with different data types, for instance, numerical or categorical. The retrieval approach must take this heterogeneity of data types into account when looking for matching visualization states.

O2 Support Fuzzy Search. Most provenance retrieval approaches support binary search that considers exact matches only. This reduces the number of search results tremendously, but has the disadvantage of displaying only results of equal importance. However, users might be interested in search results that match the formulated query only partially. The output of such a fuzzy search is a collection of search results with varying importance.

O3 Consider Inherent Temporal Coherence. Each interaction leads to a new visualization state in the provenance graph. Due to temporal coherence, adjacent states of the same interaction sequence are typically very similar. In the case of a plain retrieval, all these similar states would be listed as separate search results, cluttering the result list. To alleviate this scalability problem, the retrieval approach should support appropriate aggregation and clutter reduction techniques.

[1] Presentation by Hans Rosling, The Best Stats You've Ever Seen, 2007.
[2] Presentation by Hans Rosling, 200 Countries, 200 Years, 2010.
[3] Presentation by Hans Rosling, Religions and Babies, 2012.

With these objectives in mind we developed *KnowledgePearls*, a novel provenance-based retrieval process.

6.3. Provenance-Based Retrieval Approach

KnowledgePearls is designed as an extension to visual analysis tools that record user interactions in a provenance graph, with the goal to allow users to effectively query previous exploration states. With our approach, users can rapidly switch between two different modes, as illustrated in Figure 6.1: (1) **visual analysis mode** and (2) **retrieval mode**. In visual analysis mode (Figure 6.1a), new visualization states are continuously created with every user interaction and stored in a provenance graph. In retrieval mode (see Figure 6.1b), users can formulate search queries that are compared with all visualization states stored in the provenance graph. Matching states are grouped, ranked, and presented in a result list.

Users can switch seamlessly between these modes, that is, start a retrieval during an analysis and then continue the visual exploration from a selected search result.

6.3.1. Provenance Graph and Visualization States

When interacting with a visual analysis system, users trigger actions such as attribute changes, updates of filter settings, or item selections. Possible interactions in our *Gapminder*-inspired prototype are, for instance, choosing the data attributes mapped to the axes of the scatterplot, determining the projection type of the map, selecting countries shown as items in the plot, or determining the time point for which the scatterplot shows data.

The structure of the provenance graph as used in *KnowledgePearls* is based on the definition presented by Gratzl et. al [GLG+16]. Each user interaction triggered in the visualization results in a transformation of the visual representation. After updating the visualization, a snapshot of the selected attributes, filters, and items is automatically captured as a visualization state. Both entities—the visualization state and the actions—are stored in a provenance graph that consists of visualization states as nodes and the corresponding actions as edges (see Figure 6.1a).

The information stored in the provenance graph can be used to restore any previously seen visualization state by applying all actions from the root of the provenance graph to the desired state. Branches in the graph emerge when users jump back to a previous state in order to continue the exploration from there. This back-tracking strategy is typical for exploratory data analysis scenarios.

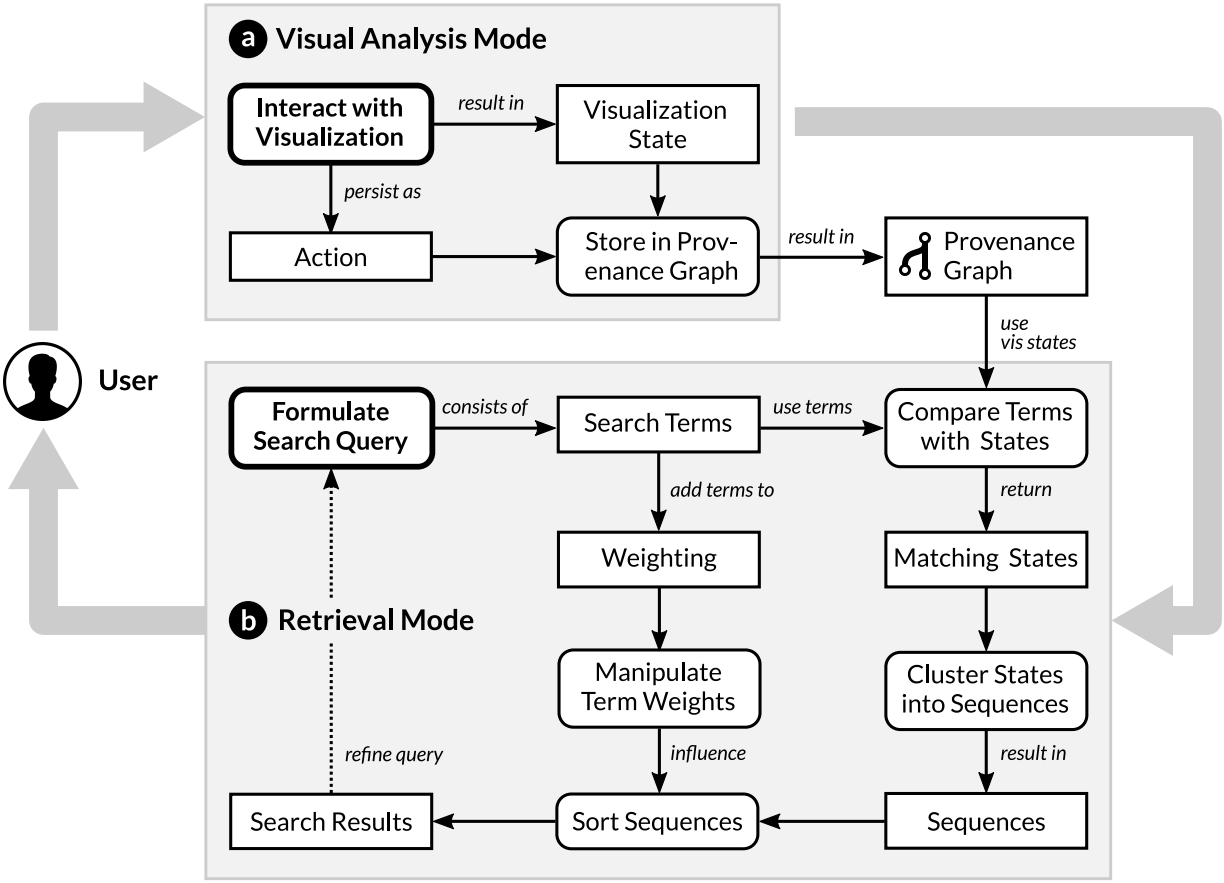


Figure 6.1.: The user can rapidly switch between two different modes: (a) User interactions change the visualization, and result in new visualization states, which are added to the provenance graph. (b) The user formulates a search query that is compared with the stored visualization states. Matching states are grouped into sequences and sorted according to the weighting of the search terms.

Visualization State Properties

A visualization state consists of a title, metadata (e.g., creator and creation date), and multiple properties describing the visualization state (**O1**).

Choosing visualization properties to be stored is a non-trivial task, as they need to be tailored to the visualization (system) at hand. While the set of tracked properties will typically vary between domains, tasks, and visualization techniques, we provide a high-level semantic characterization of property types for guiding the process of defining meaningful properties for a specific system. Properties of the visualization state can coarsely be classified as data-related or visualization-related.

Data-related properties refer to aspects of the inspected data which are independent of the visual encoding. Important examples include:

- Data attributes which are mapped to a visual variable, e.g., color or an axis. For example, users may search for states where the attribute *GDP* is displayed.
- Data items which are represented by the visualization so that distinct items can be discriminated from each other. This may include data categories for data sets with nominal data attributes. For systems supporting data selection, an important specific type of property is the set of data items in focus. For example, users could search for states where the country *United States* is selected.

Visualization-related properties are derived from the current visual encoding and thus depend on the visualization technique. Important sub-types include:

- The visualization technique itself. For example, in systems which offer different visual encoding options (e.g., *Tableau*), users could search for states that contain a map or stacked bars.
- Visualization parameters which are set by the user. For example, searching for *logarithmic* could retrieve states where data is visualized by a logarithmic scale.
- Visualization metrics which quantify aspects of the visual encoding of the particular data. Important examples include *Scagnostics* [WAG05] in case of scatterplots or *Pargnostics* [DK10] for parallel coordinates. For more information about quality metrics, we refer to Bertini et al. [BTK11], who provide an overview and a systematization of their use in high-dimensional data visualization. In general, the selection of supported quality metrics depends on the visualization techniques, the data, and the task. From the perspective of our approach, the search based on quality metrics requires the visualization system to compute these metrics for (some of) the views of the current analysis state. The provenance graph needs to store quality metrics much like other numerical properties. In the Gapminder example, a user could, e.g., search for states with high values for the property *skinniness*.

Our *Gapminder* example includes data attributes and selected data items as data-related properties as well as visualization parameters and *Scagnostics* as visualization-related properties.

Relationships Between Properties

In our concept, each property of a visualization state is treated independently. However, depending on the visualization type, relationships between properties may exist, for instance, that both a data attribute and an axis scale define an axis in a scatterplot. In this

version of our concept, we ignore potential relationships in favor of an easy-to-use search interface that can cover most of the use cases.

In early stages of our concept, we experimented with the definition of relationships between properties. An ontology is a formal way to describe the relationships between possible terms (e.g., that an axis contains attributes and a scale). Creating such an ontology that can be used for visualization states is an open research challenge, which is beyond the scope of this work. In response to early feedback from our target users to first versions of our prototype implementation, we deliberately decided to favor simplicity over being able to express more complex relationships between query terms.

For a detailed discussion of relationships between properties and hierarchical property structures, see Section 6.7.

6.3.2. Retrieval

In retrieval mode (Figure 6.1b), we utilize the captured provenance data for finding similar visualization states for a given search query. *KnowledgePearls* supports users in their current analysis in recalling earlier states, which facilitates collaboration between users. A retrieval can be performed either by the user who did the analysis in an earlier session or by a different user who searches, for instance, for similar visualization states in a provenance graph created by a colleague.

Retrieval starts with formulation of a search query that consists of one or multiple search terms. A search term can be either a string (e.g., *France*) or a property identifier followed by a numerical value (e.g., *monotonic = 0.3*) (**O1**).

Once the user has entered the search query, each search term is compared with the whole collection of visualization states stored in the provenance graph. The comparison step determines the relevance of a visualization state compared to the given search query. We use different comparison mechanisms based on the properties' data types: categorical, numerical, and set-typed. The result is a fuzzy search that presents the visualization states found according to a continuous spectrum from high to low relevance (**O2**).

Categorical properties are, for instance, displayed data attributes (e.g., *GDP*, *population*) and categorical visualization settings, such as the map project type (e.g., *mercator*, *orthographic*). To calculate the similarity between categorical properties, we index the property values and apply the *term frequency–inverse document frequency* (*tf–idf*) [SWY75]. This measure is widely used in information retrieval and reflects the importance of a word in a document (in our case in a visualization state) with respect to the whole collection of documents (i.e., provenance graph). A word is more important when the term frequency (in the given visualization state) is high and the document frequency of the term in the whole collection of documents is low. Thus, we decrease the importance

of commonly used values of categorical properties. For instance, in all recorded stories, Hans Rosling used the *population* of a country as the size of a visual mark in *Gapminder*. In contrast, he used the attribute *child mortality* in a single session only. Consequently searching for *child mortality* results in a higher similarity score in matching states than the search term *population*.

For **numerical properties**, such as the selected *year* in *Gapminder* and derived visualization metrics, we calculate the absolute difference between the query value and the state value. The smaller the difference between a numerical input value and the actual state value, the greater the importance of this property. For example, given two scatterplots with the selected years *2006* and *2003*, a search for *2005* would result in greater similarity to states from the *2006* set than to those from the *2003* set because the difference is only one year in the former case and two years in the latter.

Set-typed properties typically refer to selections of data items, for example, a set of brushed countries. Depending on the visualization type, multiple set-typed properties can exist in cases in which users can select multiple different entity types, such as countries or continents. We use the *Jaccard Index* to determine the similarity between two sets, that is, all set-typed input values and the set-typed properties of the visualization state. The higher the overlap between the compared sets, the more similar they are. In our *Gapminder* example, Hans Rosling selected *United States* along with *Vietnam* to demonstrate the economic differences in 1964. In another presentation, he selected *United States* together with eight other countries. A search for *United States* would rank the first visualization state with the selection of only two selected countries higher than the second one.

The comparison step results in a *relevance score* within the range $[0, 1]$ for each search term, where 1 denotes an exact match and 0 a non-matching search term. We normalize the tf–idf value for categorical search terms in that range since the tf–idf value can go beyond 1. Subsequently, the relevance of the state is calculated as the weighted sum of all relevance scores. The applied weights can be changed interactively by users according to their interest in each search term. This ensures that states of high interest also result in a higher rank. States that are equal to a value of 0 are considered to be non-matching and are excluded from further processing steps. Setting a higher threshold value to exclude less relevant states is also possible.

6.3.3. Grouping of Search Results

Since several visualization properties remain unchanged between multiple subsequent visualization states, displaying all matching ones individually results in a long list of highly similar search results (**O3**). To address this issue, we group visualization states into *state sequences*.

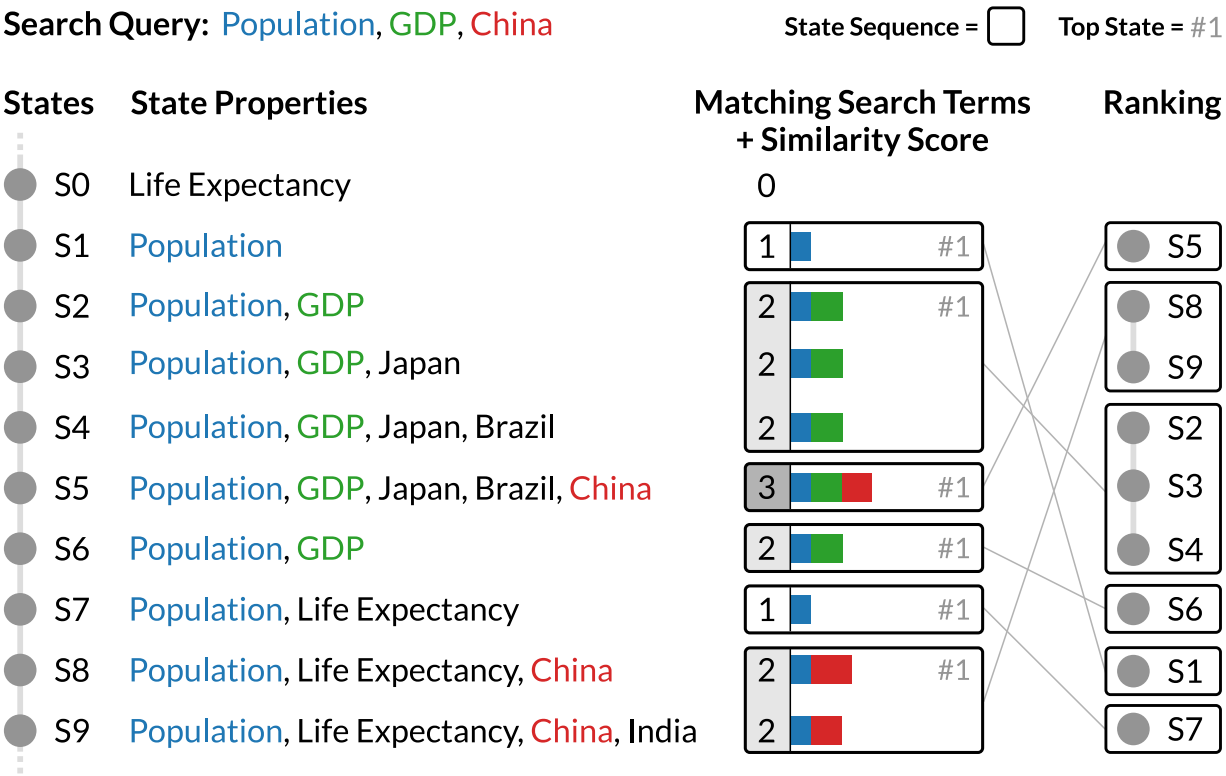


Figure 6.2.: A list of states, including their properties, is compared to a search query. We calculate the matching search terms for each state, and group subsequent states into state sequences. The ranking is defined by the number of matching terms and the weighted similarity score.

Our grouping algorithm determines matching terms, i.e., terms with a *relevance score* > 0, for every visualization state, and groups subsequent visualization states with an equal number of matching terms into sequences. States that have no matching search terms are excluded. This approach tends to create multiple short sequences, as shown in Figure 6.2. For instance, searching for *population* only results in one sequence containing states between *S1* and *S9* . Adding *GDP* as a second search term splits the sequence into three sequences. Adding *China* as a third search term results in six sequences, four of which contain only a single state.

In addition to the grouping, we identify a *top state* for each sequence. Although the number of matching search terms remains constant for all states of a sequence, the similarity score for each state may vary within a sequence because of the different comparison mechanisms that depend, for instance, on the remaining property values in the state (see Section 6.3.2). We consider the state with the highest similarity score, or in the case of multiple candidates the first candidate (see *S2* to *S4* in Figure 6.2), in a sequence as the top state. The top state is used as a representative of the sequence within the search result.

We utilize the top results to order the sequences by the weighted similarity score. This yields a ranking in which relevant sequences—those that match multiple search terms—are ranked more highly. For instance, in Figure 6.2, state *S5*, which matches the whole search query, is ranked the highest, followed by the sequence with states *S8* and *S9*, which have slightly better similarity scores than other sequences matching two search terms.

In summary, the grouping algorithm provides a trade-off between presenting individual relevant states (i.e., maximizing the number of search results) and longer sequences (i.e. minimizing the number of search results).

6.4. Visualization and User Interaction

Our prototype implementation consists of three views, as shown in Figure 6.3: The **application view** (a), the **provenance graph side panel** (b), and the **search side panel** (c). The application view contains the visualization system (e.g., *Gapminder*), with which users interact in visual analysis mode (see Figure 6.1). On the right side, users can open the provenance graph and the search side panel on demand. The provenance graph side panel (labeled "Current Session History" in our prototype) provides a visualization of all recorded states [GLG+16] (see Figure 6.3b). Interactions from the application view, which are added to the provenance graph, instantly appear in this side panel (Figure 6.3c). The user can jump back to previous states and continue the analysis from there. The search side panel is linked with the provenance graph side panel and contains a search field (Figure 6.4a), selected search terms as query (d), a weighting editor (e), and a list of search results (f).

Below, we explain individual views and their functionality in more detail. As a guiding example, we use a provenance graph that is based on selected presentations given by Hans Rosling. We focus on the parts of the presentations in which he mentioned the development of *Japan* and the *United States* at the end of World War II in *1945*.

6.4.1. Search Field and Weighting Editor

Users can formulate the search query by entering search terms in the input field shown at the top of the side panel (see Figure 6.4a). Upon entering the first term (e.g., *Japan*), a drop-down list of suggestions is presented below the input field. The list contains only property values that occur in the provenance graph and thus have been involved in the exploration. Suggestions are grouped by property names (e.g., *data attributes*, *Scagnostics*) and their possible values (e.g., *GDP*, *China*, *density = ?*) (**O1**). We allow users to search for property names and values, and highlight the matching part of the string. In addition, users can search for metadata, such as author and timestamp of creation of a visualization

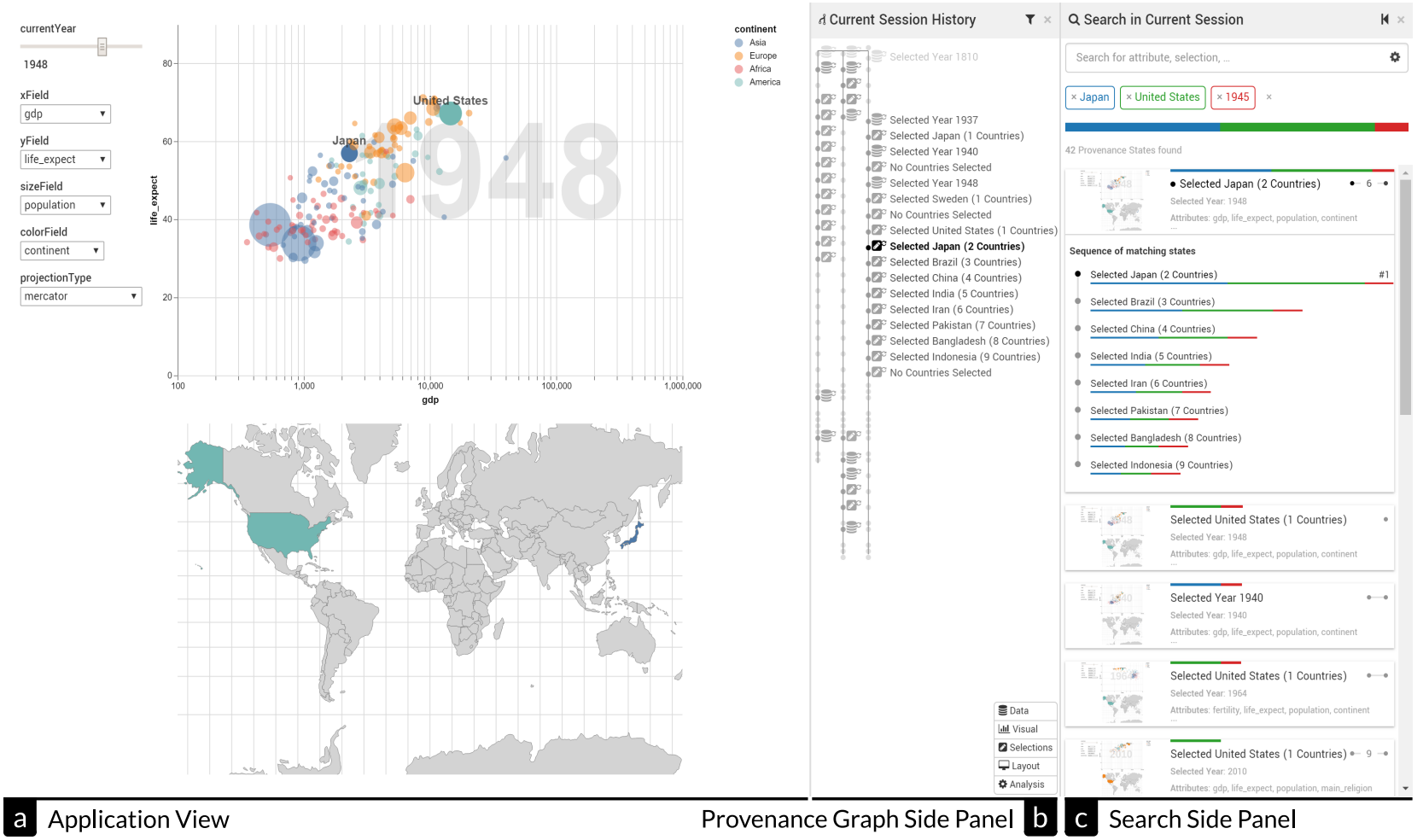


Figure 6.3.: Our *Gapminder*-inspired prototype implemented in *Vega* (a) with a captured provenance graph (b) consisting of three stories from Hans Rosling's presentations. Users can query the provenance graph for visualization states using the search side panel (c).

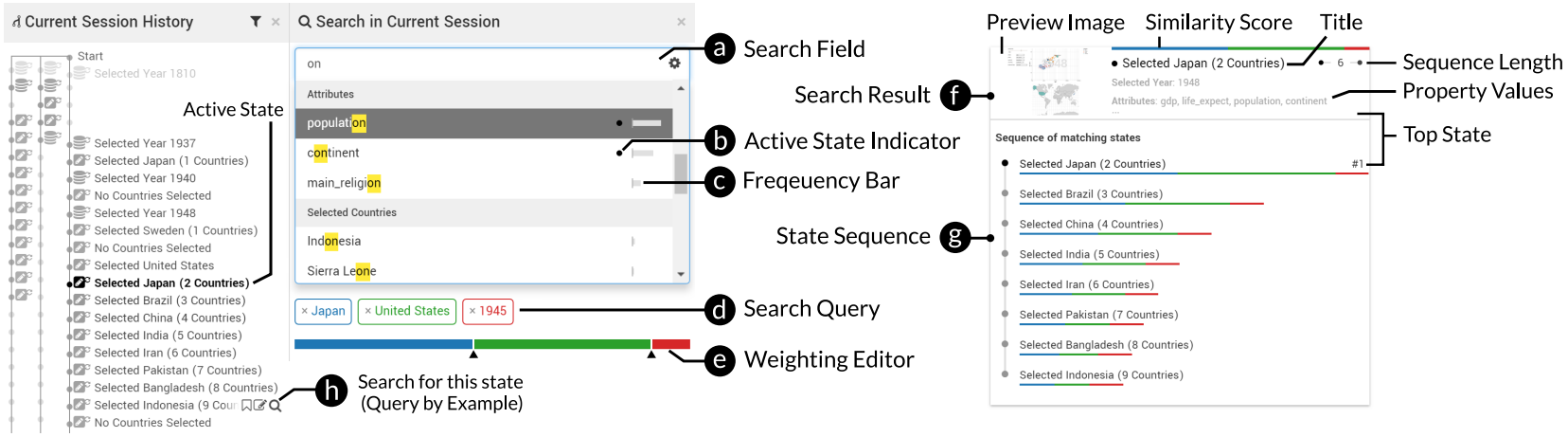


Figure 6.4.: Components of the retrieval interface. The search field (a) suggests visualization properties while typing. Properties of the active state are indicated by a black circle (b) and provide a frequency bar (c). The selected properties are added to the search query (d) and can be weighted based on the user's interest (e). The search results (f) are ranked by the state's similarity score and can be expanded to access the state sequence (g). The top state is used as search result cover and indicated in the state sequence. The active state is indicated in the sequence length glyph and in the state sequence. Users can also formulate a query with parts of an existing state (h).

state. If property values require further input, for instance, the reference value for the *Scagnostics* measures, we indicate the input type (e.g., numerical) and validate the input before accepting the search term (**O2**).

Taking the provenance graph created from Hans Rosling's presentations as basis and entering the string "Ja" for the search term *Japan* will show a drop-down list with only three possible countries—*Japan*, *Azerbaijan*, and *Jamaica*. From this filtered list, *Japan* is the only country that is active and can be selected. The other two countries do not feature in the provenance graph, and are therefore disabled to prevent a worthless, empty search result list. By default, we only list countries that are contained in the provenance graph at least once. Countries that are defined in the dataset but are not in the provenance graph can be optionally added to the suggestions as context.

For the string "United" entered as part of the second search term *United States*, all three suggestions—*United States*, *United Kingdom*, and *United Arab Emirates*—are active and can be selected. The countries are sorted in descending order by their frequency (i.e., the number of occurrences in all recorded visualization states). We indicate the frequency by the length of a bar shown next to the property value (see Figure 6.4c). In our guiding example, the retrieval returns 24 visualization states for *United States*, 2 states for *United Kingdom*, and 2 states for *United Arab Emirates*.

Derived Properties

We show the ten most frequent terms and property values in a *Top 10* group at the top of the result list. Without the need for a specific search term to be entered, the *Top 10* group provides an immediate summary of the most used properties in the provenance graph. In our guiding example, the most frequent items are presented in decreasing order of their frequency.

Similar to the *Top 10* results that are derived from the whole provenance graph, we also use all properties from the current list of search results as a subset, and assemble a list of related property values that can be used to refine the search query and narrow down the scope.

Query by Example

Aside from entering explicit search terms, users can use their current analysis state in the application view to identify similar states, which is referred to as query by example. To simplify the identification of items that are present in the currently shown visualization state, for example, when analyzing the country development in *1945*, they are marked with a black circle in the suggestion list (see Figure 6.4b). However, these properties might be scattered throughout the whole list. As a solution, we provide a setting that filters the list for marked properties only.

The three search terms (*Japan*, *United States*, and *1945*) that have been added to the search query are displayed as color-coded words and as a stacked bar below the search field (see Figure 6.4d). The user can remove individual search terms from the query or clear the entire query. Modifying the search query triggers comparison and updates the search result list (see Figure 6.1).

Weighting Editor

The selected search terms, which can be seen as multiple attributes of a ranking of visualization states, can be weighted according to user interest by using a dedicated weight editing interface (see Figure 6.4e). By default, the weights are equally distributed across all search terms. In our example, the user wants to emphasize the two countries *Japan* and *United States* and reduce the impact of the year *1945*. To achieve this, the user can distribute the weight by dragging the sliders of the stacked bar, as in *LineUp* [GLG+13] and *ThermalPlot* (see Chapter 3). Changing the weights triggers a recalculation of the similarity scores and updates the order of the search result. For the remainder of this example, we set the following weights: 45% for *Japan*, 45% for *United States*, and 10% for *1945*.

6.4.2. Search Results

We present matching visualization states in decreasing order according to their similarity scores (see Figure 6.3). As explained in Section 6.3.3, individual states are grouped into state sequences to guarantee temporal coherence across states (**O3**). Searching for the terms *Japan*, *United States*, and *1945* in our example, 41 states with a significance score greater than zero, which will be grouped into 16 state sequences. To further increase the readability of the results, only the top state of each sequence is displayed as a representative. However, users can reveal the whole sequence on demand (see Figure 6.4g).

Each search result box follows the same structure as shown in Figure 6.4f. The preview image provides a quick overview of all search results and support users in interpreting the somewhat abstract visualization state definition. This may be particularly helpful when comparing different search results.

The stacked bar at the top of a search result encodes how much each search term (i.e., the weighted relevance score) contributes to the state's similarity score. In our guiding example, the first search result matches all three search terms. To find out why some percents are missing, users can hover over the stacked bar to see more details about the distribution of the similarity score in a tool tip.

We show the property values grouped by property name as comma-separated list. In our example, the search terms *Japan* and *United States* are not visible immediately since the list exceeds the given space of the search result box. By hovering over the list, users can expand the box to see all property values and check that *Japan* and *United States* are exact matches, since they are highlighted. The selected year *1948* is not highlighted, because it does not match the search term *1945* by three years and thus explains the few missing percentages in the similarity score of the first result.

In addition to gaining an overview of the contained property values, users can explore the state sequence in detail. In the upper right corner of a search result, a glyph indicates the length of the state sequence. In the case of the search result that best matches our given search terms, the sequence represents eight states. Depending on the sequence length, the visual representation features one to three connected circles. For sequences with more than three states, we show the first and last states and indicate the remaining number of states in the center (see Figure 6.4f). If a state sequence contains the active visualization state, the circle or number is highlighted in black.

Clicking the glyph reveals the whole state sequence. All matching states are aligned from top to bottom and contain the state title and the weighted relevance score of the matching search terms as a stacked bar. We add an additional *#1* indicator to the top state of the sequence to point out that this state has the highest similarity with respect to the search query in the sequence and is already presented in greater detail above the sequence list as the representative state of this state sequence.

In our guiding example, the first sequence item was captured when *Japan* was selected as second country. The *United States* must have been selected as first country, since the title of subsequent states indicates that several other countries were selected. Moreover, the sequence shows that the similarity score for *Japan* and *United States* decreases for further selected countries in subsequent states (see Section 6.3.2). Hence, the first state of the sequence is indicated as the top state.

When the user hovers over a sequence item, the corresponding state is highlighted in the provenance graph view. Selecting an item loads the state in the visualization view. Likewise, when the user hovers over a search result box, the sequence of states is highlighted in the provenance view. Selecting a search result box loads the top state of the sequence. For the users' convenience, *KnowledgePearls* remembers the last active visualization state before loading a selected search result in the visualization view. This allows users to easily restore the visualization state in which they started their search.

6.4.3. Provenance Graph

Next to the search side panel the user can optionally open the provenance graph as a second side panel (see Figure 6.3b) containing a visualization of all recorded states as described in [GLG+16]. Both side panels are linked, that is, the active visualization state and mouse hover are highlighted in both panels.

In order to provide an alternative to querying a state by a given example, we place a search button next to the state in the provenance view (see Figure 6.4h). Clicking the search button adds a dynamic property group that contains all values of the selected state to the top of the suggestion list of the search input field. Users can select properties of this state or filter the suggestions as previously explained (see Section 6.4.1).

We further support the user's search by enhancing the visibility of matching states in the graph visualization. For matching states, we increase the degree of interest, which causes expansion of the corresponding representations and collapses non-matching ones (see Figure 6.3b). We encode the similarity score in the opacity of a state representation, which makes it easier for users to focus on relevant states.

In our example, all matching states are enhanced when hovering over the first sequence, which matched all three search terms. The highlighted states confirm that the first country selected was *United States* and that *Japan* was selected as the second country.

6.5. Implementation

To demonstrate how existing visualization applications can be extended with *Knowledge-Pearls* retrieval capabilities, we implemented two prototypes: the first one is a *Gapminder*-inspired visualization implemented in *Vega* (see Figure 6.3) and the second one is our *Ordino* drug target discovery tool [SGS+19] that our collaborator used for the case study (see Section 6.6 and Figure 6.6).

Both prototype systems consist of three main building blocks (see Figure 6.5): (a) the application view containing the actual visualization(s), (b) the provenance tracking and provenance visualization component, and (c) the retrieval component.

The provenance and retrieval components are implemented in TypeScript using the *Phovea* platform [4]. The code is open source and available on Github [5]. The *Vega Gapminder* prototype is deployed at `https://vega-gapminder.caleydoapp.org` and the *Ordino* application is available at `https://ordino-retrieval.caleydoapp.org/`.

[4] `https://phovea.caleydo.org/`
[5] `https://github.com/Caleydo/knowledge-pearls/`

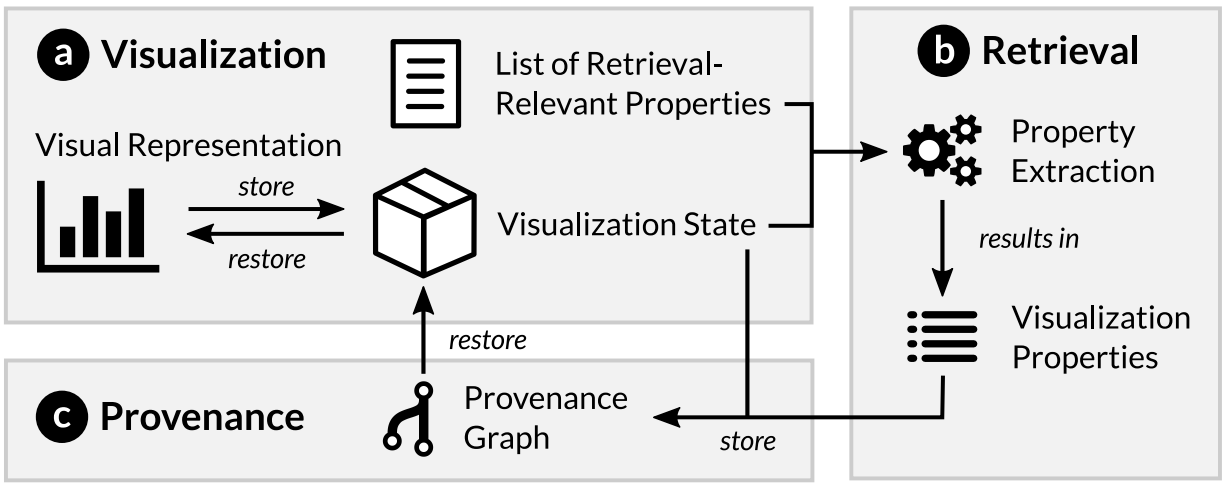


Figure 6.5.: Integration overview. The visualization component (a) can persist the representation as visualization state and provides a list of retrieval-relevant properties. The retrieval component (b) extracts the visualization properties from the state. The visualization properties and visualization state are then stored as new node in the provenance graph (c). When a user selects a node, the visualization state is pushed backed into the visualization and the representation is restored.

The provenance tracking and visualization component uses the *CLUE* provenance graph implementation by Gratzl et. al. [GLG+16]. In the course of this work, we extended the *CLUE* approach by capturing and storing visualization states in the provenance graph, since the original approach captures only the action leading to a state but not the state itself.

We build the initial search index when loading the provenance graph into the browser and update it incrementally with every new user interaction. The retrieval for the currently loaded provenance graph is performed on the client side.

6.5.1. Integration Guidelines

While our two prototypes demonstrate how *KnowledgePearls* can be connected to existing visualization systems, we also want to provide concrete guidelines how this can be achieved for other visualizations.

A system needs to fulfill two important requirements to be compatible with *KnowledgePearls* (see Figure 6.5). First, it needs to be able to store and restore a visualization state. Second, the system needs to provide a description that instructs *KnowledgePearls* which retrieval-relevant properties need to be extracted from the visualization state and

stored in the provenance graph. This includes, among other aspects, the computation of visualization quality metrics. Transferring this step to *KnowledgePearls* is in general not possible because it depends on the type and purpose of the visualization, and requires access to the data at a level that is opaque to *KnowledgePearls*. An interesting idea for future work is to incorporate approaches for computing quality metrics solely on the image results generated by the visualization system. However, this is a non-trivial topic and beyond the scope of this chapter.

6.5.2. Vega Integration

The visualization in our *Gapminder*-inspired prototype is declared as *Vega JSON specification* and rendered using the *Vega* library [6].

As *Vega* comes with the built-in functionality to store and restore visualization states, it fulfills the first requirement. To meet the second requirement, we minimally extend the *Vega* specification with the two properties `track` and `search` that mark the declaration parts that are relevant for the retrieval (see Listing B.1). As these extensions operate on the interaction level, no extensions on the dataset, or individual data item level are necessary.

Further, we need to add event listeners to the *Gapminder* visualization to receive notifications when users change an attribute, select items, or choose a different year. In *Vega*, events are declared in the form of signals, which can release further actions, as for instance, update dependent signals or the visual representation itself. We employ this mechanism to check if the `track` property is contained in the declaration of the triggered signal. If this is the case, we start the property extraction in the retrieval component (see Figure 6.5b). To be able to distinguish between different signal sources, the developer needs to declare a title and icon for nodes, which will be displayed in the provenance graph view (see Figure 6.3).

The `search` property configures the creation of the visualization properties that will be offered to the user in the search side panel (see Figure 6.5b). The declaration must contain the property data type (categorical, numerical, or set), a custom title that is used as label for the search terms and suggestions, and a group label for the suggestions (see Section 6.4.1).

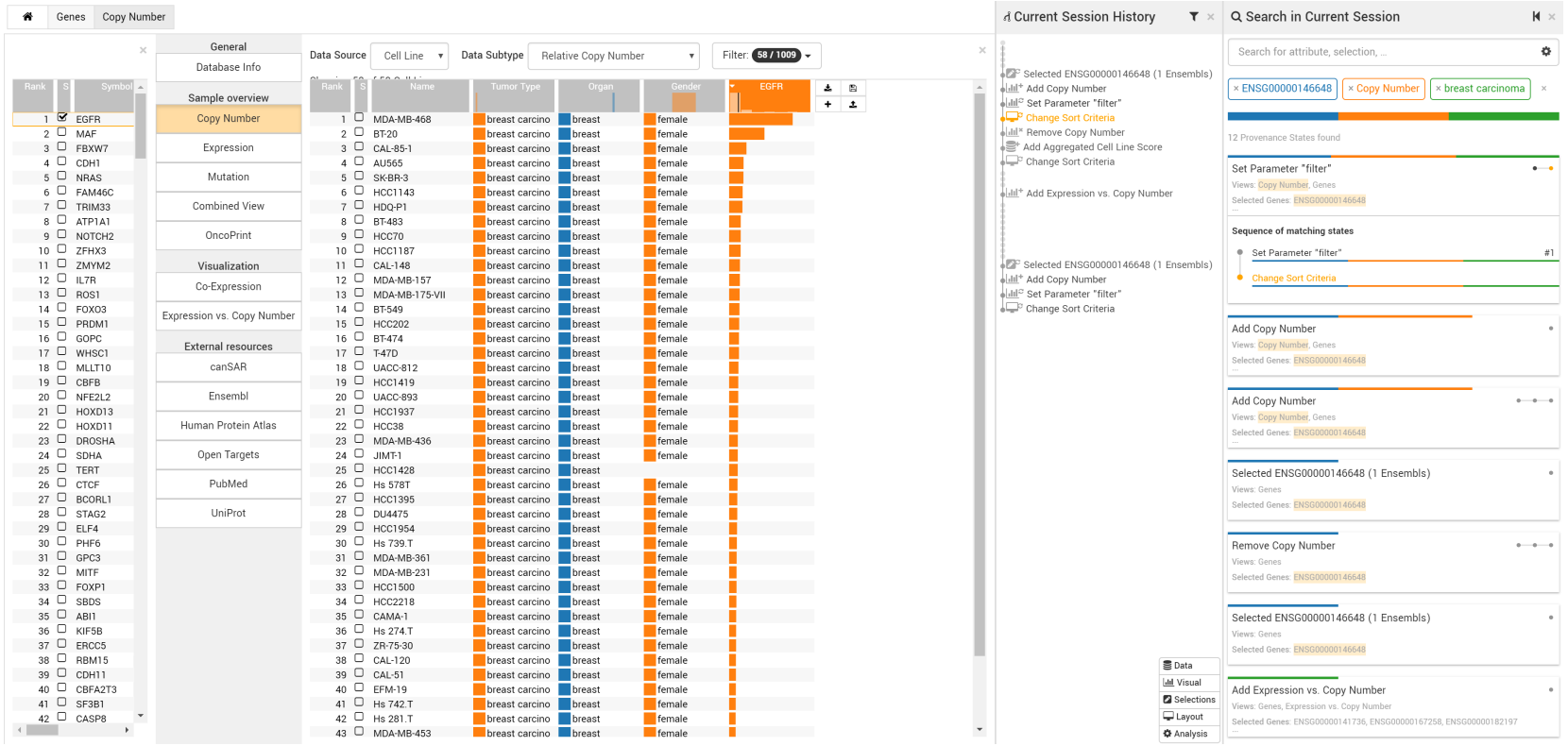


Figure 6.6.: The user has entered three search terms to find similar state sequences and to recall a previously recorded analysis. The search query results in seven state sequences, with the first sequence matching all search terms. Jumping to this state shows a ranking of breast cancer cell lines with copy number information for the gene *EGFR*. The user continues the analysis for *MDA-MB-468*, the cell line with the highest copy number.

6.6. Case Study

We demonstrate the value and utility of the *KnowledgePearls* approach by integrating it into *Ordino* [SGS+19], a web-based discovery tool that allows users to flexibly rank and explore genes, cell lines, and tissue samples (see Figure 6.6).

In *Ordino*, the user starts by selecting or defining a list of items (e.g., genes) that will be opened in a multi-attribute ranking view based on *LineUp* [GLG+13]. Once the user has selected one or multiple items in the ranked list, a collection of possible follow-up detail views for exploring the current selection is displayed. This detail view can be either another ranking, a view with additional information about the selection, or a visualization based on the selected items (e.g., a scatterplot matrix). *Ordino* follows a focus+context approach where the focus view is shown on the right and the previous focus view is shown as context on the left. New views are pushed from the right to the list of open views. Users are able to close the view and horizontally scroll back to previous views at any time. We extended *Ordino* with *KnowledgePearls* capabilities by tracking the shown views, parameter settings, and item selections as visualization states and store them in the built-in provenance graph. The set of properties that define a visualization state has been conjointly determined with our collaborators, based on what they consider as relevant for recalling previous analysis states and results.

The case study summarizes analysis sessions carried out by a collaborator working in the field of cancer research. Some time ago, the scientist performed several analyses regarding various cancer cell lines—cultured cells that are derived from tumors and that can proliferate indefinitely in the laboratory—and cancer genes (see Figures B.1-B.9 in supplementary material [7]). Now, the user comes back to that analysis session in order to find certain analysis steps and to continue with them. First, the user wants to know which previous analysis states are similar to the very last one from the session.

Ordino opens the last state at which the user left off as active state and shows a list of lung cancer cell lines ranked by the copy number of the gene *EGFR*. Using *EGFR* as search term results in two state sequences (see Figure B.10). When hovering with the mouse over the search results, the corresponding state chains are highlighted in the provenance graph view.

The user refines the search by using *Copy Number* from the related suggestions in the drop-down menu as second search term, because he remembers having inspected this attribute in detail in the previous analysis. This time, the search returns five sequences, where the first two sequences match both search terms (see Figure B.11).

[6] `https://vega.github.io/vega/`
[7] `https://knowledge-pearls.caleydo.org/`

The user knows that the initial analysis was not about lung cancer, as currently visible in the focus view, but he cannot remember the other tumor type. Entering "tum" for tumor type shows exactly two suggestions: *non-small-cell lung cancer* and *breast carcinoma*. Selecting *breast carcinoma* as the third search term results in seven state sequences. Exactly one sequence matches all search terms (see Figure B.12). Jumping to the last state of this sequence shows a ranking for breast cancer cell lines with an additional column for *EGFR* copy number (see Figure 6.6).

The user has a new idea and wants to know whether the cell line *NCI-H2170* has already been used at some earlier point in the analysis. Searching for this cell line returns exactly one state sequence (see Figure B.13). Jumping to the top result of the sequence reveals that the cell line was selected in a scatterplot in the *Expression vs. Copy Number* detail view. The user can seamlessly continue the analysis (see Figure 6.1) by opening a detail view, which provides further information about this cell line (see Figure B.14). In conclusion, the *KnowledgePearls* integration supported the user in recalling previous work and making use of this knowledge in the context of a new analysis.

Informal User Feedback

We evaluated the *KnowledgePearls* integration into the *Ordino* platform with our collaborators by means of two thinking aloud sessions, in which we observed them while using the system.

Our collaborator pointed out multiple times that simplicity of the interface is more important than being able to express more complicated queries. We used this feedback as a guiding principle throughout all phases of the design process.

Furthermore, the collaborator valued the seamless switch between analysis and retrieval mode, and the flexibility in combining and weighting search terms to find specific analysis steps within a large provenance graph. He stated that the current solution with the search field and search term suggestions is highly useful and makes it easier to find contextually relevant previous states.

Our collaborator reported that he was slightly confused by the output of the grouping algorithm (see Section 6.3.3), which splits long sequences into smaller sequences for multiple search terms. We plan to address this feedback in the next iteration of our prototype.

The *Ordino* drug discovery system extended with *KnowledgePearls* retrieval capabilities will soon be in productive use by a few dozen domain experts form multiple disciplines—including biology, cancer genomics, and bioinformatics. The active use of the tool will result in a quickly growing provenance graph. In the case study described above, our collaborator was operating on the provenance graph that he created as a single user in multiple analysis sessions over time. However, he stressed that *KnowledgePearls* will be

particularly valuable in the future when the provenance graph contains visualization states from exploration sessions done by his colleagues working on various projects. To use the full potential of *KnowledgePearls* for collaborative scenarios, we plan to extend the tool with additional annotation and filtering capabilities.

6.7. Discussion and Limitations

6.7.1. Generalizability

Many applications do not provide a visualization state and hence, do not fulfill the integration requirements (see Section 6.5.1). In this case an image of the visualization can be captured subsequent to every interaction. The images can be processed using computer vision and machine learning algorithms to extract the properties, such as axis labels or data items [SKC+11, JKS+17, PH17, PMH18]. The extracted properties can form a visualization state and/or list of retrieval-relevant properties that are served as input for our retrieval approach (see Figure 6.5b).

In case the application cannot restore a visualization state, users are unable to switch back from retrieval mode into visual analysis mode (see Figure 6.1) and must manually recover the state based on the given property information. Recovering visualizations automatically for a given visualization state (e.g., using reinforcement learning) remains an open research topic.

6.7.2. Relationship of Visualization Properties

Besides the relationships between properties of a visualization, multiple coordinated view (MCV) setups add relationships across visualizations. In our *Gapminder*-inspired prototype a world map is linked to a scatterplot, i.e., country selections are updated in both visualizations accordingly. For the retrieval we consider property values from both visualizations. As discussed above, by treating the search terms independently users cannot specify in a query that a property needs to be present in a particular view that is part of the MCV setup.

In early stages of our concept, we explored a tree-like structure for organizing and structuring visualization properties such that the root node represents the MCV setup itself and its children represent the individual visualizations. However, increasing the expressiveness of the visualization property setup increases the complexity of the query formulation. Besides MCV setups in which different visualization techniques show the same data, setups that use the same visualization technique for different data are challenging. Typical examples are scatterplot matrices and parallel coordinate plots. In both, the number of

instances (scatterplot or axis) is variable, as it depends on the number of attributes in the explored dataset. Managing such a variable set of visualizations and querying specific subsets remain open topics for future research.

Further, the current concept does not consider logical operators other than the *AND* combination, which we use by default to combine the individual query terms. A *NOT* operator, for example, could be valuable for expressing that a certain state should not contain a given search term. Integrating advanced logical operators could lead to new kinds of retrieval goals in which not the presence but the absence of certain visualization properties is desired. For instance, users could penalize the presence of a well-known and frequently used gene to filter out commonly explored cases.

6.7.3. Graph Retrieval

We designed *KnowledgePearls* for retrieving similar states from a recorded provenance graph based on a user-defined search query. However, due to the graph structure and its inherently contained metadata, such as date of creation of individual states (**O3**), more advanced queries and retrieval techniques could be applied to the graph itself. Motif-based search to query specific action sequences could be a valuable addition. Applications include cases in which users remember states that led to a certain insight rather than the state that contains the insight itself.

In addition, motif-based search and graph retrieval could be used to extract knowledge about the analysis process itself. Identifying and retrieving repetitive patterns in the provenance graph can be useful for various purposes, such as detecting common analysis patterns across users. These identified patterns could then be used to provide guidance systems that support the user throughout the analysis by suggesting common action sequences based on the current one. However, how to guide users without restricting them in the exploration process remains an open research question.

6.7.4. Scalability

Real world analysis sessions can quickly result in a provenance graph with many nodes and branches. A branch is introduced when a user jumps back to a previous state and continues the analysis from that point. In *KnowledgePearls* we improve the scalability by identifying similar visualization states and grouping them into sequences (**O3**). However, if the same or a highly similar state sequence is found in multiple branches, each sequence is displayed as separate search result. As part of future work, we plan to apply semantics-based, motif-based, and hierarchical aggregation strategies [MSOI$^+$02, AS06, SLSG16, EF10].

Besides the visual scalability of the provenance graph visualization and retrieval interface, also the computational and storage scalability play a major role for an effective provenance

retrieval solution. Both are influenced by two factors: (1) the number of states in the graph and (2) the number of attributes stored in each visualization state (**O1**). In the current prototype the whole graph containing all attributes for each state are stored in a database on the server and transferred to the client. This approach works for small provenance graphs with up to a few hundred states, which are typically generated by a single user in a consecutive exploration session. In such scenarios, computing the similarities between the search query and individual states can be executed on the client side, due to the negligible computational overhead. However, when additionally integrating provenance graphs from exploration sessions created by other users, different measures need to be taken, such as computing the similarity score on the server and transferring only the relevant states to the client. However, a trade-off exists between computing the scores on the server and the flexibility to let the user weight individual components, as the latter would require a re-computation of each score upon change. To address this issue, we plan to investigate hashing strategies.

6.8. Summary

We have presented *KnowledgePearls*, an approach to searching effectively for visualization states in provenance graphs. An intuitive visual interface enables users to query and explore previous analysis states based on a definition that can be explicitly formulated or implicitly inferred from a given reference state. As a key aspect of our work, the visualization and the used metrics support a quantitative notion of similarity. This allows for a gradual ordering of states by their relevance and enables users to express interest by assigning weights to different elements of the search definition. A case study carried out by collaborators in the field of cancer drug discovery illustrated how *KnowledgePearls* facilitates the exploration process by recalling states from earlier analyses.

7 | Conclusion

Contents

7.1. Discussion and Future Work

This thesis has described several Focus+Context approaches and techniques for time-series and provenance data, which lead to a variety of possible future work directions.

Scalability An essential aim of Focus+Context techniques is to enable the visual analysis of large data sets. Yet, the scalability of the approaches depends significantly on two factors: interaction and abstraction. On one hand, more information can be displayed by selecting several focus points. This, however, can lead to visual clutter, since several entities must be displayed at a higher LOD with the same available screen size. On the other hand (and a possible solution for the visual clutter), the number of ALs, both in the data space (aggregation hierarchy) and in the visual space (LODs), can be increased. This allows entities to be more abstracted and occupy less visual space. Numerous related works for abstraction and aggregation of time-series data [SAAF18, ACG14, AMST11] and dynamic graphs [Arc09, MSOI+02, AS06] can be found. However, with an increasing number of ALs, it becomes more complex for users to navigate between the individual levels [EF10]. Hence, a balance between interaction and abstraction must be found. New abstraction methods and interaction techniques can increase the scalability of visual analysis in the future.

Interpretability of DOI Values DOI functions are usually black boxes to the user and contradict the needs of interactive data exploration. Interpreting the computed DOI value is challenging for users, even when only a few input values are combined. Moreover, when considering the temporal dimension in the DOI computation, the interpretation becomes even harder for users, as the entities' DOI value might vary over time. In *ThermalPlot*,

we address this problem by integrating a DOI streamgraph (see Section 3.4.4) for selected items that explains how each attribute contributes to the final DOI value over time. In *KnowledgePearls*, we encode the contribution of each attribute to the overall DOI value as a stacked bar in the search result and highlight matching search terms (see Section 6.4.2). In contrast to the attribute-based DOI functions, results from topology-based DOI functions are harder to explain. For example, in a cloud-based network, as introduced in Chapter 4, we extract the shortest path between two selected nodes from the graph and visualize all intermediate nodes as an inlay. However, in scenarios that go beyond shortest path extraction, capabilities for analyzing and evaluating alternative paths, as presented in Pathfinder [PGS+16], should be considered in the future.

Analysis of Temporal Patterns The *ThermalPlot* technique, as introduced in Chapter 3, allows monitoring multi-attribute time-series data by combining short- and long-term value developments into a salient visual representation. Animation and trajectories can help users to analyze how an item developed over time. Additionally, dedicated support for pattern search could be a fertile area for future work. Apart from patterns generated by the path of an item's position, value patterns over time such as "down-up-down" are not specifically supported yet but are useful in several application domains. Consequently, better support for pattern search and exploration is on important step towards a comprehensive tool for the analysis and exploration of multi-attribute time-series data.

Prediction and Uncertainty Visual analytics approaches for time-series data, such as *CloudGazer* (see Chapter 4) and *ThermalPlot* (see Chapter 3), are often designed for the visual exploration of historical and streaming data. A possible direction for future work is the integration of forecasting algorithms for generating predictions. In *ThermalPlot*, we could indicate the future position of items and expand the trajectories we currently use for showing the items' development over time. In *CloudGazer*, methods from predictive analytics could be applied to the provenance information of a cloud-based network, which would enable system administrators to pinpoint potential future bottlenecks. We presented a first concept for the analysis of future performance predictions in Section 4.7. In a next step, the analytical processing results could also be used to make suggestions, such as how and when the topology or the mapping between components should be optimized and what the implications of the changes would be. As forecasted values become increasingly uncertain for longer periods of time, different ways to encode the introduced uncertainty must be taken into account [SSSE16] and must be carefully evaluated [HQC+19].

Provenance-Based Guidance The storage of large amounts of provenance information constitutes a valuable knowledge base that can be used to assist users in their further analysis. The next analysis steps are suggested based on the history and can potentially lead to new insights. Alternatively, the history can be used to warn users about repetitive

analysis steps. In both cases, the system suggests possible actions and targets that users can focus on next. However, depending on the type of provenance information used as an input [RESC15], the suggested next steps might differ. Furthermore, the users' prior knowledge and degree of guidance (i.e., orienting, directing, or prescribing) [CGM+17] must be considered when implementing provenance-based guidance.

Combination of Various Provenance Types The provenance-based approaches *AVOCADO* (see Chapter 5) and *KnowledgePearls* (see Chapter 6) focus on a single type of provenance information. Extending these approaches to multiple types of provenance [RESC15] would provide a more holistic understanding of analysis processes and could be used to improve future visual analysis tools. For *AVOCADO*, for instance, the additional recording of interaction provenance, might support analysts in understanding why analyses have been modified and relaunched. Furthermore, a deeper integration into tools that capture and manage data provenance information, such as the *Refinery Platform* [1], will be a step closer towards actionable provenance. This will enable analysts to replicate previous analysis results and continue particular analyses with a new or different dataset or parameters directly from the visualization. Analysts will not only be able to explore provenance information but also create new data provenance in a single visual interface. In the case of *KnowledgePearls*, it would be interesting to extend our approach with comprehensive meta-analyses capabilities, which might allow visualization designers to better understand the nonlinearity and backtracking nature of visual analysis. New approaches that can handle even larger amounts of provenance data and allow the analysis of connections between the different types of provenance graphs are an open research challenge as well.

7.2. Conclusion

This dissertation presented several contributions for analyzing time-series and provenance data using Focus+Context techniques. The solutions utilize modular DOI functions that are driven by one or multiple data attributes in *ThermalPlot* and *KnowledgePearls*, the topology of the graph in *CloudGazer*, or a combination of both in *AVOCADO*. We demonstrate the scalability and usefulness of these approaches in different case studies from the domains of cloud computing, finance, and biomedical research. For the future, we expect that the amount of data in these and other domains will continue to grow. Our presented Focus+Context solutions can then be the basis for further research leading to new approaches for analyzing larger data and generating new insights.

[1] `http://refinery-platform.org`

Bibliography

[ABJF06] Ilkay Altintas, Oscar Barney, and Efrat Jaeger-Frank. Provenance Collection Support in the Kepler Scientific Workflow System. In *Provenance and Annotation of Data*, volume 4145, pages 118–132. Springer, 2006.

[ACG14] Danielle Albers, Michael Correll, and Michael Gleicher. Task-Driven Evaluation of Aggregation in Time Series Visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*, pages 551–560. ACM Press, 2014.

[Ahl96] Christopher Ahlberg. Spotfire: An Information Exploration Environment. *ACM SIGMOD Record*, 25(4):25–29, 1996.

[AHSS13] James Abello, Steffen Hadlak, Heidrun Schumann, and H. Schulz. A Modular Degree-of-Interest Specification for the Visual Analysis of Large Dynamic Networks. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '13)*, 20(3):337–350, 2013.

[AHSS14] James Abello, Steffen Hadlak, Heidrun Schumann, and Hans-Jorg Schulz. A Modular Degree-of-Interest Specification for the Visual Analysis of Large Dynamic Networks. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '13)*, 20(3):337–350, 2014.

[AKK96] Mihael Ankerst, Daniel A Keim, and Hans-Peter Kriegel. Circle Segments: A Technique for Visually Exploring Large Multidimensional Data Sets. In *Proceedings of the IEEE Conference on Visualization (Vis '96)*, 1996.

[AMST11] Wolfgang Aigner, Silvia Miksch, Heidrun Schumann, and Christian Tominski. *Visualization of time-oriented data*. Springer, 2011.

[Arc09] Daniel Archambault. Structural Differences Between Two Graphs Through Hierarchies. In *Proceedings of the IEEE Conference on Graphics Interface (GI '09)*, pages 87–94. Canadian Information Processing Society, 2009.

[AS06] Tero Aittokallio and Benno Schwikowski. Graph-based Methods for Analysing Networks in Cell Biology. *Briefings in Bioinformatics*, 7(3):243–

255, 2006.

[BBDW14] Fabian Beck, Michael Burch, Stephan Diehl, and Daniel Weiskopf. The state of the art in visualizing dynamic graphs. In *Proceedings of the Eurographics Conference on Visualization (EuroVis '14) – State of The Art Reports*, 2014.

[BCBDH08] Olivier Biton, Sarah Cohen-Boulakia, Susan B. Davidson, and Carmem S. Hara. Querying and Managing Provenance Through User Views in Scientific Workflows. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE '08)*, pages 1072–1081. IEEE, 2008.

[BCS+05] L. Bavoil, S.P. Callahan, C.E. Scheidegger, H.T. Vo, P.J. Crossno, C.T. Silva, and J. Freire. VisTrails: Enabling Interactive Multiple-View Visualizations. In *Proceedings of the IEEE Conference on Visualization (VIS '05)*, pages 135–142. IEEE, 2005.

[BE12] C. Glenn Begley and Lee M. Ellis. Drug development: Raise standards for preclinical cancer research. *Nature*, 483(7391):531–533, 2012.

[Ber10] Jacques Bertin. *Semiology of Graphics: Diagrams, Networks, Maps*. ESRI Press, 2010. First published in French in 1967.

[BI15] C. Glenn Begley and John PA Ioannidis. Reproducibility in Science Improving the Standard for Basic and Preclinical Research. *Circulation research*, 116(1):116–126, 2015.

[BK02] U. Brandes and B. Köpf. Fast and Simple Horizontal Coordinate Assignment. In P. Mutzel, M. Jünger, and S. Leipert, editors, *Graph Drawing*, number 2265 in Lecture Notes in Computer Science, pages 31–44. Springer Berlin Heidelberg, 2002.

[BM13] M. Brehmer and T. Munzner. A Multi-Level Typology of Abstract Visualization Tasks. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '13)*, 19(12):2376–2385, 2013.

[BOH11a] M. Bostock, V. Ogievetsky, and J. Heer. D3: Data-driven documents. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '11)*, 17(12):2301–2309, 2011.

[BOH11b] M. Bostock, V. Ogievetsky, and J. Heer. D3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '11)*, 17(12):2301–2309, 2011.

[BTK11] Enrico Bertini, Andrada Tatu, and Daniel Keim. Quality Metrics in High-Dimensional Data Visualization: An Overview and Systematization. *IEEE*

Transactions on Visualization and Computer Graphics, 17(12):2203–2212, 2011.

[Buc15] Stuart Buck. Solving reproducibility. *Science*, 348(6242):1403–1403, 2015.

[BvLH+11] Sebastian Bremm, Tatiana von Landesberger, Martin Hess, Tobias Schreck, Philipp Weil, and K. Hamacherk. Interactive visual comparison of multiple trees. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST '11)*, pages 31–40. IEEE, 2011.

[BW08] L. Byron and M. Wattenberg. Stacked Graphs - Geometry & Aesthetics. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '08)*, 14(6):1245 –1252, 2008.

[BYB+13] M. Borkin, C. Yeh, M. Boyd, P. Macko, K. Gajos, M. Seltzer, and H. Pfister. Evaluation of Filesystem Provenance Visualization Tools. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '13)*, 2013.

[CDF09] E. Clarkson, K. Desai, and J. Foley. ResultMaps: Visualization for Search Interfaces. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1057–1064, 2009.

[CDRC08] Alexandre Carvalho, A. Sousa Augusto De, Cristina Ribeiro, and Emilia Costa. A Temporal Focus + Context Visualization Model for Handling Valid-Time Spatial Information. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '08)*, 7(3-4):265–274, 2008.

[CEH+09] Min Chen, David Ebert, Hans Hagen, Robert S Laramee, Robert van Liere, Kwan-Liu Ma, William Ribarsky, Gerik Scheuermann, and Deborah Silver. Data, Information, and Knowledge in Visualization. *IEEE Computer Graphics and Applications*, 29:12–19, 2009.

[CGM+17] Davide Ceneda, Theresia Gschwandtner, Thorsten May, Silvia Miksch, Hans-Jörg Schulz, Marc Streit, and Christian Tominski. Characterizing Guidance in Visual Analytics. *IEEE Transactions on Visualization and Computer Graphics (VAST'16)*, 23(1):111–120, 2017.

[CGW13] Kris Cook, Georges Grinstein, and Mark Whiting. VAST challenge dataset 2013, mini-challenge 3, 2013.

[Chi00] Ed H. Chi. A Taxonomy of Visualization Techniques Using the Data State Reference Model. In *Proceedings of the IEEE Symposium on Information Vizualization (InfoVis '00)*, pages 69–75. IEEE Computer Society, 2000.

[CKB08] Andy Cockburn, Amy Karlson, and Benjamin B. Bederson. A review of overview+detail, zooming, and focus+context interfaces. *ACM Computing*

Surveys (CSUR), 41(1):1–31, 2008.

[Cle93] William S Cleveland. *Visualizing Data*. Hobart Press, 1993.

[CLWM11] Tarik Crnovrsanin, Isaac Liao, Yingcai Wu, and Kwan-Liu Ma. Visual Recommendations for Network Navigation. *Computer Graphics Forum*, 30(3):1081–1090, June 2011.

[CMS99] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman, editors. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[CN02] Stuart K. Card and David Nation. Degree-of-Interest Trees: A Component of an Attention-Reactive User Interface. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '02, pages 231–245, New York, NY, USA, 2002. ACM.

[DK10] Aritra Dasgupta and Robert Kosara. Pargnostics: Screen-Space Metrics for Parallel Coordinates. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '10)*, 16(6):1017–1026, 2010.

[DK11] A Dasgupta and R. Kosara. Adaptive privacy-preserving visualization using parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '11)*, 17(12):2241–2248, 2011.

[ED07] Geoffrey Ellis and Alan Dix. A Taxonomy of Clutter Reduction for Information Visualisation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1216–1223, 2007.

[EF10] Niklas Elmqvist and Jean-Daniel Fekete. Hierarchical Aggregation for Information Visualization: Overview, Techniques, and Design Guidelines. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):439–454, 2010.

[FB95] George W. Furnas and Benjamin B. Bederson. Space-scale diagrams: understanding multiscale interfaces. In *Proceedings on the Conference on Human Factors in Computing Systems (CHI '95)*, pages 234–241. ACM Press / Addison-Wesley Publishing Co., 1995.

[FFM+13] Johannes Fuchs, Fabian Fischer, Florian Mansmann, Enrico Bertini, and Petra Isenberg. Evaluation of alternative glyph designs for time series data in a small multiple setting. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3237–3246. ACM, 2013.

[Fis10] Danyel Fisher. Animation for Visualization: Opportunities and Drawbacks. In *Bautiful Visualization*, volume Chapter 19, pages 329–352. 2010.

[FMK+08] Fabian Fischer, Florian Mansmann, Daniel A Keim, Stephan Pietzko, and Marcel Waldvogel. Large-scale network monitoring for visual analysis of attacks. In *Proceedings of the Workshop on Visualization for Computer Security (VizSec '08)*, pages 111–118. Springer, 2008.

[FMS08] James Frew, Dominic Metzger, and Peter Slaughter. Automatic Capture and Reconstruction of Computational Provenance. *Concurrency and Computation: Practice and Experience*, 20(5):485–496, 2008.

[FSKS08] Juliana Freire, Claudio T. Silva, David Koop, and Emanuele Santos. Provenance for Computational Tasks: A Survey. *Computing in Science & Engineering*, 10:11–21, 2008.

[Fur86] George W. Furnas. Generalized fisheye views. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '86)*, pages 16–23. ACM, 1986.

[Gar06] Everette Jr Gardner. Exponential smoothing: The state of the art–Part II. *International Journal of Forecasting*, 22(4):637–666, 2006.

[GAW+11] Michael Gleicher, Danielle Albers, Rick Walker, Ilir Jusufi, Charles D. Hansen, and Jonathan C. Roberts. Visual comparison for information visualization. *Information Visualization*, 10(4):289 –309, 2011.

[GBLZ+15] Alejandra González-Beltrán, Peter Li, Jun Zhao, Maria Susana Avila-Garcia, Marco Roos, Mark Thompson, Eelke van der Horst, Rajaram Kaliyaperumal, Ruibang Luo, Tin-Lap Lee, Tak-wah Lam, Scott C. Edmunds, Susanna-Assunta Sansone, and Philippe Rocca-Serra. From Peer-Reviewed to Peer-Reproduced in Scholarly Publishing: The Complementary Roles of Data Models and Workflows in Bioinformatics. *PLOS ONE*, 10(7):e0127612, 2015.

[GK10] Martin Graham and Jessie Kennedy. A survey of multiple tree visualisation. *Information Visualization*, 9(4):235–252, 2010.

[Gle18] Michael Gleicher. Considerations for Visualizing Comparison. *IEEE Trans. Vis. Comput. Graph.*, 24(1):413–423, January 2018.

[GLG+13] Samuel Gratzl, Alexander Lex, Nils Gehlenborg, Hanspeter Pfister, and Marc Streit. LineUp: Visual Analysis of Multi-Attribute Rankings. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '13)*, 19(12):2277–2286, 2013.

[GLG+16] Samuel Gratzl, Alexander Lex, Nils Gehlenborg, Nicola Cosgrove, and Marc Streit. From Visual Exploration to Storytelling and Back Again.

Computer Graphics Forum, 35(3):491–500, 2016.

[GNT10] Jeremy Goecks, Anton Nekrutenko, and James Taylor. Galaxy: A Comprehensive Approach for Supporting Accessible, Reproducible, and Transparent Computational Research in the Life Sciences. *Genome Biology*, 11(8):R86, 2010.

[GST13] Stefan Gladisch, Heidrun Schumann, and Christian Tominski. Navigation Recommendations for Exploring Hierarchical Graphs. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Baoxin Li, Fatih Porikli, Victor Zordan, James Klosowski, Sabine Coquillart, Xun Luo, Min Chen, and David Gotz, editors, *Advances in Visual Computing*, volume 8034, pages 36–47. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[GZ09] David Gotz and Michelle X Zhou. Characterizing users' visual analytic activity for insight provenance. *Information Visualization*, 8(1):42–55, 2009.

[Hau06] Helwig Hauser. Generalizing Focus+Context Visualization. In Georges-Pierre Bonneau, Thomas Ertl, and Gregory M. Nielson, editors, *Scientific Visualization: The Visual Extraction of Knowledge from Data*, Mathematics and Visualization, pages 305–327. Springer Berlin Heidelberg, 2006.

[HC04] Jeffrey Heer and Stuart K. Card. DOITrees Revisited: Scalable, Space-Constrained Visualization of Hierarchical Data. In *Proceedings of the Conference on Advanced Visual Interfaces (AVI '04)*, pages 421–424. ACM, 2004.

[HDBL17] Melanie Herschel, Ralf Diestelkämper, and Houssem Ben Lahmar. A survey on provenance: What for? What form? What from? *The VLDB Journal*, 26(6):881–906, December 2017.

[HDKS05] Ming C. Hao, Umeshwar Dayal, Daniel A. Keim, and Tobias Schreck. Importance-driven visualization layouts for large time series data. In *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on*, pages 203–210. IEEE, 2005.

[HDKS07] Ming C. Hao, Umeshwar Dayal, Daniel A. Keim, and Tobias Schreck. Multi-Resolution Techniques for Visual Exploration of Large Time-Series Data. *Proceedings of the Symposium on Visualization (EuroVis '07)*, pages 27–34, 2007.

[HG13] Y. Huang and R. Gottardo. Comparability and reproducibility of biomedical data. *Briefings in Bioinformatics*, 14(4):391–401, 2013.

[HKA09] Jeffrey Heer, Nicholas Kong, and Maneesh Agrawala. Sizing the Horizon: The Effects of Chart Size and Layering on the Graphical Perception of Time Series Visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*, pages 1303–1312. ACM, 2009.

[HLD02] Helwig Hauser, Florian Ledermann, and Helmut Doleisch. Angular Brushing of Extended Parallel Coordinates. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '02)*, pages 127–130. IEEE, 2002.

[HMM00] Herman, Melançon, and Marshall. Graph Visualization and Navigation in Information Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.

[HMSA08] Jeffrey Heer, Jock Mackinlay, Chris Stolte, and Maneesh Agrawala. Graphical Histories for Visualization: Supporting Analysis, Communication, and Evaluation. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '08)*, 14(6):1189–1196, 2008.

[HQC+19] J. Hullman, X. Qiao, M. Correll, A. Kale, and M. Kay. In Pursuit of Error: A Survey of Uncertainty Visualization Evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):903–913, January 2019.

[HR07] J. Heer and G. G Robertson. Animated transitions in statistical data graphics. *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '07)*, 13(6):1240–1247, 2007.

[HSS15] Steffen Hadlak, Heidrun Schumann, and Hans-Jörg Schulz. A Survey of Multi-faceted Graph Visualization. In *Proceedings of the Eurographics Conference on Visualization (EuroVis '15) – State of The Art Reports*. The Eurographics Association, 2015.

[HW08] Danny Holten and Jarke J. van Wijk. Visual comparison of hierarchically organized data. *Computer Graphics Forum (EuroVis '08)*, 27(3):759–766, 2008.

[JCPB11] Jian Zhao, F. Chevalier, E. Pietriga, and R. Balakrishnan. Exploratory Analysis of Time-Series with ChronoLenses. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2422–2431, December 2011.

[JKS+17] Daekyoung Jung, Wonjae Kim, Hyunjoo Song, Jeong-in Hwang, Bong-

shin Lee, Bohyoung Kim, and Jinwook Seo. ChartSense: Interactive Data Extraction from Chart Images. pages 6706–6717. ACM Press, 2017.

[JM97] Michael Jünger and Petra Mutzel. 2-Layer Straightline Crossing Minimization: Performance of Exact and Heuristic Algorithms. *Journal of Graph Algorithms and Application*, 1:1–25, 1997.

[JME10] Waqas Javed, Bryan McDonnel, and Niklas Elmqvist. Graphical perception of multiple time series. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '10)*, 16(6):927–934, 2010.

[JS91] B. Johnson and B. Shneiderman. Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the IEEE Conference on Visualization (Vis '91)*, pages 284–291, 1991.

[Kai15] Jocelyn Kaiser. The Cancer Test. *Science*, 348(6242):1411–1413, 2015.

[KAK95] Daniel A. Keim, Mihael Ankerst, and Hans-Peter Kriegel. Recursive Pattern: A Technique for Visualizing Very Large Amounts of Data. In *Proceedings of the IEEE Conference on Visualization (Vis '95)*, pages 279–286. IEEE, 1995.

[KBK11] M. Krstajic, E. Bertini, and D. Keim. Cloudlines: Compact Display of Event Episodes in Multiple Time-Series. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2432–2439, 2011.

[KC14] Brittany Kondo and Christopher Collins. DimpVis: Exploring Time-varying Information Visualizations by Direct Manipulation. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '14)*, 20(12):2003–2012, 2014.

[Kin10] R. Kincaid. SignalLens: Focus+Context Applied to Electronic Time Series. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):900–907, November 2010.

[KKC14] Natalie Kerracher, Jessie Kennedy, and Kevin Chalmers. The design space of temporal graph visualisation. In *Proceedings of the Eurographics Conference on Visualization (EuroVis '14, Short Papers Track)*, 2014.

[KKW+16] Saiful Khan, Urszula Kanturska, Tom Waters, James Eaton, René Bañares-Alcántara, and Min Chen. Ontology-assisted provenance visualization for supporting enterprise search of engineering and business files. *Advanced Engineering Informatics*, 30(2):244–257, 2016.

[KL83] J. B. Kruskal and J. M. Landwehr. Icicle plots: Better displays for hierarchical clustering. *The American Statistician*, 37(2):162, 1983.

[KMS+08] Daniel A. Keim, Florian Mansmann, Jörn Schneidewind, Jim Thomas, and Hartmut Ziegler. Visual Analytics: Scope and Challenges. In Simeon J. Simoff, Michael H. Böhlen, and Arturas Mazeika, editors, *Visual Data Mining*, number 4404 in Lecture Notes in Computer Science, pages 76–90. Springer Berlin Heidelberg, January 2008.

[KNS04] M. Kreuseler, T. Nocke, and H. Schumann. A History Mechanism for Visual Data Mining. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '04)*, pages 49–56. IEEE, 2004.

[KPS14] Simone Kriglstein, Margit Pohl, and Michael Smuc. Pep Up Your Time Machine: Recommendations for the Design of Information Visualizations of Time-Dependent Data. In Weidong Huang, editor, *Handbook of Human Centric Visualization*, pages 203–225. 2014.

[KRD+15] S. Kairam, N. H. Riche, S. Drucker, R. Fernandez, and J. Heer. Refinery: Visual Exploration of Large, Heterogeneous Networks through Associative Browsing. *Computer Graphics Forum*, 34(3):301–310, June 2015.

[LAB+09] T. Lammarsch, W. Aigner, A. Bertone, J. Gartner, E. Mayr, S. Miksch, and M. Smuc. Hierarchical Temporal Patterns and Interactive Aggregated Views for Pixel-Based Visualizations. In *IEEE Transactions on Visualization and Computer Graphics (InfoVis '09)*, pages 44–50. IEEE, 2009.

[LPB+06] Bongshin Lee, C.S. Parr, B.B. Bederson, V.D. Veksler, W.D. Gray, C. Kotfila, and C. Kotfila. TreePlus: Interactive Exploration of Networks with Enhanced Tree Layouts. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1414–1426, November 2006.

[LPK+13] Alexander Lex, Christian Partl, Denis Kalkofen, Marc Streit, Samuel Gratzl, Anne Mai Wasserman, Dieter Schmalstieg, and Hanspeter Pfister. Entourage: Visualizing relationships between biological pathways using contextual subsets. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '13)*, 19(12):2536–2545, 2013.

[LPP+06] 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 (BELIV '06)*, pages 1–5. ACM, 2006.

[LZ10] Su Te Lei and Kang Zhang. A visual analytics system for financial time-series data. In *Proceedings of the Symposium on Visual Information Communication (VINCI '10)*, page 20. ACM, 2010.

[MA14] Silvia Miksch and Wolfgang Aigner. A Matter of Time: Applying a

Data–Users–Tasks Design Triangle to Visual Analytics of Time-Oriented Data. *Computers & Graphics*, 38(Supplement C):286–290, 2014.

[Mac86] Jock Mackinlay. Automating the Design of Graphical Presentations of Relational Information. *ACM Transactions on Graphics*, 5(2):110–141, 1986.

[Mar99] Martin Wattenberg. Visualizing the Stock Market. *Extended Abstracts on Human Factors in Computing Systems (CHI '99)*, pages 188–189, 1999.

[MG11] Peter Mell and Tim Grance. The NIST definition of cloud computing, 2011.

[MGT+03] Tamara Munzner, François Guimbretière, Serdar Tasiran, Li Zhang, and Yunhong Zhou. TreeJuxtaposer: Scalable tree comparison using focus+context with guaranteed visibility. In *Proceedings of the ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '03)*, pages 453–462. ACM, 2003.

[MMKN08] Peter McLachlan, Tamara Munzner, Eleftherios Koutsofios, and Stephen North. LiveRAC: Interactive visual exploration of system management time-series data. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*, pages 1483–1492. ACM, 2008.

[MRSS+12] E. Maguire, P. Rocca-Serra, S.-A. Sansone, J. Davies, and Min Chen. Taxonomy-Based Glyph Design with a Case Study on Visualizing Workflows of Biological Experiments. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '12)*, 18(12):2603–2612, 2012.

[MRSS+13] E. Maguire, P. Rocca-Serra, S.-A. Sansone, J. Davies, and M. Chen. Visual Compression of Workflow Visualizations with Automated Detection of Macro Motifs. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2576–2585, 2013.

[MS11] Peter Macko and Margo I. Seltzer. Provenance Map Orbiter: Interactive Exploration of Large Provenance Graphs. In *Proceedings of the USENIX Workshop on the Theory and Practice of Provenance (TaPP '11)*, 2011. Engineering and Applied Sciences.

[MSDK12] T. May, M. Steiger, J. Davey, and J. Kohlhammer. Using Signposts for Navigation in Large Graphs. *Computer Graphics Forum*, 31(3pt2):985–994, June 2012.

[MSOI+02] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network Motifs: Simple Building Blocks of Complex Networks. *Science*,

298(5594):824–827, 2002.

[Mun14] Tamara Munzner. *Visualization Analysis and Design*. CRC Press, Taylor & Francis Group, Boca Raton, 2014.

[NJ04] Steven Noel and Sushil Jajodia. Managing Attack Graph Complexity Through Visual Hierarchical Aggregation. In *Proceedings of the ACM Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC '04)*, pages 109–118. ACM, 2004.

[NSH+18] Christina Niederer, Holger Stitz, Reem Hourieh, Florian Grassinger, Wolfgang Aigner, and Marc Streit. TACO: Visualizing Changes in Tables Over Time. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '17)*, 24(1):677–686, 2018.

[NSS07] Galileo Namata, Brian Staats, and Ben Shneiderman. DualNet: A coordinated view approach to network visualization. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM '07)*. ACM, 2007.

[OEY+13] Larsson Omberg, Kyle Ellrott, Yuan Yuan, Cyriac Kandoth, Chris Wong, Michael R. Kellen, Stephen H. Friend, Josh Stuart, Han Liang, and Adam A. Margolin. Enabling Transparent and Collaborative Computational Analysis of 12 Tumor Types within The Cancer Genome Atlas. *Nature Genetics*, 45(10):1121–1126, 2013.

[OW93] G. Ozsoyoglu and H. Wang. Example-based graphical database query languages. *Computer*, 26(5):25–38, May 1993.

[Par09] Peter J. Park. Chip–Seq: Advantages and Challenges of a Maturing Technology. *Nature Reviews Genetics*, 10(10):669–680, 2009.

[PCJ07] Pat Hanrahan, Chris Stolte, and Jock Mackinlay. Tableau: Visual Analysis for Everyone, 2007.

[PF93] Ken Perlin and David Fox. Pad: An Alternative Approach to the Computer Interface. In *Proceedings of the ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '93)*, pages 57–64. ACM, 1993.

[PGS+16] Christian Partl, Samuel Gratzl, Marc Streit, Anne Mai Wassermann, Hanspeter Pfister, Dieter Schmalstieg, and Alexander Lex. Pathfinder: Visual analysis of paths in graphs. *Computer Graphics Forum (EuroVis '16)*, 35(3):71–80, jun 2016.

[PH17] Jorge Poco and Jeffrey Heer. Reverse-Engineering Visualizations: Recovering Visual Encodings from Chart Images. *Computer Graphics Forum*,

36(3):353–363, June 2017.

[PHT+17] Robert Pienta, Fred Hohman, Acar Tamersoy, Alex Endert, Shamkant Navathe, Hanghang Tong, and Duen Horng Chau. Visual Graph Query Construction and Refinement. pages 1587–1590. ACM Press, 2017.

[PKL+17] Robert Pienta, Minsuk Kahng, Zhiyuan Lin, Jilles Vreeken, Partha Talukdar, James Abello, Ganesh Parameswaran, and Duen Horng Chau. Facets: Adaptive local exploration of large graphs. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 597–605. SIAM, 2017.

[PLS+13] Christian Partl, Alexander Lex, Marc Streit, Denis Kalkofen, Karl Kashofer, and Dieter Schmalstieg. enRoute: Dynamic path extraction from biological pathway maps for exploring heterogeneous experimental datasets. *BMC Bioinformatics*, 14(Suppl 19):S3, 2013.

[PMH18] Jorge Poco, Angela Mayhua, and Jeffrey Heer. Extracting and Retargeting Color Mappings from Bitmap Images of Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):637–646, January 2018.

[PPS14] A. Johannes Pretorius, Helen C. Purchase, and John T. Stasko. Tasks for Multivariate Network Analysis. In Andreas Kerren, Helen C. Purchase, and Matthew O. Ward, editors, *Multivariate Network Visualization*, number 8380 in Lecture Notes in Computer Science, pages 77–95. Springer International Publishing, January 2014.

[PRSA18] Beatriz Pérez, Julio Rubio, and Carlos Sáenz-Adán. A systematic review of provenance systems. *Knowledge and Information Systems*, February 2018.

[PSTW+17] Stephan Pajer, Marc Streit, Thomas Torsney-Weir, Florian Spechtenhauser, Torsten Möller, and Harald Piringer. WeightLifter: Visual Weight Space Exploration for Multi-Criteria Decision Making. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '16)*, 23(1):611–620, 2017.

[PVH12] Adam Perer and Frank Van Ham. Integrating querying and browsing in partial graph visualizations. Technical report, Technical Report 12-01, IBM Research, 2012.

[RAM+11] Alexander Rind, Wolfgang Aigner, Silvia Miksch, Sylvia Wiltner, Margit Pohl, Felix Drexler, Barbara Neubauer, and Nikolaus Suchy. Visually Exploring Multivariate Trends in Patient Cohorts Using Animated Scatter Plots. In Michelle Robertson, editor, *Ergonomics and Health Aspects of Work with Computers*, volume 6779 of *Lecture Notes in Computer Science*,

pages 139–148. Springer, 2011.

[RCC05] George G. Robertson, Mary P. Czerwinski, and John E. Churchill. Visualization of mappings between schemas. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI '05)*, pages 431–439. ACM, 2005.

[RESC15] Eric Ragan, Alex Endert, Jibonananda Sanyal, and Jian Chen. Characterizing Provenance in Visualization and Data Analysis: An Organizational Framework of Provenance Types and Purposes. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '15)*, 22(1):31 – 40, 2015.

[RFF+08] George Robertson, Roland Fernandez, Danyel Fisher, Bongshin Lee, and John Stasko. Effectiveness of animation in trend visualization. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '08)*, 14(6):1325–1332, 2008.

[RLMJ05] Ruth Rosenholtz, Yuanzhen Li, Jonathan Mansfield, and Zhenlan Jin. Feature congestion: a measure of display clutter. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems (CHI '05)*, pages 761–770. ACM Press, 2005.

[Row07] Jennifer Rowley. The wisdom hierarchy: representations of the DIKW hierarchy. *Journal of Information Science*, 33(2):163–180, April 2007.

[RSBM+10] Philippe Rocca-Serra, Marco Brandizi, Eamonn Maguire, Nataliya Sklyar, Chris Taylor, Kimberly Begley, Dawn Field, Stephen Harris, Winston Hide, Oliver Hofmann, Steffen Neumann, Peter Sterk, Weida Tong, and Susanna-Assunta Sansone. ISA Software Suite: Supporting Standards-compliant Experimental Annotation and Enabling Curation at the Community Level. *Bioinformatics*, 26(18):2354–2356, 2010.

[SAAF18] G. Shurkhovetskyy, N. Andrienko, G. Andrienko, and G. Fuchs. Data Abstraction for Visualizing Large Time Series: Data Abstraction for Visualizing Large Time Series. *Computer Graphics Forum*, 37(1):125–144, February 2018.

[SCM+06] Greg Smith, Mary Czerwinski, B. Robbins Meyers, G. Robertson, and Daniel Stanley Tan. FacetMap: A Scalable Search and Browse Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5), 2006.

[SFMB12] Michael Sedlmair, Annika Frank, Tamara Munzner, and Andreas Butz. Relex: Visualization for actively changing overlay network specifications. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '12)*,

18(12):2729–2738, 2012.

[SGAS16] Holger Stitz, Samuel Gratzl, Wolfgang Aigner, and Marc Streit. ThermalPlot: Visualizing Multi-Attribute Time-Series Data Using a Thermal Metaphor. *IEEE Transactions on Visualization and Computer Graphics*, 2016. To Appear.

[SGKS15] Holger Stitz, Samuel Gratzl, Michael Krieger, and Marc Streit. CloudGazer: A Divide-and-Conquer Approach for Monitoring and Optimizing Cloud-Based Networks. In *Proceedings of the IEEE Pacific Visualization Symposium (PacificVis '15)*, pages 175–182. IEEE, 2015.

[SGP+18] Holger Stitz, Samuel Gratzl, Harald Piringer, Thomas Zichner, and Marc Streit. KnowledgePearls: Provenance-Based Visualization Retrieval. *IEEE Transactions on Visualization and Computer Graphics (VAST '18)*, page 11, 2018.

[SGS+19] Marc Streit, Samuel Gratzl, Holger Stitz, Andreas Wernitznig, Thomas Zichner, and Christian Haslinger. Ordino: a visual cancer analysis tool for ranking and exploring genes, cell lines and tissue samples. *Bioinformatics*, 2019.

[Shn96] Ben Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Proceedings of the IEEE Symposium on Visual Languages (VL '96)*, pages 336–343. IEEE, 1996.

[SKC+11] Manolis Savva, Nicholas Kong, Arti Chhajta, Li Fei-Fei, Maneesh Agrawala, and Jeffrey Heer. ReVision: automated classification, analysis and redesign of chart images. page 393. ACM Press, 2011.

[SLN05] P. Saraiya, P. Lee, and C. North. Visualization of graphs with associated timeseries data. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '05)*, pages 225–232. IEEE, 2005.

[SLSG16] H. Stitz, S. Luger, M. Streit, and N. Gehlenborg. AVOCADO: Visualization of Workflow–Derived Data Provenance for Reproducible Biomedical Research. *Computer Graphics Forum*, 35(3):481–490, 2016.

[SMWH17] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):341–350, 2017.

[SNHS13] Hans-Jörg Schulz, Thomas Nocke, Magnus Heitzler, and Heidrun Schumann. A Design Space of Visualization Tasks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2366–2375, 2013.

[SRHH16] Arvind Satyanarayan, Ryan Russell, Jane Hoffswell, and Jeffrey Heer. Reactive Vega: A Streaming Dataflow Architecture for Declarative Interactive Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):659–668, January 2016.

[SS06] Hans-Jorg Schulz and Heidrun Schumann. Visualizing Graphs: A Generalized View. In *Proceedings of the IEEE Conference on Information Visualisation (IV '06)*, pages 166–173. IEEE, 2006.

[SS17] A. Srinivasan and J. Stasko. Orko: Facilitating Multimodal Interaction for Visual Exploration and Analysis of Networks. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '17)*, PP(99):1–1, 2017.

[SSS+14] D. Sacha, A. Stoffel, F. Stoffel, B. C. Kwon, G. Ellis, and D. A. Keim. Knowledge Generation Model for Visual Analytics. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1604–1613, December 2014.

[SSSE16] J. Schwank, S. Schöffel, J. Stärz, and A. Ebert. Visualizing Uncertainty of Edge Attributes in Node-Link Diagrams. In *2016 20th International Conference Information Visualisation (IV)*, pages 45–50, July 2016.

[SSTR93] Manojit Sarkar, Scott S. Snibbe, Oren J. Tversky, and Steven P. Reiss. Stretching the rubber sheet: A metaphor for viewing large layouts on small screens. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '93)*, pages 81–91. ACM, 1993.

[STH02a] Chris Stolte, Diane Tang, and Pat Hanrahan. Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):52–65, 2002.

[STH02b] Chris Stolte, Diane Tang, and Pat Hanrahan. Query, Analysis, and Visualization of Hierarchically Structured Data Using Polaris. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '02)*, pages 112–122. ACM, 2002.

[STKF07] Tobias Schreck, Tatiana Tekušová, Jörn Kohlhammer, and Dieter Fellner. Trajectory-based visual analysis of large financial time series data. *ACM SIGKDD Explorations Newsletter*, 9(2):30–37, 2007.

[SVK+08] Carlos E. Scheidegger, Huy T. Vo, David Koop, Juliana Freire, and Claudio T. Silva. Querying and re-using workflows with VisTrails. In *Proceedings of the ACM SIGMOD Conference on Management of Data (SIGMOD '08)*, pages 1251–1254. ACM, 2008.

[SvW08] Yedendra Babu Shrinivasan and Jarke J. van Wijk. Supporting the analytical reasoning process in information visualization. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1237–1246. ACM, 2008.

[SWA92] Ben Shneiderman, Christopher Williamson, and Christopher Ahlberg. Dynamic Queries: Database Searching by Direct Manipulation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '92)*, pages 669–670. ACM, 1992.

[SWH14] Arvind Satyanarayan, Kanit Wongsuphasawat, and Jeffrey Heer. Declarative Interaction Design for Data Visualization. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '14)*, pages 669–678. ACM Press, 2014.

[SWY75] Gerard Salton, Anita Wong, and Chung-Shu Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[TA08] Alexandru Telea and David Auber. Code flows: Visualizing structural evolution of source code. *Computer Graphics Forum (EuroVis '08)*, 27(3):831–838, 2008.

[TAHS06] Christian Tominski, James Abello, Frank van Ham, and Heidrun Schumann. Fisheye Tree Views and Lenses for Graph Visualization. In *Proceedings of the IEEE Conference on Information Visualisation (IV '06)*, pages 17–24. IEEE, 2006.

[TGK+14] Christian Tominski, Stefan Gladisch, Ulrike Kister, Raimund Dachselt, and Heidrun Schumann. A Survey on Interactive Lenses in Visualization. In *Proceedings of the Eurographics Conference on Visualization (EuroVis '14) – State of The Art Reports*, pages 43–62. Eurographics, 2014.

[TK07] Tatiana Tekusova and Jörn Kohlhammer. Applying animation to the visual analysis of financial time-dependent data. In *IEEE Transactions on Visualization and Computer Graphics (InfoVis '07)*, pages 101–108. IEEE, 2007.

[Tuf83] Edward Tufte. *The Visual Display of Quantitative Information.* Graphics Press, 2nd edition, 1983.

[VBW15] Corina Vehlow, Fabian Beck, and Daniel Weiskopf. The State of the Art in Visualizing Group Structures in Graphs. In *Proceedings of the Eurographics Conference on Visualization (EuroVis '15) – State of The Art Reports*, 2015.

[vdEHBvW15] Stef van den Elzen, Danny Holten, Jorik Blaas, and Jarke J. van Wijk. Reducing Snapshots to Points: A Visual Analytics Approach to Dynamic Network Exploration. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '15)*, 22(1):1–10, 2015.

[vHP09] F. van Ham and A. Perer. "Search, Show Context, Expand on Demand": Supporting Large Graph Exploration with Degree-of-Interest. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '09)*, 15(6):953–960, 2009.

[VKB+15] Corinna Vehlow, David P Kao, Michael R Bristow, Lawrence E Hunter, Daniel Weiskopf, and Carsten Görg. Visual Analysis of Biological Data-Knowledge Networks. *BMC Bioinformatics*, 16(1), 2015.

[vLBRS09] Tatiana von Landesberger, Sebastian Bremm, Peyman Rezaei, and Tobias Schreck. Visual analytics of time dependent 2d point clouds. In *Proceedings of the Computer Graphics International Conference (CGI '09)*, pages 97–101. ACM, 2009.

[vLKS+11] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J.J. van Wijk, J.-D. Fekete, and D.W. Fellner. Visual analysis of large graphs: State-of-the-art and future research challenges. *Computer Graphics Forum*, 30(6):1719–1749, 2011.

[VM04] Andrew Vande Moere. Time-Varying Data Visualization Using Information Flocking Boids. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '04)*, pages 97–104. IEEE, 2004.

[WAG05] L. Wilkinson, A. Anand, and R. Grossman. Graph-Theoretic Scagnostics. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '05)*, pages 157–164, 2005.

[War08] Matthew O. Ward. Multivariate data glyphs: Principles and practice. In *Handbook of Data Visualization*, pages 179–198. Springer, 2008.

[Wat06] Martin Wattenberg. Visual Exploration of Multivariate Graphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 811–819, New York, NY, USA, 2006. ACM.

[WEF+14] Michael Wybrow, Niklas Elmqvist, Jean-Daniel Fekete, Tatiana von Landesberger, Jarke J. van Wijk, and Björn Zimmer. Interaction in the Visualization of Multivariate Networks. In Andreas Kerren, Helen C. Purchase, and Matthew O. Ward, editors, *Multivariate Network Visualization*, number 8380 in Lecture Notes in Computer Science, pages 97–125. Springer International Publishing, January 2014.

[WHe13] Katherine Wolstencroft, Robert Haines, and et al. The Taverna Workflow Suite: Designing and Executing Workflows of Web Services on the Desktop, Web or in the Cloud. *Nucleic Acids Research*, 41(W1):W557–W561, 2013.

[Wil05] Leland Wilkinson. *The Grammar of Graphics*. Springer, 2nd edition, 2005.

[Win60] Peter R. Winters. Forecasting Sales by Exponentially Weighted Moving Averages. *Management Science*, 6(3):324–342, 1960.

[YDG+15] V. Yoghourdjian, T. Dwyer, G. Gange, S. Kieffer, K. Klein, and K. Marriott. High-Quality Ultra-Compact Grid Layout of Grouped Networks. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '15)*, 22(1):339–348, 2015.

[ZJGK10] H. Ziegler, M. Jenny, T. Gruse, and D.A. Keim. Visual market sector analysis for financial time series data. pages 83–90. IEEE, 2010.

[Zlo77] M. M. Zloof. Query-by-Example: A data base language. *IBM Systems Journal*, 16(4):324–343, 1977.

A | AVOCADO Supplement

A.1. Usage Scenario: Additional Figures and Workflows

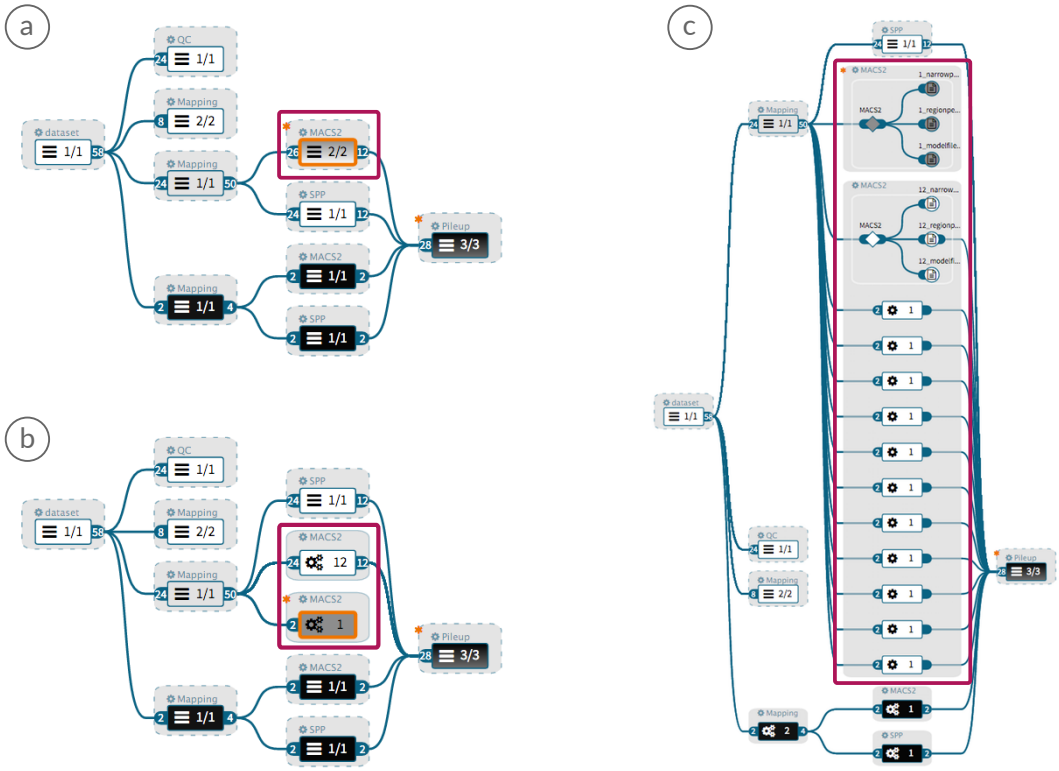


Figure A.1.: Aggregated layer nodes that contain analyses with minor differences. The orange asterisk indicates that analyses that are using the same workflow template but with different parametrization (w.r.t. inputs, outputs, execution time, etc.) are aggregated into a single layer node. (a) The layer node is aggregated. (b) The layer node is expanded to reveal the two analyses, one with 12 analysis input groups and one with 1 analysis input group. (c) Further expansion reveals the workflow instances, which are only partially expanded in the bottom analysis.

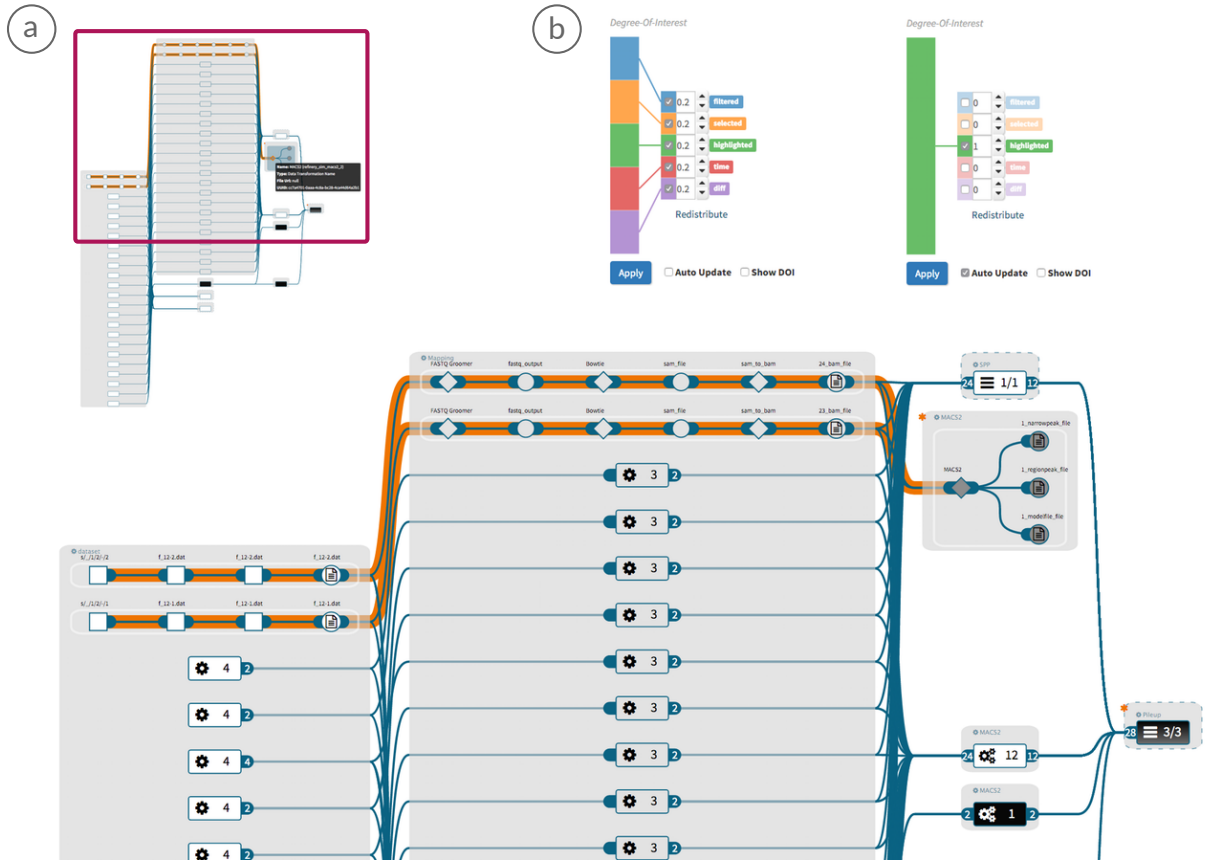


Figure A.2.: Reviewing data provenance for the inputs of a selected tool. (a) The overview shows the whole provenance graph at different levels of aggregation. The nodes along the orange highlighting are fully expanded. The view at the bottom shows details of every data transformation step that was applied before the files were used as input for the *MACS2* tool. (b) The DOI function controller set to default values (left) and to auto-expansion of selected paths (right). Auto-expansion is achieved by increasing the weight of the *highlighted* component to 1 (maximum) and reducing the weight of all other components to 0 (minimum). The *Auto update* option applies the DOI function immediately on every user interaction.

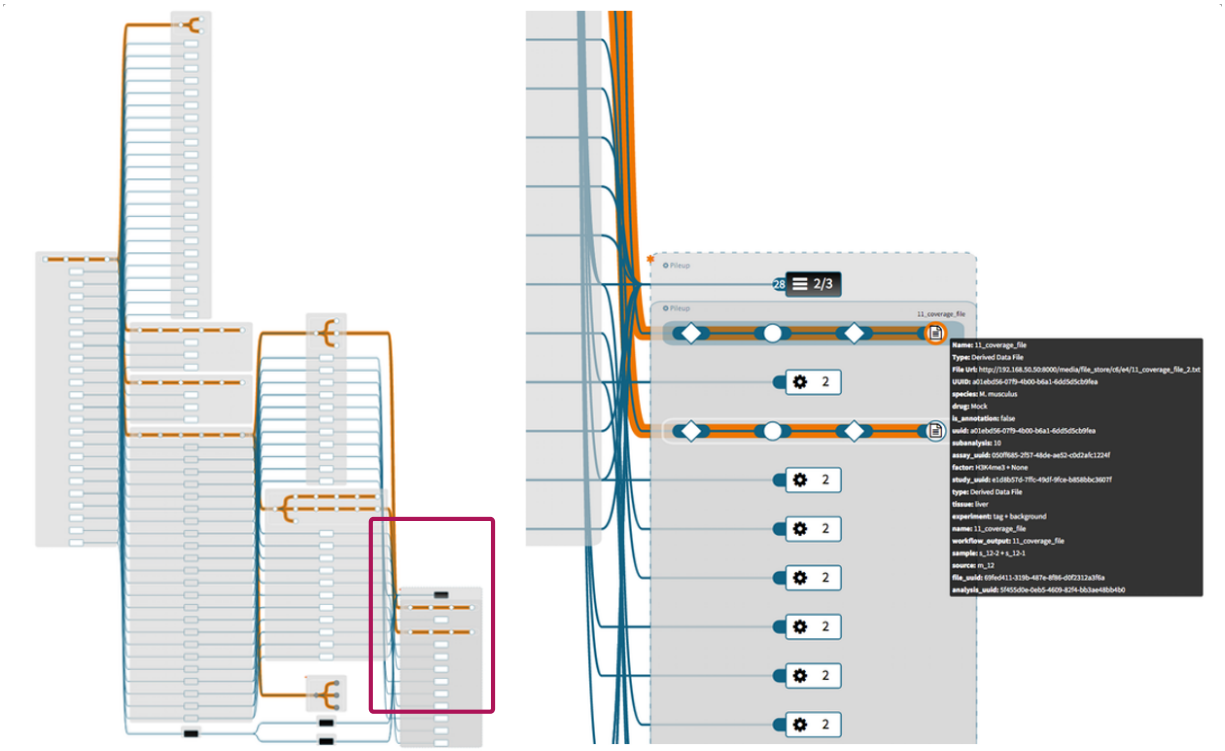


Figure A.3.: Identifying results generated based on a selected raw data file. The overview on the left shows the paths from the selected raw data file to all derived results. Note that the semantic zoom removes labels and other details when there is not enough room to display them. The view on the right shows details about a *Pileup* workflow that generated some of the results derived from the selected raw data file. The expansion of detailed workflow level information is automatically limited to those nodes that are relevant for this task.

Figure A.4.: Quality control workflow with 1 input and 2 outputs.

Figure A.5.: Mapping workflow with 1 input and 1 output.

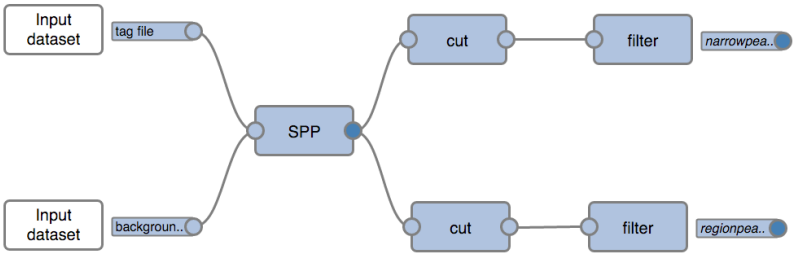


Figure A.6.: SPP workflow with 2 paired inputs and 3 outputs.

Figure A.7.: MACS2 workflow 2 paired inputs and 3 outputs.

Figure A.8.: Pileup workflow 1 input and 1 output.

B | KnowledgePearls Supplement

B.1. Vega integration

Listing B.1: Vega signal declaration with additional `track` and `search` properties for the retrieval.

```
"signals": [{
  "name": "xField",
  "value": "gdp",
  "bind": {
    "input": "select",
    "options": ["gdp", "population", "life_expect",
                "fertility", "child_mortality"]
  },
  "track": {
    "title": "X = {{value}}",
    "category": "data",
    "operation": "update"
  },
  "search": {
    "type": "category",
    "title": "{{value}}",
    "group": "Attributes"
  }
}]
```

B.2. Case Study: Visual Analysis

The following figures show the user interaction with Ordino in chronological order. We describe the user's insight and rationale in each figure caption to make the interaction more transparent.

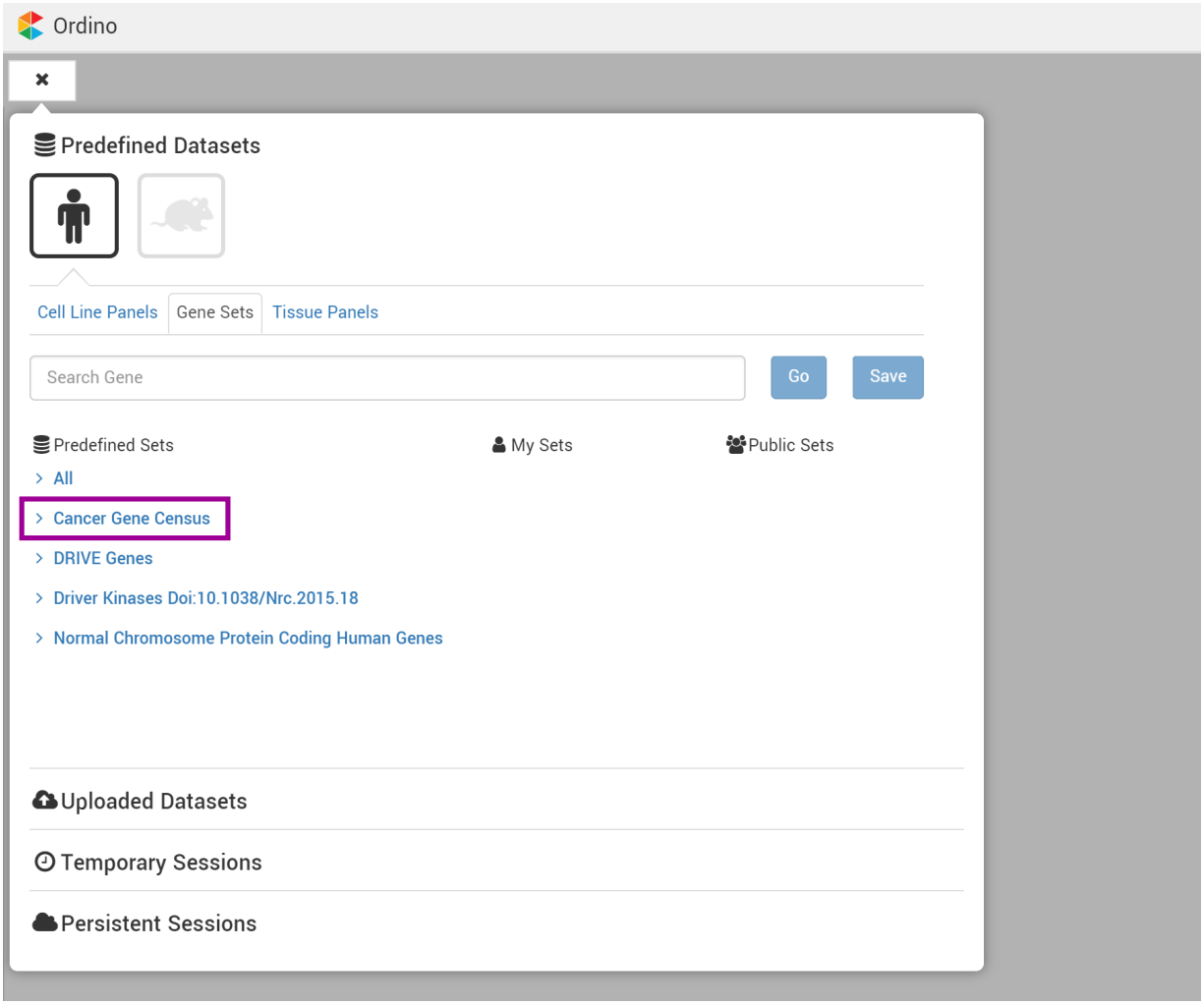


Figure B.1.: The user is interested in the breast cancer cell line *BT-20* and wants to know which cancer gene(s) could be important for growth in that cell line? In Ordino, users can choose between predefined datasets or upload custom datasets. For the case study the user starts with the *Cancer Gene Census* dataset.

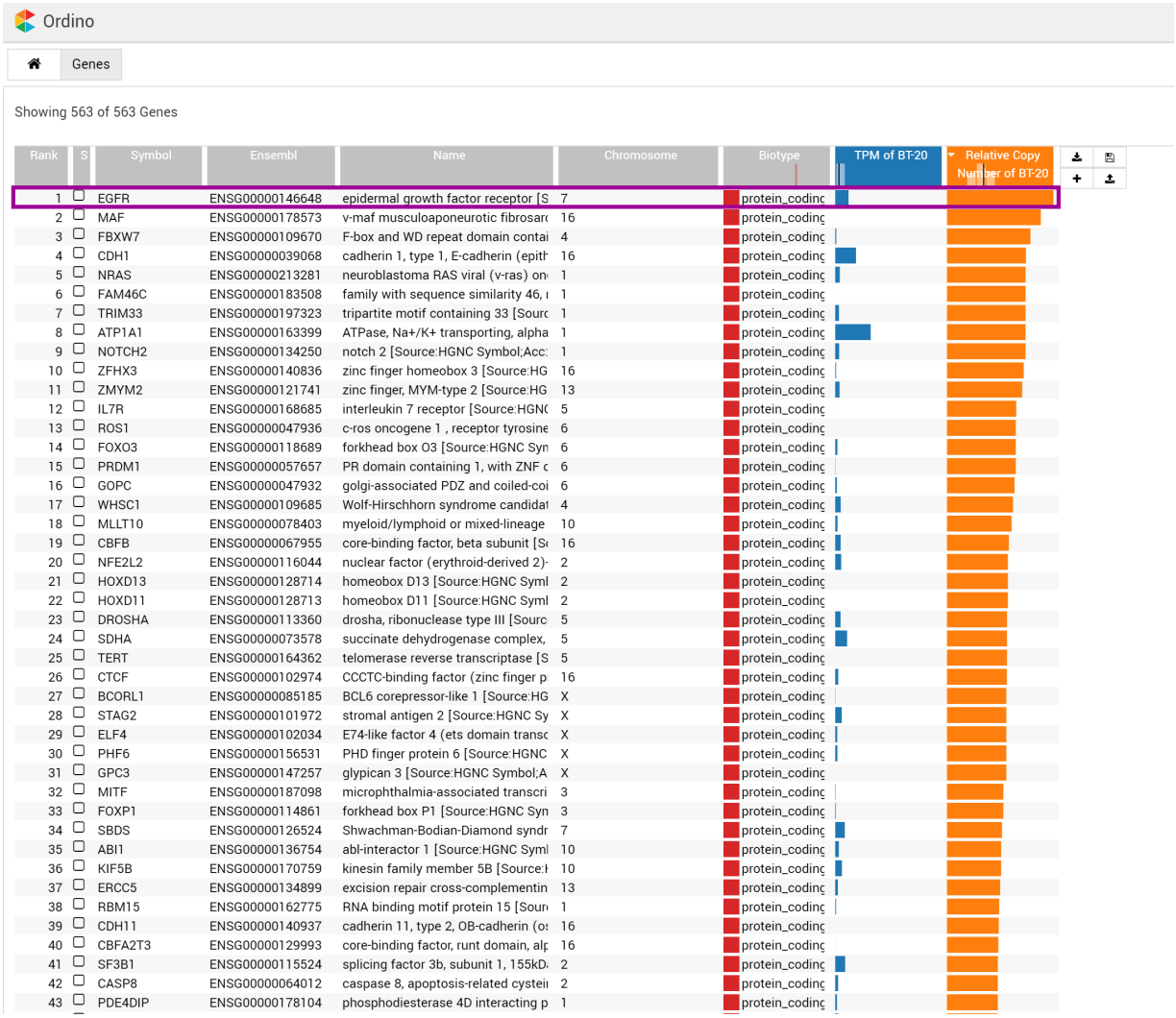


Figure B.2.: Adding the two score columns *TPM* (gene expression level) and *Relative Copy Number* for the cell line *BT-20* and sorting the ranking by the relative copy number column reveals the gene *EGFR* as an amplified and also highly expressed candidate. Likely, it is important for cell growth in *BT-20*.

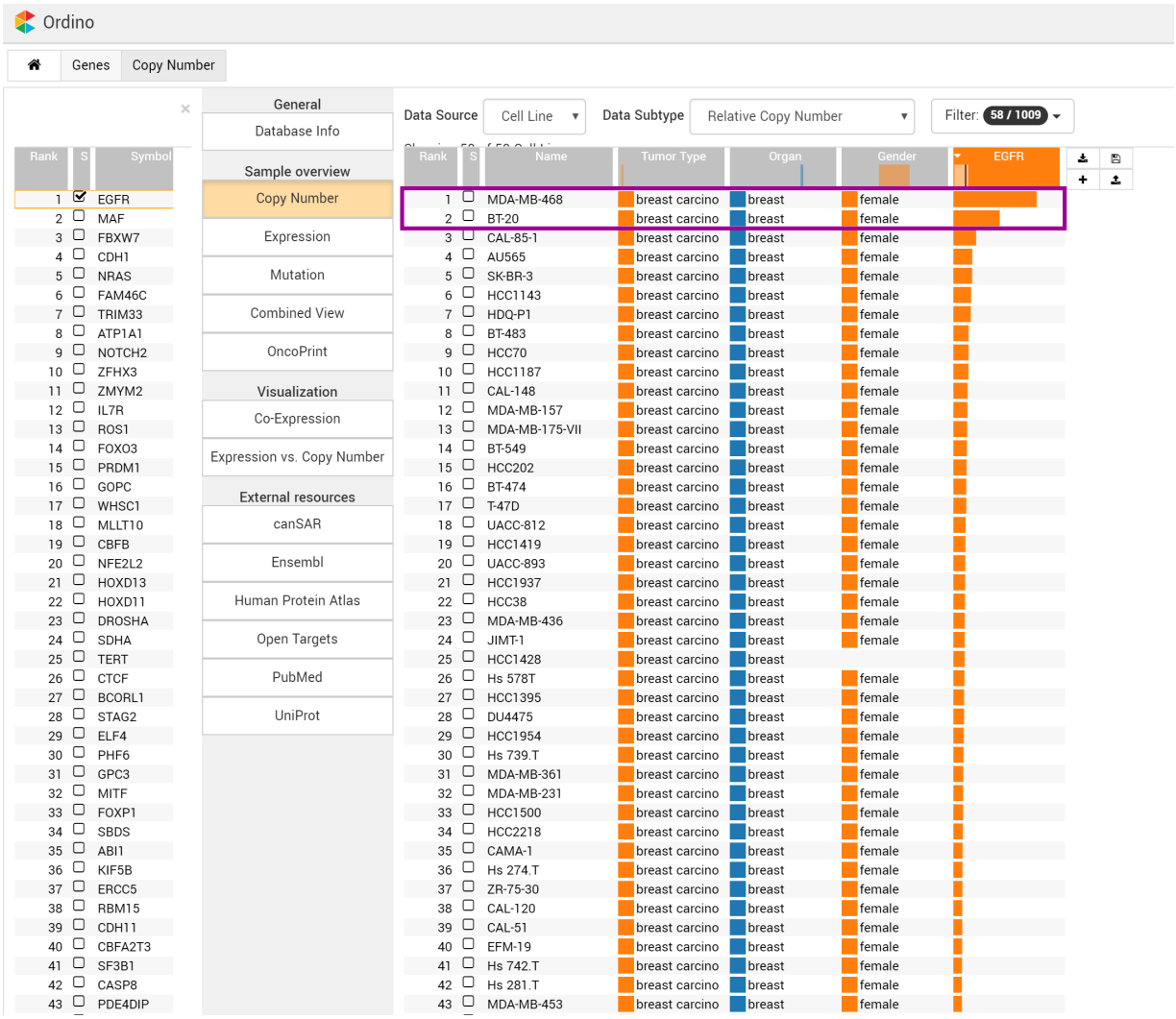


Figure B.3.: Next, the user wants to know, if there are other breast cancer cell lines with *EGFR* amplification? Selecting the gene *EGFR* from the ranking brings up a list of further detail views. The user selects the *Copy Number* view, filters the ranking by tumor type *breast cancer*, and sorts it by the *EGFR* copy number values. The insight is that two breast cancer cell lines have a clear *EGFR* copy number gain ($CN > 4$). There is one cell line, *MDA-MB-468*, that has a higher *EGFR* copy number than *BT-20*.

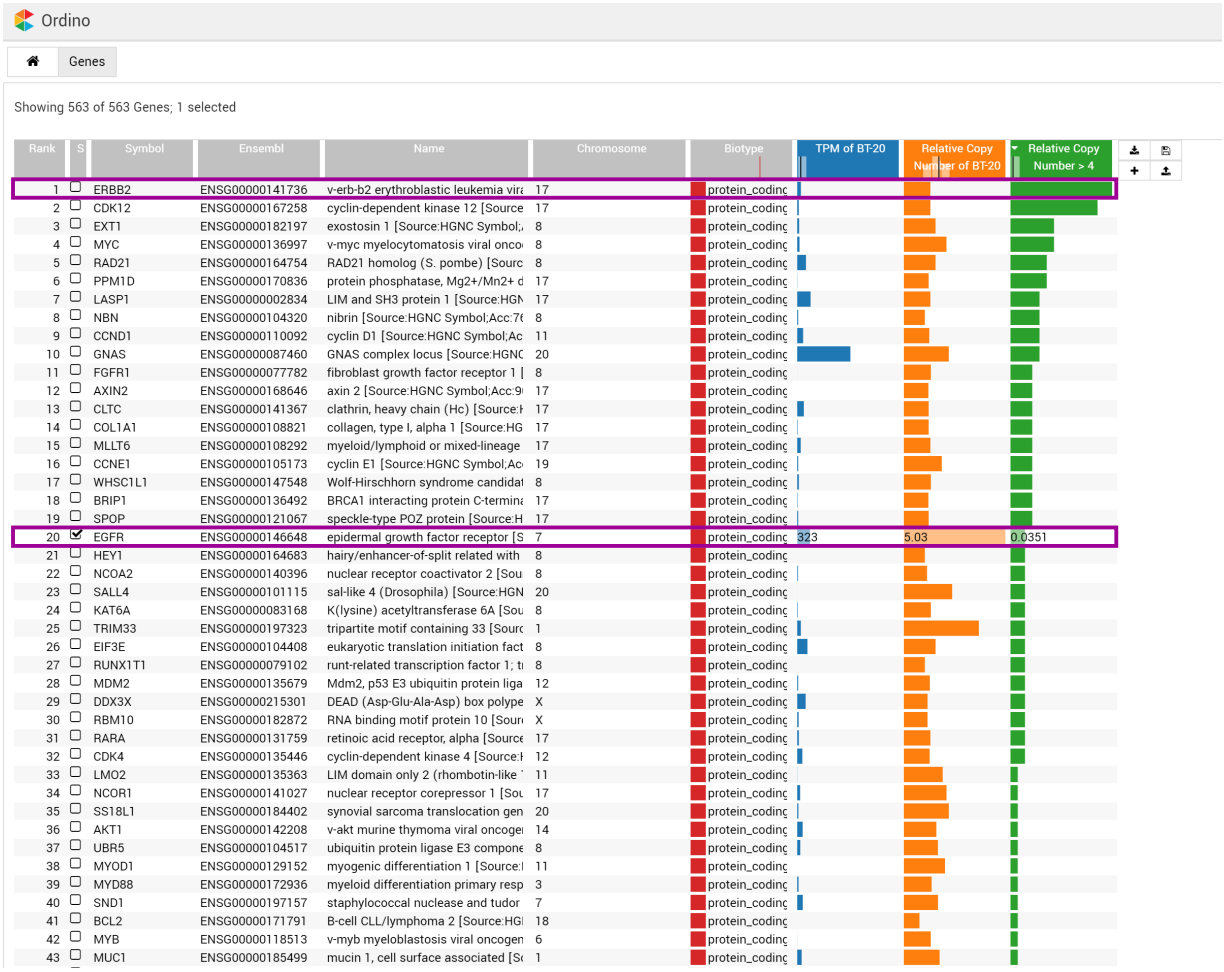


Figure B.4.: The user has a new thought and wants to know which cancer genes are more frequently amplified in breast cancer than *EGFR*? The previous detail view is closed and another score column with the parameters tumor type *breast cancer*; data type *Relative Copy Number*; aggregation *Frequency*; cut-off > 4 is added to the ranking. Sorting this column reveals that *EGFR* is only ranked on place 20 in the list. Most frequently amplified in the assessed breast cancer cell lines is *ERBB2*, which is amplified in about 25% of cell lines.

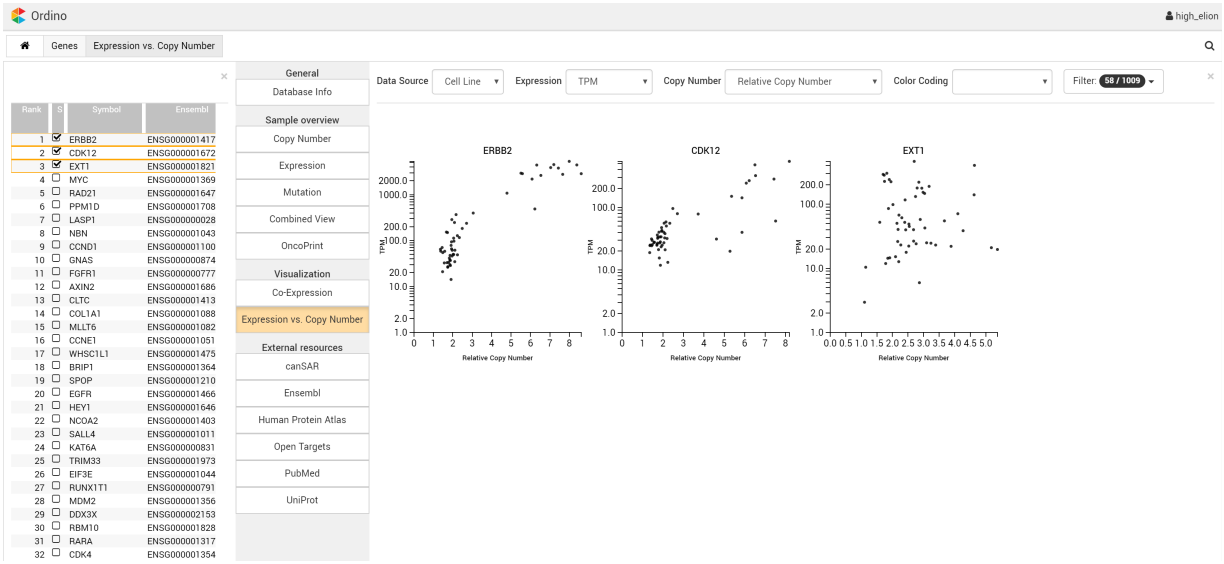


Figure B.5.: Next the user wants to know, if the amplifications of the top genes also lead to a high gene expression? Selecting the first three genes (*ERBB2*, *CDK12*, and *EXT1*) and opening the detail view *Expression vs. Copy Number* (still filtered for *breast cancer* cell lines) shows a scatterplot for every selected gene. It is obvious that the gene expression for all *ERBB2* amplified cell lines is very high. This correlation is also present in a weaker form for *CDK12* and seems to be absent for *EXT1*.

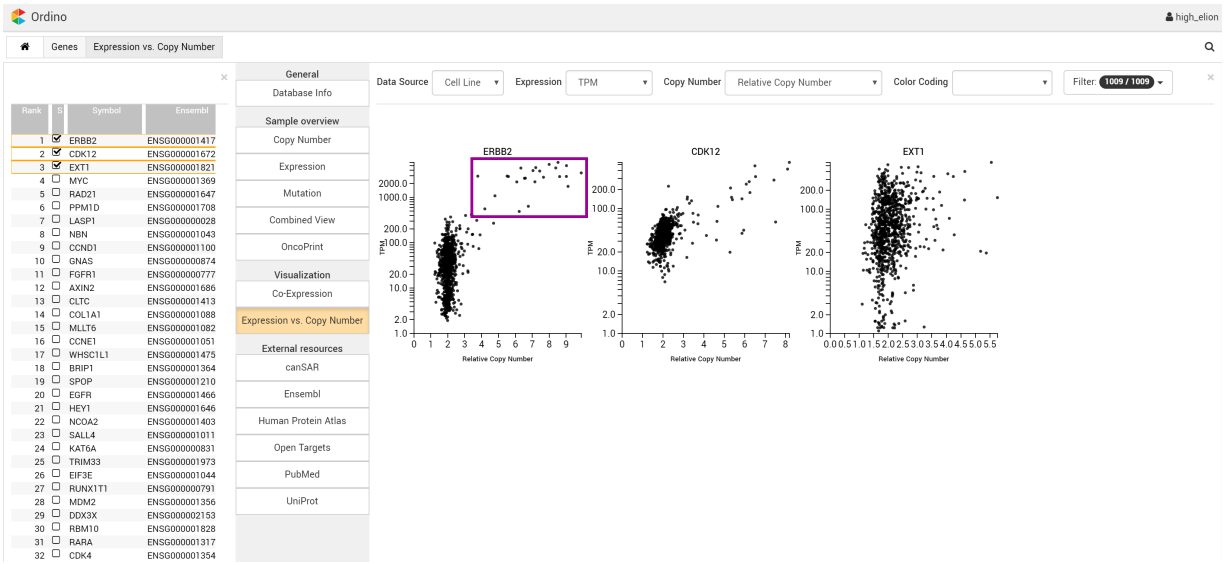


Figure B.6.: The user removes the filter to explore if the expression to copy number correlation is also present in other tumor types. The result is that all cell lines with a very high *ERBB2* copy number have high *ERBB2* expression.

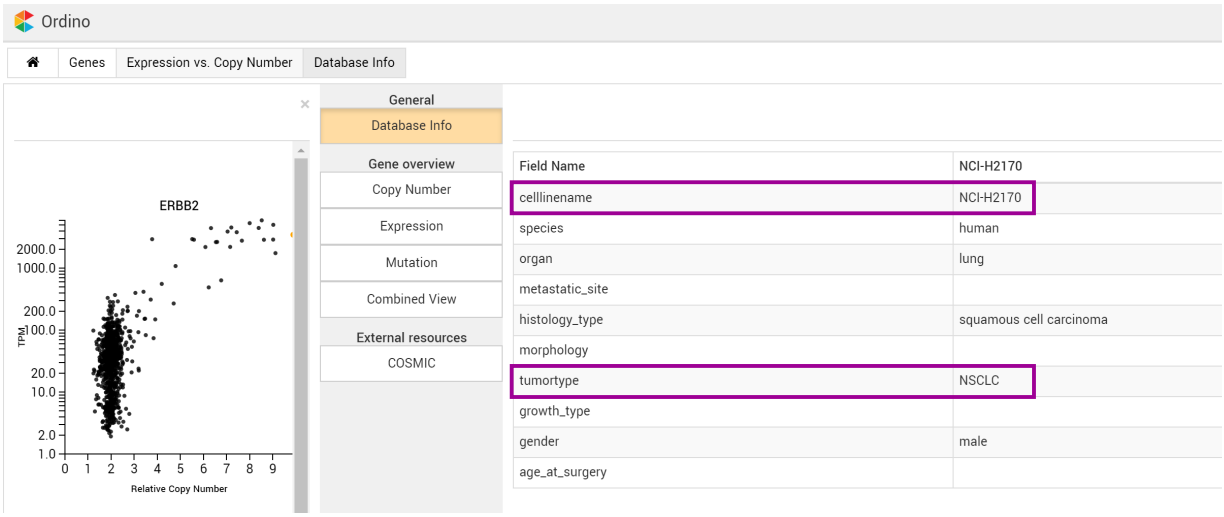


Figure B.7.: Selecting the most right point in the *ERBB2* scatterplot and opening the *Database Info* detail view, reveals that the cell line with the highest *ERBB2* amplification is *NCI-H2170* and belongs to *non-small-cell lung cancer* (NSCLC).

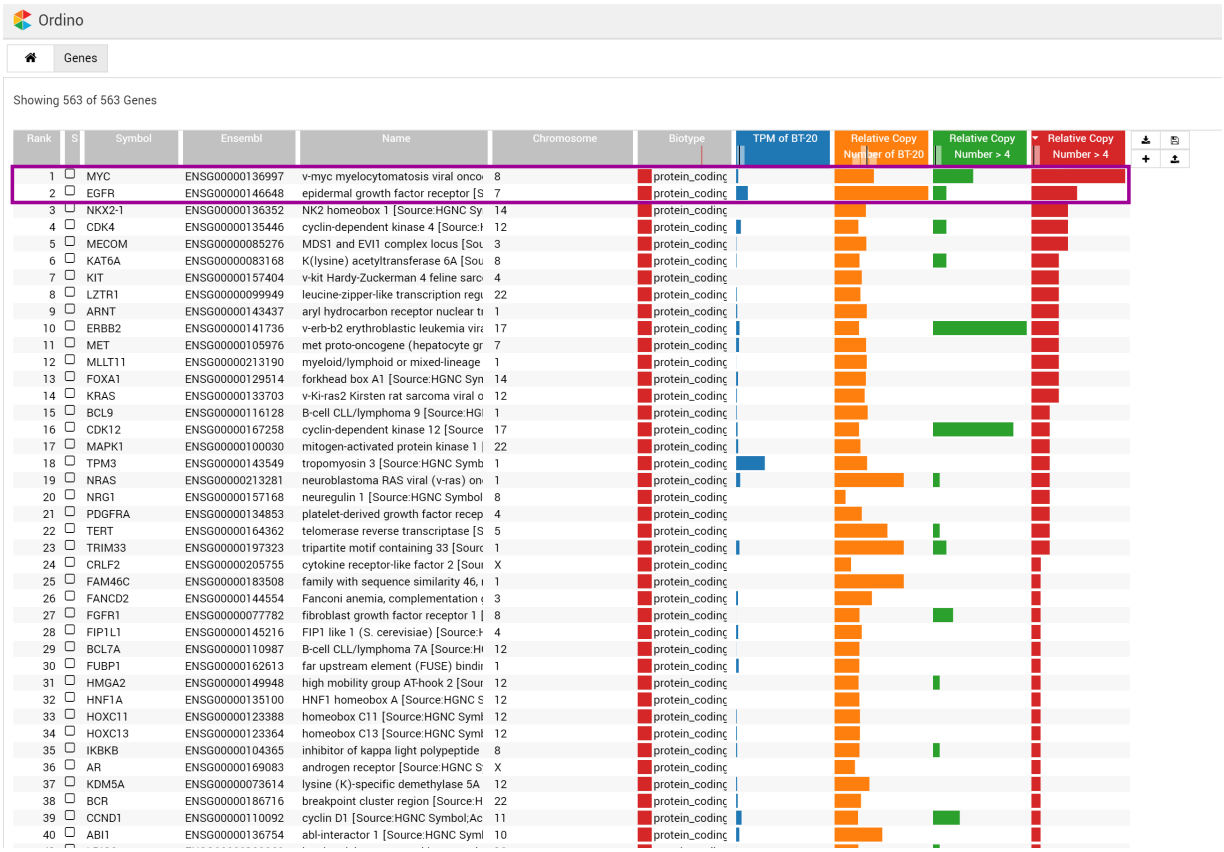


Figure B.8.: With the next step the user wants to investigate, which genes are typically amplified in *non-small-cell lung cancer* (NSCLC)? He switches back to the initial gene ranking, adds another cell line score column with the parameters tumor type *NSCLC*; data type *Relative Copy Number*; aggregation *Frequency*; cut-off > 4, and sorts the ranking by this column. Interestingly, the gene *EGFR* is ranked second this time after the gene *MYC*.

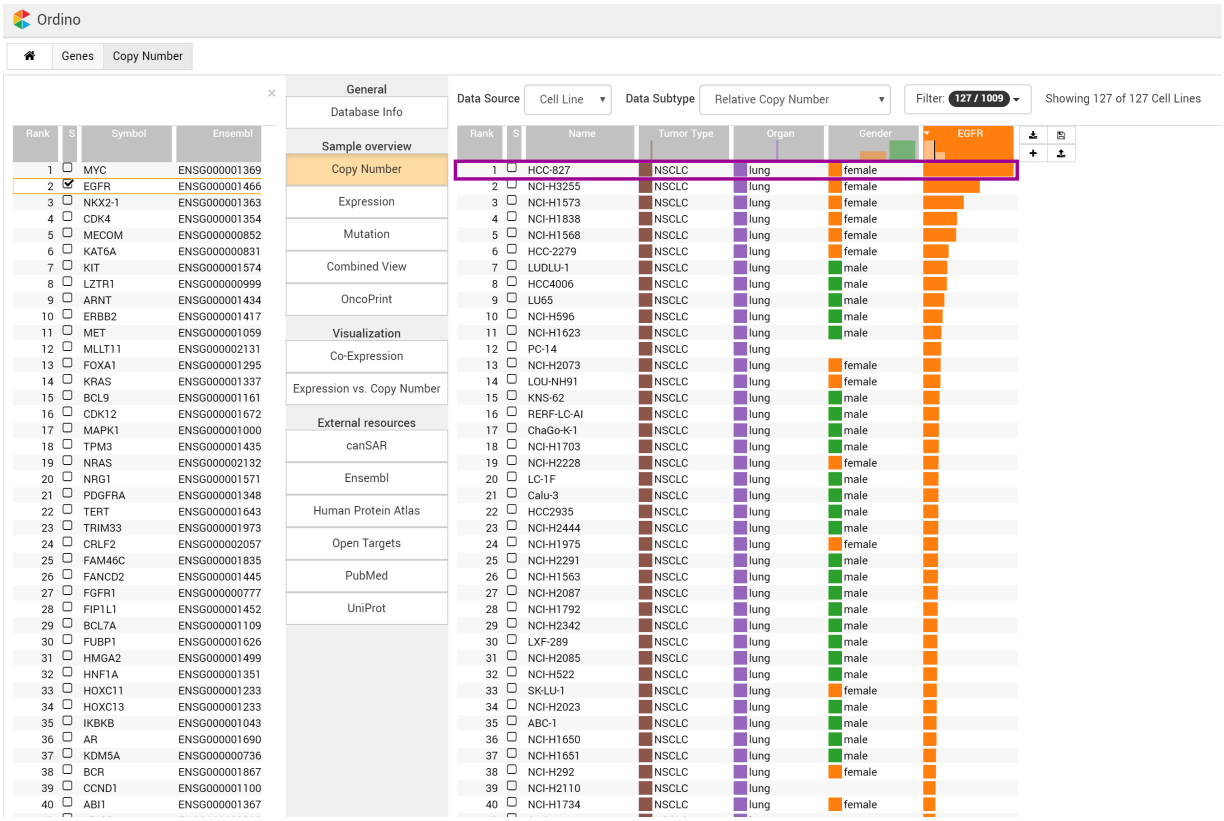


Figure B.9.: The user is still interested in *EGFR* and wants to know which *NSCLC* cell lines have an *EGFR* amplification? He selects the gene *EGFR* from the ranking and opens the *Copy Number* detail view. The copy number value for *EGFR* is added as additional column to the following cell line ranking. Additionally the ranking is filtered by *NSCLC* cell lines only and sorted by the relative copy number. The top hit is the cell line *HCC-827*.

B.3. Case Study: Search and Continuation of Analysis

The initial question is: which analysis states that are similar to the last one were done previously?

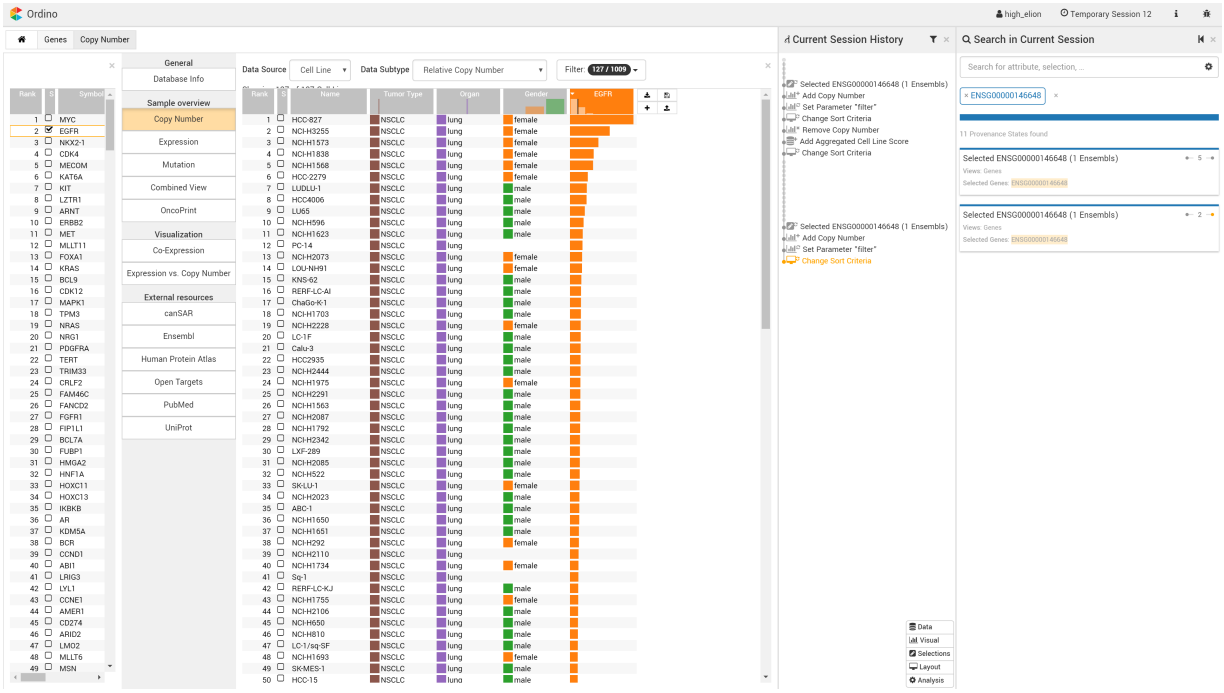


Figure B.10.: Filtering the search suggestions to properties of the active state and selecting the only gene *EGFR* with the identifier *ENSG00000146648* as search term results in two sequences where this gene was selected. When hovering over the search results the corresponding states in the provenance graph are highlighted.

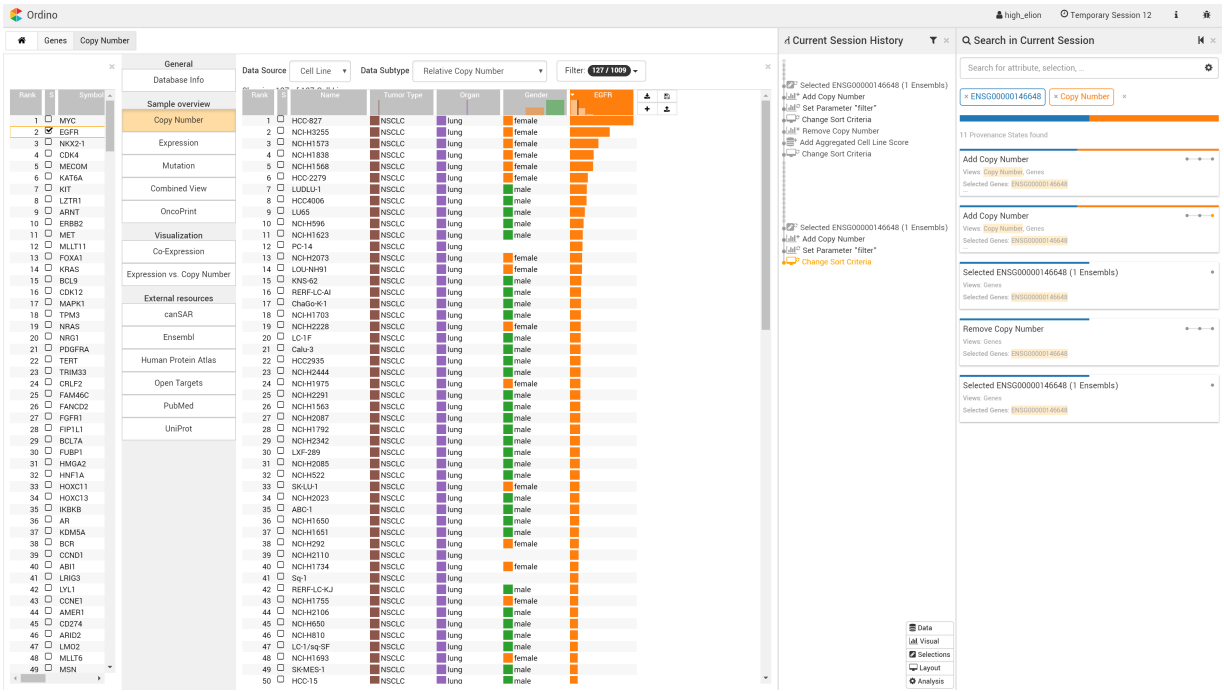


Figure B.11.: Refining the search with the *Copy Number* detail view results in five shorter sequences, where two of them are matching both search terms. The latter sequence contains the currently active state.

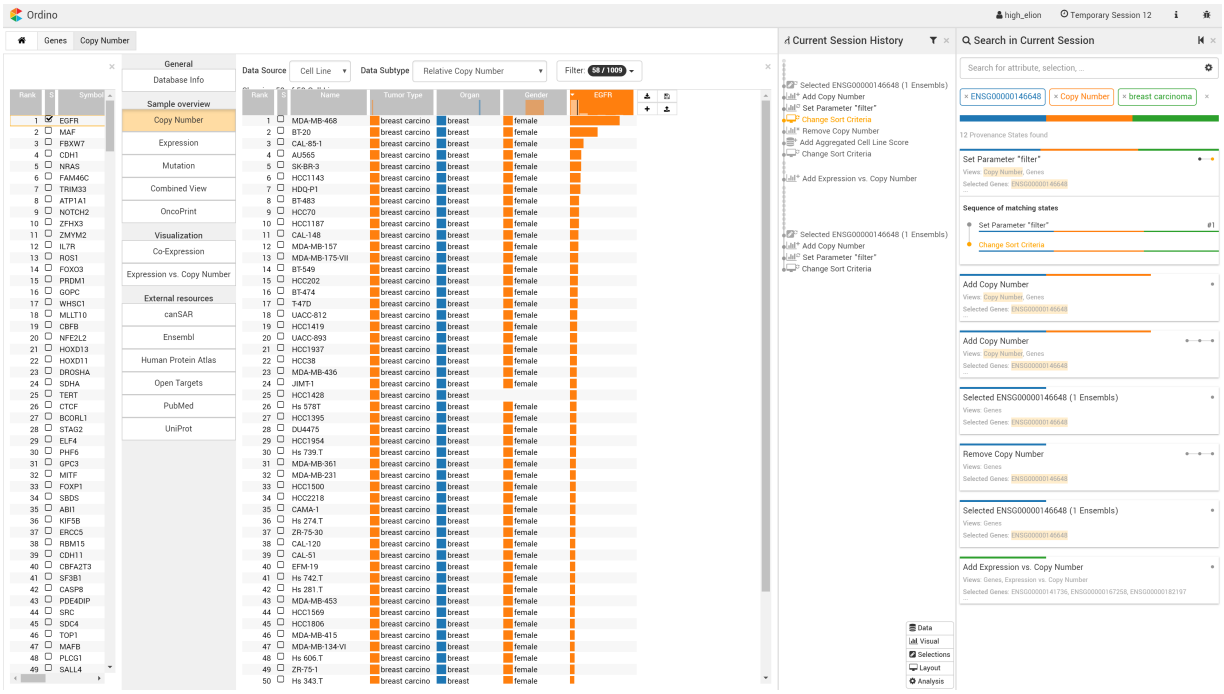


Figure B.12.: Refining the search results with *breast cancer* tumor type as third search term shows that only one sequence matches all search terms. Expanding the sequence and jumping to the last state shows that the *EGFR* copy number was assessed for *breast cancer* cell lines with *MDA-MB-468* as the top hit.

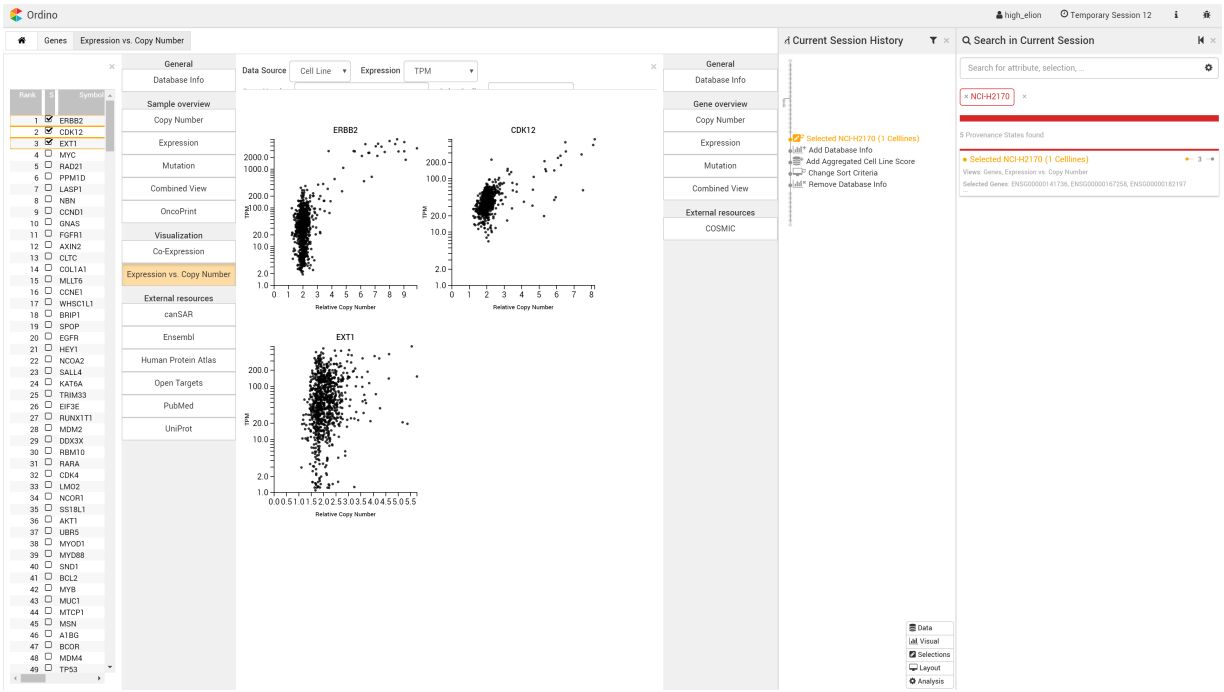


Figure B.13.: The user searches for the cell line *NCI-H2170* to recall if that cell line was used in the analysis. Indeed, the cell line was found in one state sequence and was selected in the *Expression vs. Copy Number* detail view.

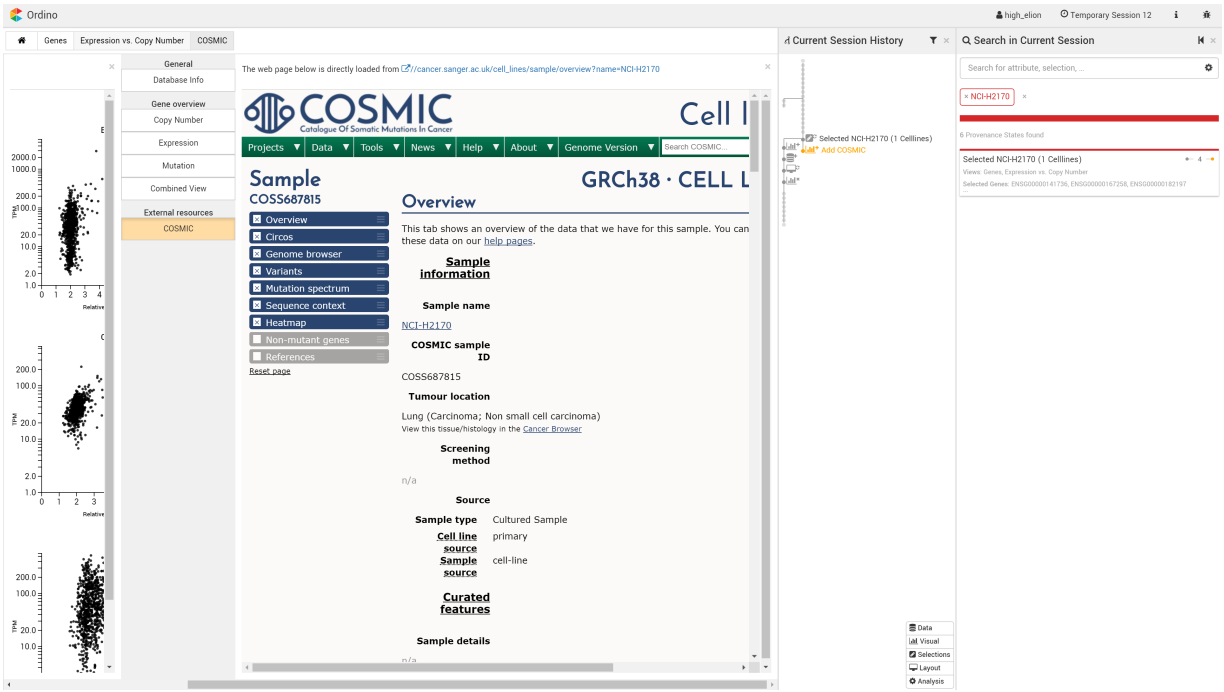


Figure B.14.: Following up with the analysis of cell line *NCI-H2170* the user opens the *COSMIC* detail view and browses the information that is available about this cell line in the COSMIC data base.

Curriculum Vitae

Personal Data

Name	**Holger Stitz**, MSc.
E-Mail	kontakt@holgerstitz.de
Website	holgerstitz.de

Professional

since 11/2013	Project Assistant at Institute of Computer Graphics, Johannes Kepler University Linz, Austria
since 10/2013	Lecturer at University of Applied Science Upper Austria, Hagenberg, Austria
10/2011 - 12/2013	Research Assistant at University of Applied Science Upper Austria, Hagenberg, Austria
10/2009 - 09/2010	Freelancer for Frontend Development at argonauten G2 GmbH, Berlin, Germany
04/2009 - 10/2009	Junior Web Developer for Frontend Development at argonauten G2 GmbH, Berlin, Germany
10/2008 - 03/2009	Student Assistant for Flash- and Web Development at argonauten G2 GmbH, Berlin, Germany
04/2008 - 09/2008	Internship for Flash- and Web Development at argonauten G2 GmbH, Berlin, Germany
03/2007 - 03/2008	Student Assistant for IT Administration at Hochschule Harz (University of Applied Science), Wernigerode, Germany

Education

since 03/2014	Doctoral program in Computer Science at Johannes Kepler University Linz, Austria Supervision: Univ.-Prof. Dipl.-Ing. Dr. Marc Streit
10/2011	Master's degree (MSc) from University of Applied Science Upper Austria, Hagenberg, Austria *with highest distinction* Thesis: *Website-Ontologien für semantische Content Management Systeme (CMS)* Supervision: Prof. (FH) DI Rimbert Rudisch-Sommer
10/2009 - 10/2011	Master's studies in Interactive Media at Graz University of Technology
03/2009	Bachelor's degree (BSc) from Hochschule Harz (University of Applied Science) Thesis: *Suchmaschinenoptimierung von Flash durch XHTML als Datenquelle* Supervision: Prof. Jürgen K. Singer, Ph.D.
10/2005 - 03/2009	Bachelor's studies in Media Informatics at Hochschule Harz (University of Applied Science)

Awards And Scholarships

2017	**Best Poster Award** IEEE Conference on Visual Analytics Science and Technology (VAST'17)
2015	**Best Poster Award** (of 81 submissions) IEEE Information Visualization (InfoVis'15)
2015	**Honorable Mention Poster Award** (top 4 / 81 submissions) IEEE Information Visualization (InfoVis'15)
2010, 2011	**Award for Excellent Performance as a Student** (Leistungsstipendium) Granted by the University of Applied Science Upper Austria, Hagenberg, Austria

Publications

Peer-reviewed Journal Publications

[1] Marc Streit, Samuel Gratzl, Holger Stitz, Andreas Wernitznig, Thomas Zichner, and Christian Haslinger. Ordino: a visual cancer analysis tool for ranking and exploring genes, cell lines and tissue samples. *Bioinformatics*, 2019.

[2] Holger Stitz, Samuel Gratzl, Harald Piringer, Thomas Zichner, and Marc Streit. KnowledgePearls: Provenance-Based Visualization Retrieval. *IEEE Transactions on Visualization and Computer Graphics (VAST '18)*, page 11, 2018.

[3] Christina Niederer, Holger Stitz, Reem Hourieh, Florian Grassinger, Wolfgang Aigner, and Marc Streit. TACO: Visualizing Changes in Tables Over Time. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '17)*, 24(1):677–686, 2017.

[4] Holger Stitz, Samuel Gratzl, Wolfgang Aigner, and Marc Streit. ThermalPlot: Visualizing Multi-Attribute Time-Series Data Using a Thermal Metaphor. *IEEE Transactions on Visualization and Computer Graphics*, 22(12):2594–2607, 2016.

[5] H. Stitz, S. Luger, M. Streit, and N. Gehlenborg. AVOCADO: Visualization of Workflow–Derived Data Provenance for Reproducible Biomedical Research. *Computer Graphics Forum*, 35(3):481–490, 2016.

[6] Werner Kurschl, Mirjam Augstein, Holger Stitz, Peter Heumader, and Claudia Pointner. A user modelling wizard for people with motor impairments. In *Proceedings of International Conference on Advances in Mobile Computing & Multimedia*, MoMM '13, pages 541–550. ACM, 2013.

Peer-reviewed Conference and Workshop Publications

[1] Holger Stitz, Samuel Gratzl, Michael Krieger, and Marc Streit. CloudGazer: A Divide-and-Conquer Approach for Monitoring and Optimizing Cloud-Based Networks. In *Proceedings of the IEEE Pacific Visualization Symposium (PacificVis '15)*, pages 175–182. IEEE, 2015.

[2] Werner Kurschl, Mirjam Augstein, and Holger Stitz. Adaptive user interfaces on tablets to support people with disabilities. *Mensch & Computer Workshop*, pages 91–94, 2012.

Posters

[1] Martin Ennemoser, Peter Ruch, Holger Stitz, Marc Streit, and Hendrik Strobelt. ConfusionFlow: Visualizing Neural Network Confusion Across Epochs. In *Poster Compendium of the IEEE Conference on Visual Analytics Science and Technology (VAST '18)*, 2018.

[2] Holger Stitz, Samuel Gratzl, Harald Piringer, and Marc Streit. Provenance-Based Visualization Retrieval. In *Poster Compendium of the IEEE Conference on Visual Analytics Science and Technology (VAST '17)*, 2017.

[3] Katarina Furmanova, Martin Ennemoser, Miroslava Jaresova, Samuel Gratzl, Bikram Kawan, Alexander Lex, Holger Stitz, and Marc Streit. Taggle: Scaling Table Visualization through Aggregation. In *Poster Compendium of the IEEE Conference on Information Visualization (InfoVis '17)*, 2017.

[4] Holger Stitz, Samuel Gratzl, Harald Rogner, and Marc Streit. Visual Evaluation of Cloud Infrastructure Performance Predictions. In *Poster Compendium of the IEEE Conference on Information Visualization (InfoVis '16)*, 2016.

[5] Reem Hourieh, Holger Stitz, Nils Gehlenborg, and Marc Streit. TaCo: Comparative Visualization of Large Tabular Data. In *Poster Compendium of the IEEE Conference on Information Visualization (InfoVis '16)*, 2016.

[6] Holger Stitz, Samuel Gratzl, Wolfgang Aigner, and Marc Streit. ThermalPlot: Visualizing Multi-Attribute Time-Series Data Using a Thermal Metaphor. In *Poster Compendium of the IEEE Conference on Information Visualization (InfoVis '15)*. IEEE, 2015.

[7] Stefan Luger, Holger Stitz, Samuel Gratzl, Nils Gehlenborg, and Marc Streit. Interactive Visualization of Provenance Graphs for Reproducible Biomedical Research. In *Poster Compendium of the IEEE Conference on Information Visualization (InfoVis '15)*. IEEE, 2015.

[8] Holger Stitz, Samuel Gratzl, Stefan Luger, Nils Gehlenborg, and Marc Streit. Transparent Layering for Visualizing Dynamic Graphs Using the Flip Book Metaphor. In *Poster Compendium of the IEEE Conference on Information Visualization (InfoVis '14)*. IEEE, 2014.

Linz, April 2019

Sworn declaration

I certify that this research thesis is the result of my own work, except as acknowledged, and has not been submitted, either in part or whole, for a degree at this or any other University.

The present thesis is identical to the document that has been submitted electronically.

Individual chapters of this cumulative thesis have been published as international conference articles and journal papers (be referred to [SGKS15, SGAS16, SLSG16, SGP$^+$18] for further details).

Linz, April 2019

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Dissertation selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die vorliegende Dissertation ist mit dem elektronisch übermittelten Textdokument identisch.

Einzelne Kapitel dieser kumulativen Dissertation wurden als Arbeiten auf internationalen Konferenzen und in Journalen publiziert. Die entsprechenden Artikel sind unter [SGKS15, SGAS16, SLSG16, SGP$^+$18] im Literaturverzeichnis gelistet.

Linz, April 2019