DOI: 10.1111/cgf.15003

COMPUTER GRAPHICS *forum*
Volume 43 (2024), number 1, e15003

Guided Exploration of Industrial Sensor Data

Tristan Langer, Richard Meyes and Tobias Meisen

Institute for Technologies and Management of Digital Transformation, University of Wuppertal, Wuppertal, Germany
{tlanger, meyes, meisen}@uni-wuppertal.de

Abstract

In recent years, digitization in the industrial sector has increased steadily. Digital data not only allows us to monitor the underlying production process using machine learning methods (anomaly detection, behaviour analysis) but also to understand the underlying production process. Insights from Exploratory Data Analysis (EDA) play an important role in building data-driven processes because data scientists learn essential characteristics of the data in the context of the domain. Due to the complexity of production processes, it is usually difficult for data scientists to acquire this knowledge by themselves. Hence, they have to rely on continuous close collaboration with domain experts and their acquired domain expertise. However, direct communication does not promote documentation of the knowledge transfer from domain experts to data scientists. In this respect, changing team constellations, for example due to a change in personnel, result in a renewed high level of effort despite the same knowledge transfer problem. As a result, EDA is a cost-intensive iterative process. We, therefore, investigate a system to extract information from the interactions that domain experts perform during EDA. Our approach relies on recording interactions and system states of an exploration tool and generating guided exploration sessions for domain novices. We implement our approach in a software tool and demonstrate its capabilities using two real-world use cases from the manufacturing industry. We evaluate its feasibility in a user study to investigate whether domain novices can reproduce the most important insights from domain experts about the datasets of the use cases based on generated EDA sessions. From the results of this study, we conclude the feasibility of our system as participants are able to reproduce on average 86.5% of insights from domain experts.

Keywords: analytic provenance, exploratory data analysis, guidance, sensor data, time series, visual analytics

CCS Concepts: • Human-centred computing → Visual analytics; • Computing methodologies → Knowledge representation and reasoning; • Mathematics of computing → Exploratory data analysis

1. Introduction

When building a data processing pipeline, data scientists spend a large portion of their time in a project collecting, exploring and preparing the data, and optimizing the data quality for the application of a machine learning algorithm [Boe16]. In doing so, the data scientist starts to collect and explore a dataset to find interesting insights and discover knowledge within the dataset by performing visual interactive exploratory data analysis (EDA). During this process, data scientists use visual systems to examine the data to find patterns and trends, detect anomalies (e.g. outliers), and check the validity of hypotheses within the data. The open-ended and creative nature of this process allows data scientists to dive deeper into the data at their own discretion to learn about the characteristics of the data and especially the associated domain [LS10]. Thus, EDA is an important step at the beginning of the data science process. The generated insights mainly influence subsequent tasks during data preparation such as data cleansing, feature engineering and labelling. However, EDA is known to be a difficult process since the quality of findings depends on the analyst's ability to ask the right questions and to translate these questions into the right sequence of analysis interactions within the data visualization system [LDC13]. The degree of difficulty in building up a data understanding also varies depending on the complexity of the domain and the dataset. A domain with an increasing demand for data-driven optimization and high process complexity is the manufacturing industry where even methodically skilled data scientists face the particular challenge of understanding complex production processes and have to deal with large volumes of diverse sensor datasets. Therefore, due to the complexity of the domain and special domain knowledge required to understand the data, data scientists have to rely on the expertise of in-house domain experts [WJL*15, AABZ20]. However, these experts are usually a very limited resource in companies and have a variety of tasks, where teaching data scientists about domain

characteristics is not necessarily one of the core activities. This results in expensive iterative information exchange between data scientists (domain novices) and domain experts. In collaborative exploration, the domain novice generates new insights, and knowledge is transferred from the domain expert, but it is not externalized. This means that if one of the two is no longer available, for example because they leave the project, knowledge transfer and information exchange are not preserved. When new data scientists join the project, they cannot benefit from this process and gain insights from past EDA sessions.

In this paper, we present a method to extract information in interactive visualization systems that domain novices need to generate insights about a dataset from the way domain experts explore data. We thus show that it is useful to record interaction data during EDA with a visual interactive system in order to make subsequent EDA sessions more efficient and externalize domain knowledge. To determine how domain experts proceed at EDA and what they find interesting, we record their exploration sessions and derive statistically relevant sequences of views. We combine these sequences into guided EDA sessions for domain novices and highlight particular regions of interest. Along two real-world use cases, we aim to demonstrate the feasibility of domain novices reproducing some of the insights provided by domain experts about industrial sensor datasets using EDA sessions created in this way. By demonstrating the viability of our system design, we aim to lay the groundwork for further community research on guidance in visual EDA systems, with a particular focus on optimizing individual components. One of the most critical aspects of this is to explore how generated EDA sessions can be effectively integrated with advanced visualization systems to provide domain knowledge to data scientists. Methodologically, we proceed as follows: In Section 2, we give a brief overview of related work. Based on two real-world exemplary use cases from the manufacturing industry (Section 3), we design a conceptual guidance system (Section 4) based on expert EDA sessions. To test the feasibility of our concept, we implement the system as a prototype by combining two provenance tracking solutions: we record the states that this tool goes through during EDA by experts in Section 5.1 and we record the interactions that each expert performs in the process in Section 5.2. We proceed with the development of a guidance generation algorithm (Section 5.3) that extracts statistically relevant action sequences and regions of interest from the collected data to generate a representative EDA session. Finally, we evaluate our system by collecting interaction data from experts for the two real-world use cases, generating guided EDA sessions from the collected data, and evaluating them in a user study with domain novices (Section 6). In summary, our contributions are (1) a conceptual system design to extract guided EDA sessions from interaction data of historical EDA sessions that were performed by domain experts on visual interactive systems, (2) an initial prototype implementation of the conceptual system design, and (3) the evaluation of the prototype with 23 data scientists to test the feasibility of the system and discuss limitations as well as the research path forward.

2. Related Work

Recently, the research of tracking, understanding and utilizing analytic reasoning processes has become an increasingly important research topic known as *Analytic Provenance*. For an extensive review of recently published work in analytic provenance, we refer to the comprehensive survey papers by Ragan et al. [RESC16] and by Xu et al. [XOW*20]. In the design of our approach, we rely on capturing user interactions and generating guidance for EDA based on the collected data, therefore, we give a brief overview of relevant research in both areas.

2.1. Interaction-based recording

Probably the most direct way to record user interactions within a web application is to record the mouse and keyboard interactions that are performed by a user. Such approaches have also been used before for desktop applications. For example, *Mousetrap* [KHW*19] by Kieslich et al. is a software tool for capturing mouse interactions on the desktop with a focus on user studies. The data can be used to identify the type of user, as shown by Boukhelifa et al. [BBT*19], or patterns in user behaviour, as shown by Barczewski et al. [BBB20], in which interactions are grouped into larger semantic blocks. This shows that it can be useful to record interactions such as mouse clicks and keyboard interactions, as they can provide information about the user's state, such as specific regions of interest or confidence during decision making. Langer et al. [LMM22] developed *Gideon Replay*, a library that allows the recording and analysis of user interactions for selected components of web applications. Most of the related work we present in this section also contains user studies that indicate recurring patterns in the way users interact with analysis systems and, thus, enable extraction and prediction of these patterns. Notably, the work of Battle and Heer [BH19] indicates that this fact also applies to EDA using visual analysis tools.

2.2. Application state-based recording

An alternative user tracking method is to store each action and state of an application interface while a user interacts with it. In contrast to the interactions from Section 2.1, the actions are not individual application-independent mouse clicks or keyboard inputs, but application-specific events triggered by the user that transfer the application from one state to the next one, e.g., applying a filter to a chart transforms the original chart to the chart containing only the filtered data. For example, Moritz et al. [MHHH15] present a tool that logs application states in order to profile query execution times and tune query plans in distributed systems. Stitz et al. [SGP*18] developed a tool called *KnowledgePearls* that presents a query system for retrieval of application states and visualizations from a collected provenance dataset. In addition to these solutions that are integrated into specific applications, there are also efforts to create more abstract libraries that enable application state-based tracking of user interactions in any web application. *SIMProv* by Camisetty et al. [CCSK18] is a JavaScript library for capturing provenance in GUIs of web applications. Thus, enabling users to replay and revisit steps of their sensemaking process. *Trrack* by Cutler et al. [CGL20] uses a method that stores the difference between the current and last state. Trrack is therefore more memory efficient and also periodically stores snapshots of the whole application state to ensure fast recovery of states. Trrack encapsulates application-specific information and has been released as a library,

making it accessible for developers to integrate into their own applications.

2.3. Guidance for exploratory data analysis

There are a number of systems that apply statistical measures to find interesting data points and recommend them for visualization (e.g. [THY*17, LKL*18, CBYE19]). One of the most prominent examples is Voyager [WMA*16], a mixed-initiative system that supports faceted browsing of recommended charts. It allows users to explore dataset dimensions and suitable visualizations, and then automatically recommends views related to the current user specified chart based on statistical and perceptual measures. In addition, there are systems that deal with generating entire EDA sessions or proposing EDA actions. Kim et al. [KWHH17] developed *GraphScape*, a system embeds the state of an EDA system into a graph and applies a cost function to evaluate paths through the graph. Using the cost function, it is possible to generate connected, easy to follow EDA sessions through a dataset. A similar approach can be found in Bar El et al. [BMS20] who use deep reinforcement learning for the generation of whole EDA sessions as Python notebooks. They describe a reward function based on measures for interestingness, coherency and diversity, which an agent uses to learn to generate informative EDA notebooks. Zhao et al. [ZXC*21] recommend the partitioning of charts previously generated by users and then orders them into sequences of comic layouts. Thus, generating full EDA sessions from those charts. In contrast to the previous approaches presented, *REACT* by Milo and Somech [MS18] uses historical interaction data from experts. From the data, REACT abstracts interaction types and the context, that is the state of the EDA system. For new EDA sessions, REACT then searches for similar states within the historical data using a distance function to obtain interaction candidates to suggest next step recommendations.

Related work that deals with recording interaction has already produced reusable open source libraries that can be integrated into web-based EDA applications. Other related work on guiding EDA shows approaches for generating and visualizing data views. However, current approaches mainly rely on metrics, like interestingness and coherency, to construct recommendations that do not necessarily correspond to the insights that an expert would provide in a collaborative EDA session. We therefore develop a complementary approach that benefits from previously conducted EDA sessions with related work systems to efficiently guide future sessions. It generates entire EDA sessions based on historical interaction data from experts and thus also shows insights that are less likely to be mapped by a measure.

3. Exemplary Use Cases

In the following, we present two real use cases that we worked on together with experts from the manufacturing industry. We use them as representative examples to show the feasibility of our system. Both use cases contain large amounts of data from multiple sensors, whose interpretation requires an understanding of the underlying production process. We utilize the use cases for the development of our guidance system in Section 5 and particularly for the evaluation of our system in Section 6. We briefly present them in order to make our design and evaluation process more comprehensible to the reader.

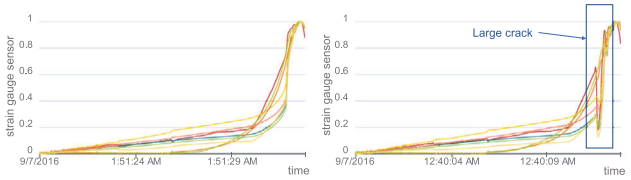

Figure 1: *Data from two exemplary strokes of the deep drawing tool with data from eight sensors each (distinguished by colour). Left: good stroke of the deep drawing tool with a smooth progression of strain gauge sensor curves. Right: erroneous stroke of the tool with a sudden drop of the values of some of the strain gauge sensors caused by a large crack in the metal sheet.*

3.1. Deep drawing of metal sheets

Deep drawing is a sheet metal forming process in which the mechanical force of a punch draws a sheet metal blank radially into a forming die [DIN03]. Our data comes from a deep drawing tool that is used to make the tailgate of a car from a metal sheet. In order to digitally record the course of a stroke, the tool was equipped with eight strain gauge sensors at its blank holder. Each sensor contains a strain sensitive pattern that measures the mechanical force exerted by the punch to the bottom part of the tool. The deformation of the metal during a stroke sometimes causes cracks in the metal sheet resulting in a brief loss of pressure to the bottom of the tool and thus a sudden drop in the values of the strain gauge values [MDSM19]. Broken sheets can be replaced and lead to defective material. However, when a crack occurs, the punch directs its force to the bottom of the tool instead of the metal sheet. The force of the impact can damage the tool, which then has to go through a cost-intensive repair and production downtime. Therefore, cracks should be detected at an early stage based on the sensor data. Our dataset is pre-segmented by strokes and entire segments have already been labelled with clean, small crack, large crack, and configuration. Figure 1 shows two example sensor value curves: a curve of a good stroke on the left where the progression of the sensor values is smooth and no crack occurs, and a curve of a bad stroke on the right where there appears a sudden drop in sensor values caused by a crack. Our dataset consists of 3400 strokes with the values of eight sensors, each containing about 7000 readings per stroke. According to domain experts, in order to accurately label the dataset, a data scientist needs to understand what forms of normal curves exist, what cases of error can occur, how severe each case is, which curves represent invalid strokes, and which sensors are more relevant than others because of their placement on the tool.

3.2. Metal arc welding

Metal arc welding is a fusion welding process in which electric gas is used as an energy carrier. An electrical discharge between two voltage-carrying electrodes in an ionized gas creates an arc of electric energy that performs the welding process in two phases. In the first phase, a low energy input melts the base material and filler

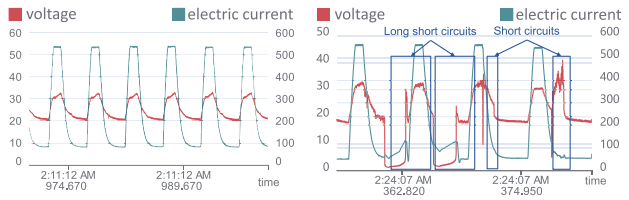

Figure 2: *Two example time intervals from the metal arc welding use case. Left: a normal progression of a welding process with a regular increase in electric current and voltage from the welding electrode. Right: a long short-circuit transition which may lead to bad quality of the weld seam.*

whereas the current density is not sufficient to detach droplets by the pinch effect. During the second phase high current density is used to initiate the droplet transition before it reaches a size that would trigger a short-circuit transition [KSM13]. Repeated execution of both phases results in a pattern that gives insights into the quality of the resulting welding seam as a long short-circuit phase may indicate a faulty welding process with a burned or deformed welding seam. Therefore, it is important to identify these short-circuit transitions in order to control process parameters or sort out defective goods. Figure 2 shows two example time intervals of welding runs. The dataset contains 32 welding procedures with about three million voltage and electric current readings per welding procedure. Conspicuous patterns, such as short-circuits, were labelled. According to domain experts, in order to accurately label the dataset, a data scientist needs to understand that the process runs in cyclic phases, what anomalies and short-circuits look like and that long short-circuits can lead to bad welding quality.

4. Conceptual System Design

For the conceptual design, we take one of our earlier works [LM21] and refine the concept for the manufacturing domain using the framework for guidance designers by Ceneda et al. [CAA*20]. The framework defines that we first identify the **analysis goals** of our users (here: data scientists) and the **knowledge gap** that hinders the users from achieving their goals. We then design an appropriate **guidance system design** that will help them to overcome these gaps.

During exploration, data scientists have the **analysis goal** to learn the most important characteristics of a dataset in the context of the domain. However, it is not clear what these most important characteristics include, as only domain experts are able to judge this. Data scientists in the manufacturing industry usually have limited domain knowledge regarding production processes. This is because they are methodologically trained in data science methods, a field in computer science, but not in manufacturing techniques, a field in mechanical engineering. Manufacturing processes require specialized technical knowledge and skills that can take years of training and experience to acquire. Additionally, they are often highly complex, involving multiple stages and a variety of inputs and outputs [Klo11]. Data scientists are therefore most often domain novices in the field of manufacturing techniques and a **knowledge gap** arises that hinders them in their analysis goal. Thus, they do not know the key insights that are required to analyse a specific production process and, consequently, they do not know the sequence of actions they need to take to learn those insights from the data. Hence, we frame the knowledge gap in EDA as a data and task knowledge gap. A data knowledge gap describes a lack of knowledge about the data, for example relationships between variables, whereas a task knowledge gap indicates a lack of procedural knowledge, that is which sequence of actions (like selecting or filtering data) to execute in order to achieve the analysis goal [CAA*20]. According to the framework, we have to choose between a *knowledge gap interface* where the users actively query the system for guidance and *knowledge gap inference* where we infer the need for guidance from the users' behaviour. As we assume our users are aware of their knowledge gaps, we decide to supply a knowledge gap interface that our users can actively query to receive guidance.

Because domain experts are a very limited resource in the manufacturing industry, we want to **design a guidance system** that guides the data scientist through the most important insights within a dataset, similar to the way a domain expert would explain them in a collaborative session. In doing so, our vision is to anticipate some of the insights that are provided in a collaborative exploration process between a data scientist and domain expert and, thus, reduce the time required for direct exchange in the future. This leads us to the question of how an expert would provide information about insights. According to Sacha et al. [SSS*14], insight is generated when domain experts perform exploratory data analysis by carrying out actions on a dataset and encounter a conspicuous phenomenon in the data that was previously unknown to them. They then hypothesize about the process responsible for the phenomenon and try to explain it or confirm the hypothesis on the basis of further actions. Thus, our goal must be to recommend not one single interesting view, consisting of one or more charts, but a whole sequence of actions that domain experts go through as they interact with an EDA tool to identify a conspicuous behaviour inside the dataset and understand the context for the explanation of this behaviour. The second aspect we need to keep in mind is that experts can go into more detail in a conversation regarding which regions are relevant in more complex multi-chart views. For example, which of the displayed data areas are particularly relevant for detecting a specific error. It is not possible to identify exactly on the basis of a single expert session which views are relevant and which were simply viewed by an expert without generating any insight. However, the related work in Section 2 has shown that there are repeating patterns in the way experts explore data and that it is possible to extract these patterns statistically. Consequently, we develop a semi-automated approach that uses historical data from expert EDA sessions to generate a representative new EDA session consisting of a sequence of statistically relevant views as well as regions of interest inside those views.

Figure 3 shows a high level overview of our concept for such a system. It is structured into three areas: guidance input, guidance generation, and guidance output which we explain in more detail in the following.

Guidance input For the collection of data that we can use for guidance input, we assume that exploration and labelling of the data are to be initially performed either by a domain expert or a data scientist with the assistance of a domain expert, as in current practice. Now we have to decide what options are available to us to capture those experts' reasoning processes. As discussed in Sections 2.1

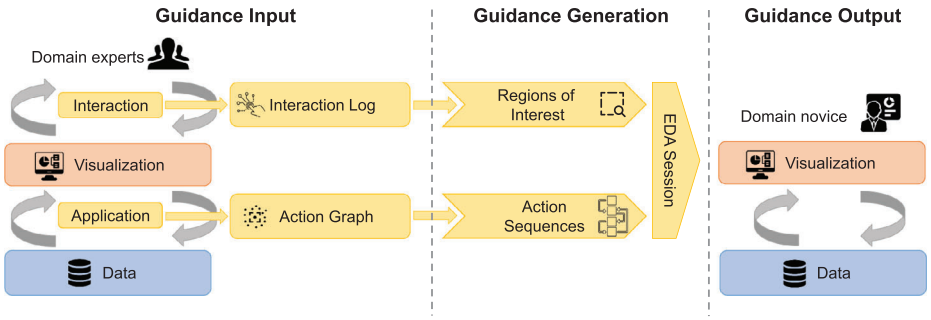

Figure 3: *Overview of our conceptual system design to generate sensor data exploration sessions from interaction data of domain experts.*

and 2.2, on the one hand, we can capture the actions and state of the EDA tool with which experts interact. On the other hand, we can capture mouse interactions of domain experts within the visualizations of each state—that we call *view*—to infer which regions of a view were of particular interest to the experts. For this purpose, we introduce the *Interaction Log* component and the *Action Graph* component in our conceptualization. The action graph component is responsible for storing all tool states (e.g. empty chart area, one chart in chart area with data from sensor one, etc.) and actions from the tool specific action space (i.e., actions that can be triggered by buttons and other interactive elements) performed within the tool (e.g. creating a chart by dragging data from sensor 1 into the empty chart area). This results in an action graph $AG = (S, A)$ with S is a set of tool states s_i defined by parameters $s_i = p_1, \ldots, p_n$ that describe a unique state of this tool, and $A \subseteq (x, y, t)|(x, y) \in S^2, x \neq y, t \in T$ a list of actions that transfer the tool from one state x to the next state y with t being the type of action from all action types T. The interaction log component stores all interactions performed in state s of the action graph AG with $UI_s = i_1, \ldots, i_n$ with $i_m \in I$ and I is the set of valid interactions with the tool, that is clicks, mouse movement, scrolling, and keyboard interactions. For each state stored in the action graph AG, this allows us to track exactly what the domain expert did within the associated view, in which regions the mouse was located the most, and which data points were viewed in detail. We call the combination of the action graph and interaction log *interaction data* in the following.

Guidance generation From the collected data, we generate statistically relevant *Action Sequences* and *Regions of Interest*, as discussed at the beginning of this section. Since we only have the collected action graph, interaction logs, and the original sensor dataset available, we choose a semi-automated unsupervised approach. We need to transform the action graphs from all experts AG into individual segments of action sequences and cluster them into groups of frequently occurring patterns using a similarity measure. Interesting areas within the views of a state are identified on the basis of the interaction logs UI_s by identifying and highlighting regions with a high density of mouse interactions. These are regions where experts interacted most with the data and looked at details, thus, we can assume that this is where most of their attention has been because of an interesting phenomenon within the data. Finally, these candidates are ranked based on a metric for how early the corresponding sequence occurred on average in the original action graph.

Guidance output The output of our guidance system for EDA is a session of sequences of views, each visualizing an application state, through which domain novices learn the most important characteristics of a dataset. The task of the output component is to communicate the generated EDA session to the data scientist via appropriate visualizations to assist the data scientist in identifying interesting data points within the context of the domain, that is the concrete target is unknown to the data scientist. Based on the sequences and regions of interest identified by the guidance generation, a visual EDA notebook needs to be generated in such a way that the information is visualized and presented in an understandable way to the data scientists. Since we are generating complete sessions, we assume that it makes the most sense to create a separate visualization component that can be used to follow all sequences within a generated EDA session. This allows the session to be quickly available and prevents novices from having to construct their own visualizations. Instead, they can quickly consume the prepackaged sequences and be efficiently guided through the insights identified by experts. With regard to the guidance degrees defined by Ceneda et al. [CGM*17], our system is therefore a prescribing guidance system with the highest degree of guidance, which guides the user almost automatically through the most important findings of a dataset. This completes the conception of our system. In the following, we present details of the prototypical implementation of the individual components and then evaluate them in a user study to demonstrate the feasibility of our system (see Supporting Information).

5. Generating Exploration Sessions from Historical Interaction Data of Domain Experts

We designed an exploration prototype that we use to collect interaction data from domain experts with a Python backend and an Angular frontend. For a detailed description of the design decisions and the source code, we refer to the corresponding publication [LWM22]. Figure 4 shows the design of user interface components. The components of the view are based on common components needed for interactive exploration. On the left side (1) we implement a project overview as a tree view containing segments (here: deep drawing strokes) and dimensions (here: sensors). On the top right (2) is an overview of the complete dataset. In the case of a pre-segmented dataset, like our deep drawing dataset, the segments are displayed concatenated. Below that, also on the right side, is a freely configurable chart area (3). In this area, users create charts

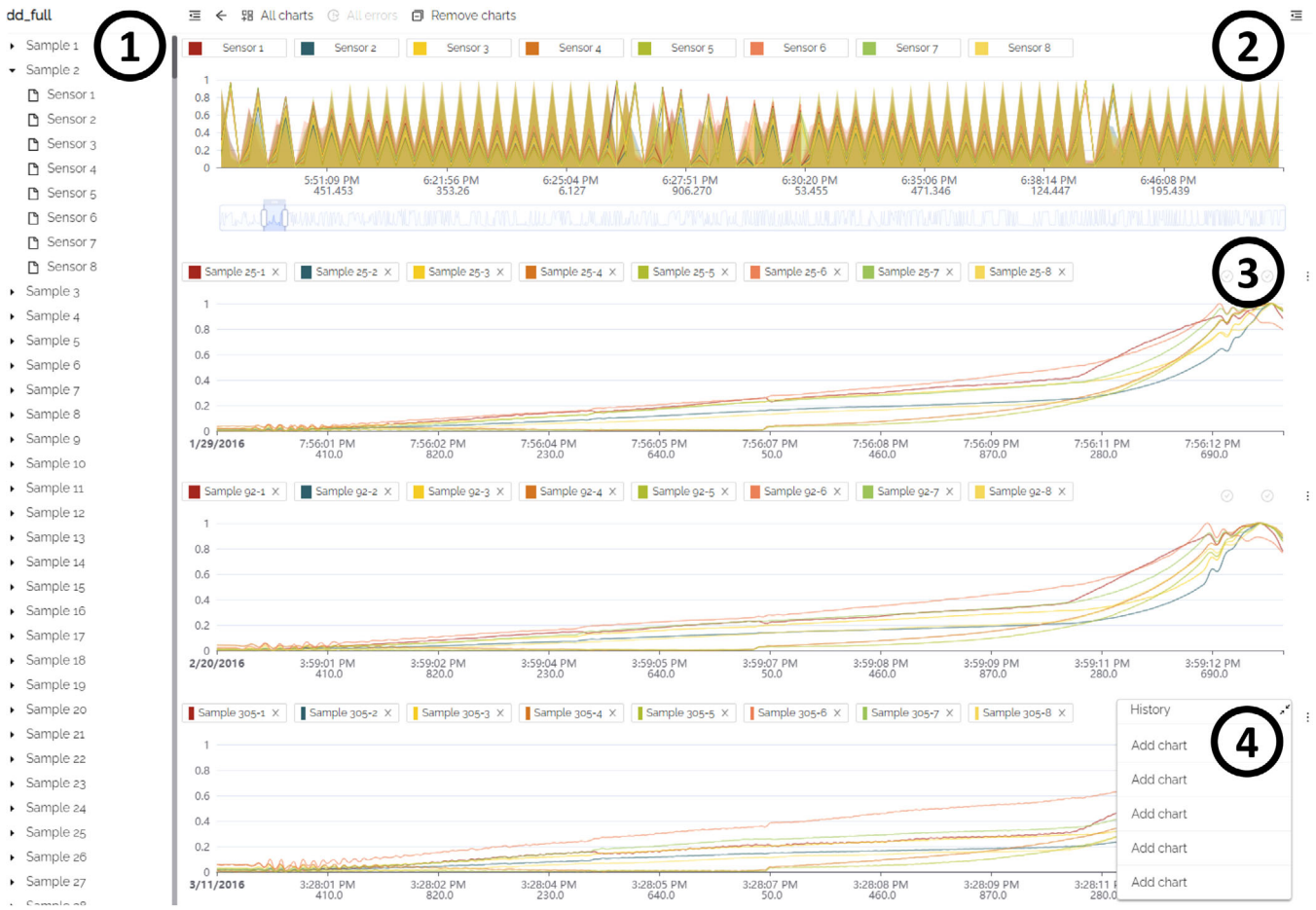

Figure 4: *Overview of the exploration interface of our prototype EDA system: (1) project structure to select sensor segments and dimensions, (2) aggregated overview of all sensor data with a data zoom, (3) configurable charts visualize individual segments and sensors, and (4) history of exploration actions.*

by dragging and dropping the segments and dimensions from the project overview area. During dragging, available drop zones are highlighted above or below the existing charts. Each chart also contains its own legend to temporarily hide individual segments or dimensions. In addition, by selecting an area or scrolling with the mouse wheel, users are able to adjust the zoom level or zoom into certain regions of interest. This allows users to freely explore and compare the dataset, for example two segments that contain similar errors. The last element is a history component (4) where users can track their previous EDA actions and thus track their EDA progress.

A large number of data points to display in a single chart leads to performance and visualization problems. To solve this we apply a strategy that aggregates the data in intervals to make it easier to comprehend. Additionally for displaying many elements in a single display in the browser, we rely on the virtualization of components. When virtualizing list components, such as tree views or list elements, the data, such as chart configurations, are still kept in the browser, but only the elements that are actually in the users' visible area of view are rendered. For the presented interface of our EDA prototype, we use the virtual scroll components for the tree views (project overview) and the freely configurable area of the charts to avoid performance issues. This allows us to support large trees (e.g. overview of projects with many samples and dimensions) and a large number of charts.

5.1. Guidance input: Tracking application states

To enable tracking in our system, we first need a way to record the states of our EDA prototype. In the related work, we have seen that the Trrack [CGL20] library can record the states of web applications, is publicly available, and therefore can be integrated into our prototype (cf. Section 2.2). In Trrack, actions that change the state of an application and the parameters that describe the state of the application must be registered. While the user interacts with the application, an action graph is generated that contains the list of performed actions and the states passed through. If a user returns to a state that he or she has already visited, a new branch is created from the corresponding application state node of the graph. We register the following actions to represent states within the capabilities of our prototype: *add chart*, *remove chart*, *reset (i.e. remove all*

charts), *pan or zoom chart*, and *filter (i.e. select or deselect sensors from the legend)*. For each state, we capture all the charts created by the user, the sensor data they contain, labels, and the current pan x and y coordinates, as well as the zoom level. We also create an interaction log for each chart, which we describe in more detail in the following Section 5.2. The state of our application is described by the following data model:

```
class State {
  id: UUID;
  charts: Chart[];
  // image preview for the history
  preview: string;
}

class Chart {
  // multivariate sensors
  data: number[][];
  labels: Label[];
  panZoom: PanZoom;
  events: InteractionLog[];
}

class PanZoom {
  // pan x and y
  x: number;
  y: number;
  // zoom level
  zoom: number;
}

class Label {
  label_class: LabelClass;
  start: number;
  end: number;
  sensor: number;
  dimensions: number[];
}

class LabelClass {
  name: string;
  severity: 'okay' | 'warning' | 'error';
}
```

5.2. Guidance input: Tracking interaction logs

Now that we are able to track which states our application goes through during an EDA session by a domain expert, we are still interested in what interactions (mouse clicks and movement) the domain expert performs on the resulting views. We use the Gideon-Replay [LMM22] library to record these interactions as it is published as open source library and therefore can be integrated into our prototypical EDA application as well (cf. Section 2.1). In Gideon-Replay, individual web components can be selected, for which a separate interaction log is created. Gideon-Replay captures all mouse, scroll, and keyboard interactions performed within this web component. We register each user-created chart component of our prototype for each state from the data model of the previous section, as we are interested in what interactions are performed on each individual chart in this specific view state.

5.3. Guidance generation: Action sequences and regions of interest

For our guidance generation, we want to extract frequently occurring action sequences from the action graphs (application states) and add regions of interest from the interaction data (mouse clicks and movement) as described in Section 4. For this purpose, we develop an algorithm that identifies frequently occurring patterns through clustering and combines them into a notebook. Figure 5 gives an overview of the general procedure of our algorithm: (1) expert exploration datasets (action graphs) are transformed into an abstract representation, (2) the action graphs are segmented into single sequences, (3) short sequences and noise are filtered out, (4) sequences are clustered using a custom sequence distance, (5) the most representative sequence is calculated for each cluster, and (6) the abstract sequence is mapped back to the original dataset. In detail, the algorithm proceeds as follows:

(1) We receive recorded exploration sessions as action graphs from domain experts as input, as described in the previous sections. However, we cannot use action sequences the way they occur in the collected data directly, since they contain very specific data points (e.g. create a chart with segment 2312, which shows a clean stroke; create a second chart with segment 912 next to it, which shows a stroke with a crack). The data-specific sequence occurs relatively rarely over all action graphs, while an abstract version of it, that is comparing a clean stroke to a stroke with a crack, is more common. Therefore, we transform all sessions into an abstract representation, where we remove all data specific attributes. This means that we remove specific data references and action parameters like filter ranges. We then replace timestamps with continuous numbers and convert concrete value ranges for the labels and pan zoom coordinates into local coordinates (e.g. a label start timestamp is converted to a relative 30.87% value that states the relative label range within the shown data). This way, for example, two states, each containing a stroke from our deep drawing use case with data taken at different times, become the same state. Each abstract state also receives a reference to the original data specific state in order to be able to retrieve it later in step (6).

(2) We want to compare which sequences occur most frequently over all action graphs. Therefore, we segment the full action graph into smaller semantically coherent segments that we can more easily compare for similarity. To do this, we pay particular attention to the points where a specific reset action is performed or where there is a difference in the frequency of activity. Thus, we perform the segmentation at the points where a reset action was performed (i.e. all charts are removed) or if there were a lot of mouse interactions with the charts within a view, followed by views with no interactions. That is, we calculated the active time from the interaction log, that is the time in which actions were executed. To decide on a specific action time for segmentation, we follow well-established design guidelines such as the *Visual Information-Seeking Mantra* by Shneiderman [Shn96] or the *Visual Analytics Mantra* by Keim

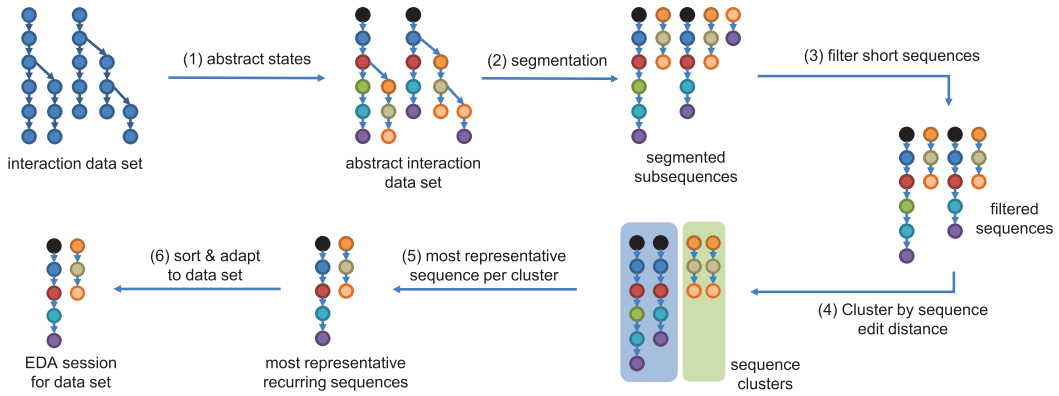

Figure 5: *Overview of the procedure of our guidance generation algorithm.*

et al. [KMSZ06]. At this point, the expert has probably either found an answer to the current question or will try another way in the following; either way, the current mental sequence ends as the user explored details followed by going back to an overview state. We chose a value based on the reference times for common tasks (2–4 s) from Shneiderman [SPC*17] and split the sequence if there was an active time of above 4 s followed by a state with less than 4 s of active time. Finally, we save the position of each segment in the original action graph (number of links to the root node), to rank the segment in the final ordering of the generated EDA session in step (6).

(3) After segmenting the graph into individual sequences, we get many sequences that have been created by free exploration, where experts have looked at data but have not generated insights. We want to filter these out so that they do not end up in the final notebooks. We filter out sequences that contain fewer than three actions since we assume that these are not relevant or even noise as they are too short to pursue longer strands of thought or to involve following up on a specific question. We also filter sequences in which the user was active for less than 4 s. Again, with respect to the aforementioned mantras, we assume that a sequence in which no details were viewed does not contain any interesting findings.

(4) In the next step, our aim is to detect recurrent patterns within the filtered sequences. However, the patterns may not be entirely identical. Depending on the approach of the domain experts, they will consist of slightly divergent sequences, which vary in certain intermediate steps while maintaining similar overall states. Therefore, we choose a clustering-based approach with a custom distance function to identify clusters of similar sequences. This has the advantage, for example, in comparison with sequential pattern mining methods [MR13], that we have a representative selection of frequently occurring sequences. In addition, our method ensures that coherent sequences are identified, since they were originally performed this way in the EDA tool. This ensures that we do not get invalid sequences, such as two consecutive states where the second state cannot be reached by any valid action in the tool from the first state. For this purpose, we use DBSCAN as a clustering method together with a custom sequence edit distance. DBSCAN is a density-based clustering method, that identifies core samples in areas of high density and creates clusters around these samples. DBSCAN takes *EPS* and *min samples* as input parameters. Samples within EPS distance from any core sample are assigned to the respective cluster. It also identifies outliers, where an outlier is not within EPS distance from any of the resulting clusters [EKSX96]. The *min samples* parameter describes the minimum amount of data points in proximity of a data point to be considered as a core sample. DBSCAN also allows us to use a pre-calculated custom distance matrix of all sequences to identify the clusters. We compute the custom edit distance between the input sequences x and y, given the state-wise distance function $dist(s_1, s_2)$ [PMH15]. This function finds an alignment $M = (i, j) \subset 1, \ldots, len(x) \times 1, \ldots, len(y)$, such that the following loss is minimized:

$$\sum_{(i,j)\in M} dist(x_i, y_j) + \sum_{i\notin M} dist(x_i, -) + \sum_{j\notin M} dist(-, y_j) \quad (1)$$

where i is said to be not in M if there exists no tuple (i, j) in M for any j, and j is said to be not in M if there exists no tuple (i, j) in M for any i and the state-wise distance function:

$$\begin{aligned} dist(s_1, s_2) = c_w * \left| \frac{c(s_1) - c(s_2)}{max(c(s_1), c(s_2))} \right| \\ + l_w * \frac{1}{n} \sum_{c=1}^{n} \left| \frac{l_c(s_1) - l_c(s_2)}{max(l_c(s_1), l_c(s_2))} \right| \\ + d_w * \left| \frac{d(s_1) - d(s_2)}{max(d(s_1), d(s_2))} \right| \end{aligned} \quad (2)$$

where $c(state)$ returns the number of charts in a state, $l_c(state)$ returns the number of labels of label class c in a state and $d(state)$ returns the number of dimensions, that is sensors, in a state, and c_w, l_w and d_w are adjustable weights with $c_w + l_w + d_w = 1$. We thus define abstract patterns as similar if they have a similar number of charts, contained labels per class, and contained sensor data. We finally filter outliers and receive clusters of frequent abstract action sequences from all expert action graphs.

In our initial prototype, we performed exploratory laboratory tests to find a suitable initial parameterization for our algorithm, that is we tested different parameters until it reproduced

all frequent sequences in a laboratory test dataset. We found $c_w = 0.1, l_w = 0.45$ and $d_w = 0.45$ to produce accurate results which means that the number of charts was less important for the similarity than labels and sensor data. Since our sequence edit distance outputs values between 0 and 1, we set the initial EPS $= 0.3$ as a threshold value. We set the min samples parameter in relation to the number of input expert sessions. As we want to find sequences that occur frequently in these sessions, we set min samples $= \frac{number\ of\ sessions}{3}$, that is outliers are clusters of sequences that do not occur in at least one third of the input sessions.

(5) We have now identified clusters of frequently occurring patterns of action sequences from all the original action graphs. Now we need to select a representative sample to use for generating the final notebook from each cluster. For each cluster, we calculate the number of neighbours in the EPS range for each sequence using our custom distance function (step 4) and taking the sequence with the highest number of neighbours. This respective sequence forms the density centre of the cluster and is thus the most similar to all included action sequences.

(6) In the last step, we transform the abstract action sequences back into a sequence for a concrete dataset. In the simplest case, the dataset matches the original dataset, where we simply restore the original data of the most representative sample (from the reference stored in step 1). For applying abstract sequences to newly collected data from the same use case, we use Dynamic Time Warping [BC94] as a distance measure to find the most similar new sensor data and replace abstract attributes with the so identified most similar data. All parameters and relative timestamps are adjusted to the identified most similar data segment. Since a complete match with the original data is unlikely, users are informed that they will receive a derived sequence.

Finally, we need to bring the set of concrete sequences, that we get from our algorithm, into a unified notebook. To do this, we need to decide on the order in which the sequences are joined together. We rank the sequences according to how early, on average per cluster, the sequences occurred in the original action graphs (from the position stored in step 2). Thus, sequences that are performed earlier on average by experts are also displayed earlier in the final EDA session.

This enables us to create EDA sessions consisting of several statistically relevant sequences from the expert interaction dataset. In order to give domain novices an indication of which regions represent potentially interesting patterns when inspecting the session, we identify *regions of interest* from the mouse interactions that we recorded for each view state. To do this, we divide the visible area into evenly spaced bins and count the interactions within the bins. Then we filter out bins with few interactions and create a rectangular overlay defined as a triple of (origin, width, length) that includes all these bins. Figure 6 shows an illustrative example of this procedure where experts inspected a drop in sensor data leading to a blue box around this area. Thus, domain novices receive an indication from the box in the guided EDA sessions that this area contains potentially interesting patterns, as this area was of most interest to experts. A higher number of bins will result in a higher resolution, that is a more accurate area, but determining a good number to identify bins with many interactions becomes more difficult, as interactions are spread over more bins and, thus, there are fewer interactions overall per bin. For our prototype, we identified 50 bins and a filter threshold of at least five interactions per bin as parameterization through initial experiments with recorded interaction data.

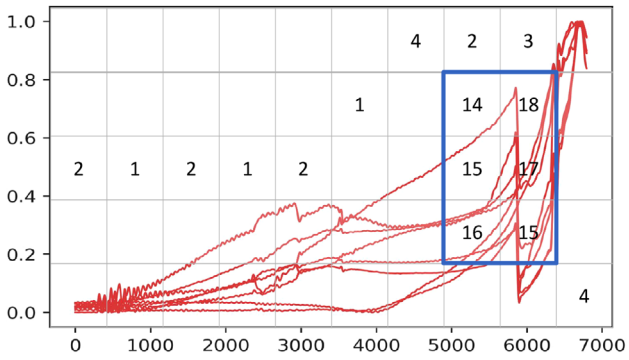

Figure 6: *An illustrative example of how regions of interest are calculated. We bin regions of the visible area inside a chart and count the contained interactions. Then, we calculate a rectangular overlay (blue area) around bins that contain more than a threshold of interactions.*

5.4. Guidance output: Guidance visualization

Using the process described previously, we can create and visualize guided EDA sessions. Since we currently assume fully guided EDA sessions (prescribing guidance system), where the domain novices have no control over the process, we create entire EDA notebooks. For each state in each action sequence within the generated EDA session, the corresponding charts are rendered in a row resulting in multiple rows and columns of charts. We highlight the region of interest with a blue box, where the experts have interacted most with the chart. Figure 7 shows an example of an EDA sequence for our exemplary deep drawing use case, where the expert adds three charts one after another resulting in a sequence of three states. Complete EDA notebooks consist of multiple such sequences.

Since charts usually contain titles as a further indication of what is being displayed, we have developed a procedure to generate titles that describe the pattern displayed in a generated action sequence. To this end, we have determined that presented insights can be divided into categories. For our use cases, these are: overview of label classes, variants of normal curves, variants of warning curves, variants of error curves. To decide which of the categories will be displayed as title, we train a simple one layer artificial neural network (ANN) on the respective pattern using a vector that classifies the number of charts and the contained set of labels of the last five states of a sequence. We only take the last five states because they are more likely to be relevant for the insight.

6. Evaluation

To assess the feasibility of our guided EDA approach, we conducted a user study to test whether domain novices are able to reproduce the insights of domain experts using the generated EDA sessions for

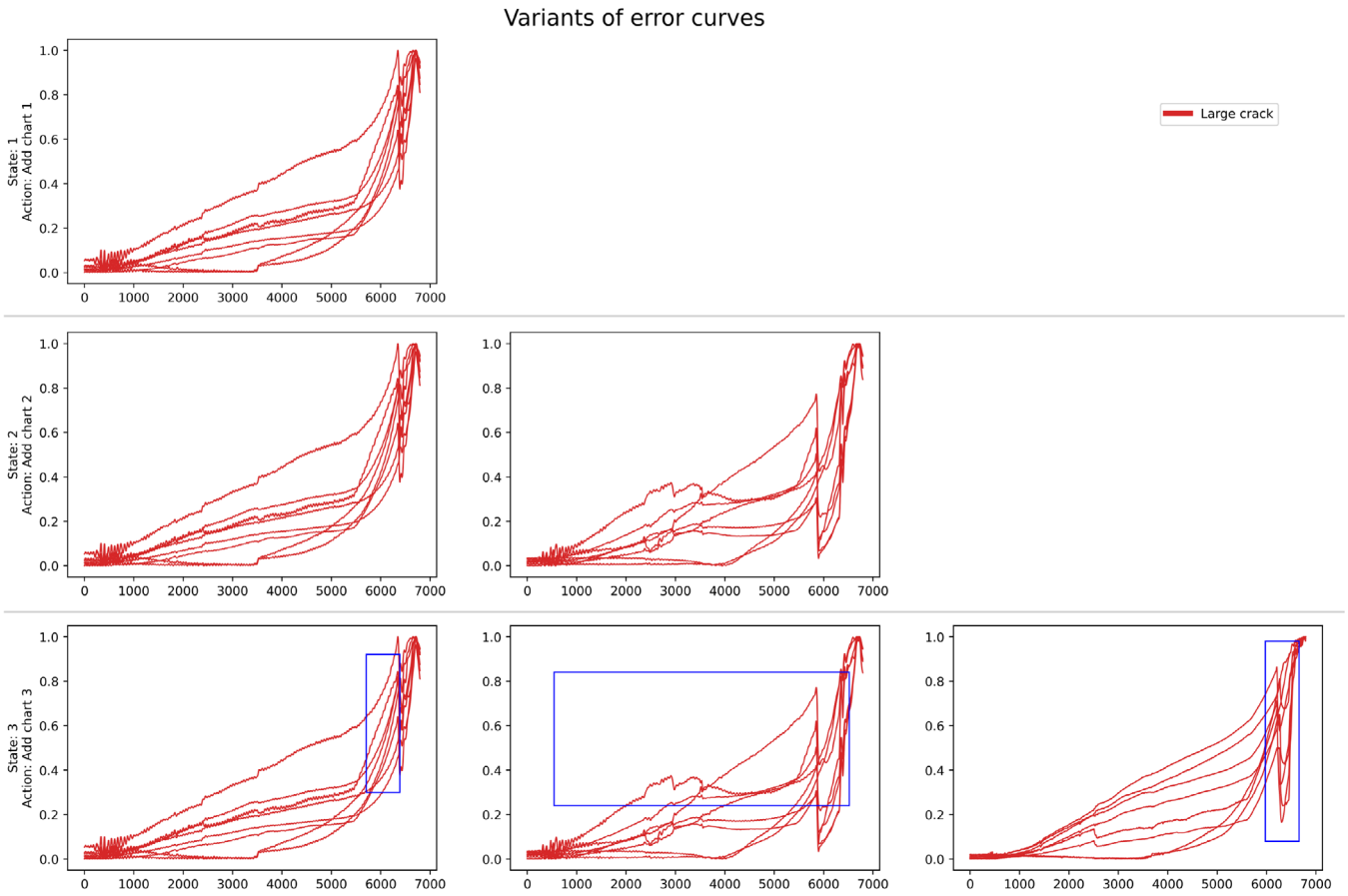

Figure 7: *An example of visualization for single action sequences generated by our system with regions of interest highlighted as blue rectangles. The complete EDA notebooks consist of multiple such sequences.*

our two representative exemplary use cases. In addition, we want to test whether the assumption that presentation as a sequence of views offers more context and regions of interest, as well as our generated titles, are helpful. Our goal was to show that our complementary approach is useful when recording classically conducted EDA sessions in order to use the data to make subsequent EDA sessions more efficient. A printable version of the original study website, as well as all generated data, scripts, and materials of our study are publicly available at https://github.com/tmdt-buw/gideon-core.

6.1. Participants

To test whether data scientists can reproduce the insights from generated EDA notebooks, we recruited 23 participants 21 males and 2 females aged 25–35 with data science experience and education in the STEM area, primarily computer science and engineering with a Master's or PhD degree, through a call for participation by mail at our university and industry partners. We did a pre-screening for participants' previous experience with data science tools and industrial use cases in brief interviews. We selected participants who were familiar with standard data science tools and already had practical experience in data science use cases in an industrial environment. All participants stated that they had experience with common data science tools like Jupyter and Python, data science libraries like numpy, pandas, or scikit-learn, as well as visualization solutions like Matplotlib or EDA tools like Tableau. We also wanted to make sure that our participants already had data science experience in the industrial sector, so that they would be skilled in applying data science methods and interpreting sensor data, and would only have to learn the specifics of the new use cases from the notebooks. The participants indicated that they already had practical data science experience in the industrial sector in other use cases. Industries included manufacturing (18), chemical (2), and mobility (3) sectors. Sensor types included a wide variety, from sensors that map physical values to image data.

6.2. Task and design

Our study aims to test whether it is possible to replicate insights using the insights generated by our approach. To test this, we designed our study as follows. First, we interviewed domain experts for key insights on both use cases (cf. Section 3) which served as baseline insights that experts would provide during an EDA session. From interviews, we compiled the following list of relevant reference

insights for the deep drawing (**DD**) and metal arc welding (**MAW**) use case:

[DD1] There are three classes of strokes in the dataset that are relevant for the use case: clean, small crack, and large crack which can be distinguished by the shape of sensor curves.
[DD2] There are different variants of clean strokes, each showing smooth sensor curves.
[DD3] There are different variants of strokes with small cracks (non-critical errors), each showing variants of small fluctuations in the last quarter of the sensor curves.
[DD4] There are different variants of strokes with large cracks (critical errors), each showing a sudden drop in values in the last quarter of the sensor curves.
[DD5] There are invalid segments containing configuration strokes in the dataset.
[DD6] Two sensors (Sensors 4 and 5) never show cracks (as they are placed in a bad position at the edge of the tool).

[MAW1] The welding process runs normally cyclically in uniform curves.
[MAW2] There are anomalies and configurations that break the cyclic pattern.
[MAW3] There are short short-circuits that are not problematic, and longer short-circuits (non-critical errors).
[MAW4] Several long short-circuits can potentially result in poor quality of the welding seam.

Second, we collected input data for our guidance system. We recorded six EDA sessions for the deep drawing use case and four EDA sessions for the metal arc welding use case. In each session, the above insights were shown in a self-selected order and in a self-selected manner. In this way, we could be sure that our baseline insights would be reflected within the collected interaction data, and thus would also potentially be shown in the final notebooks if our generation algorithm performed sufficiently well. For all sessions, we recorded the interactions and tool states as described in Sections 2.1 and 2.2 to receive expert exploration data. Based on this data, we generated one EDA notebook per use case as described in Section 5.3. In order to make the EDA notebooks accessible to the participants, we built a study website on which we gave a brief introduction to each use case without specific information about the data and displayed both EDA notebooks with free text fields next to each sequence shown in the notebook.

In the final study, the participants then had the task of inspecting the notebooks and describing the displayed insight in the free text field. We chose free text fields in order not to bias participants by, for example predefined options, but to ensure the generation of insights purely on basis of inspecting the EDA notebooks. While predefined answers would make the evaluation easier, EDA is not a multiple-choice task and the presentation of answer options alone could influence participants or create an insight they would not have had without the predefined answer options. In the last section of the study, we asked how useful they found the context from showing a sequence of actions **[C1]**, the blue marked regions of interest **[C2]**, and the titles **[C3]**. Participants were also free to give us further qualitative feedback. Finally, we evaluated the free texts with a domain expert from each use case and compared whether the description corresponded to the original insights.

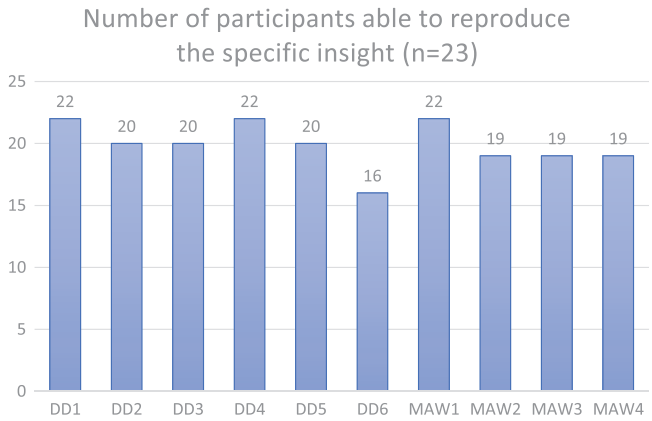

Figure 8: *Overview of how often each insight could be reproduced for our two exemplary use cases in our user study.*

6.3. Procedure

We conducted the study using remote conferencing software as well as in personal meetings in group sizes of one to four participants. The study website with EDA notebooks and questionnaire was hosted remotely on a server. The website was accessed from the participants' own computers. The participants received a short introduction, which can also be seen on the first slide of the questionnaire. We explained the subsequent task of interpreting the insights of the notebooks shown and describing them in the free text boxes. In addition, the structure of the notebooks was explained briefly, that is the visualization layout of sequences and the display of blue boxes that contain potentially interesting areas. To minimize the influence of the type of visualization as a static notebook, the participants could ask questions if they could not clearly distinguish a visual representation (e.g. because of the colour scheme) but no questions about the data, the use case, or the insight itself were allowed. We did not specify a time constraint. The whole study lasted between 20 and 35 min for each participant.

6.4. Results

Figure 8 gives an overview of how often each insight could be reproduced by the participants based on the information given by the EDA notebooks. On average, our participants were able to successfully reproduce approx. 8.65 out of 10 insights. Whereby DD1, DD4, and MAW1 were the most reproducible with 22 participants each, while DD6 was the least reproducible with 16 participants.

Overall, the participants found our approach useful and appreciated the visualization (P6) and consistent colour scheme (P10). However, some participants (P5 and P22) would prefer "an interactive over a static visualization of the notebooks". Some minor points concerning the visualization were mentioned, for example the lack of axis labels (P25), which cannot always be realized due to the automatic generation from data, as axis labels are optional in the dataset.

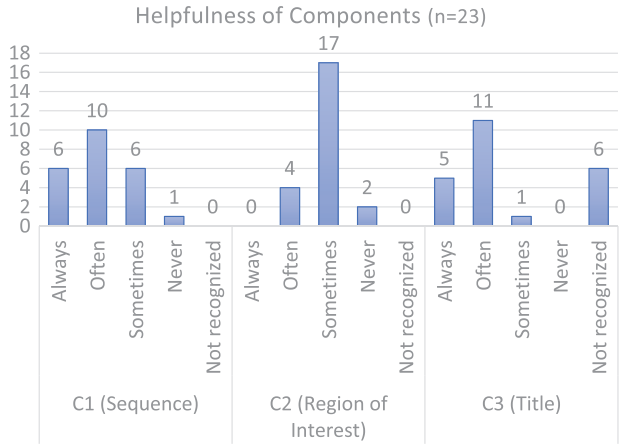

Figure 9: *Overview of perceived helpfulness of the components of our generated EDA notebooks.*

Figure 9 gives an overview of the perceived helpfulness of the visualization as sequences, the regions of interest, and the generated titles. For the most part, the **presentation as sequences** was rated as 'always' (6), 'often' (10), or 'sometimes' (6) useful. One participant mentioned that a direct encoding of the time component into the EDA notebooks would be helpful. *'A simple "add chart" was not always helpful. A time history of how quickly [individual visualizations] were created might help to see if they were viewed only for a short time or were inspected individually for a longer period of time.'* – P5. **Regions of interest** were mostly rated as 'sometimes' (17) useful. The qualitative feedback suggests that this is due to the boxes not always being accurately placed (P10 and P11) and sometimes not being comprehensible – especially when they are created from interaction noise in white space and do not contain data. *'A description of how and when the blue boxes are created would be useful. They felt random as they were sometimes shown and sometimes not.'* – P7. The **chart titles** were rated as 'always' (5) or 'often' (11) useful or were not recognized at all (6). Thus, they are most affected by the personal preferences of the participants. They were also sometimes misleading. *'Titles would have been even more helpful if they hadn't been sometimes misleading.'* – P13. When asked, the participant explained that, for example in DD6 the title 'variants of error curve' had distracted from the actually important factor of the applied filters, and in the MAW use cases, on the basis of the large number of labels presented, the corresponding title was not always helpful to identify the presented insight.

7. Discussion and Future Work

Our study found that our generated EDA sessions, which were created using recorded interactions from expert EDA sessions, can be valuable in generating insights about a dataset. Participants in our study were able to reproduce on average 86.5% of insights based on the information presented by our generated notebooks, with a range of 60% to 100%. However, the ability to convert information into insights is heavily reliant on the individual's ability to interpret the presented data, and domain knowledge remains a crucial factor in exploratory analysis. While our system can provide some of the information from experts, it cannot fully replace communication between domain experts and data scientists. Nevertheless, our study demonstrates the potential benefits of obtaining information directly from expert interactions to create follow-up EDA sessions, eliminating the need for relying on abstract measures of interestingness for previously explored datasets. Although our prototype employed a statistics-based approach, different methods and algorithms can be used for each component of the system design. We selected this approach as it is easier to understand and identify potential sources of error. Based on the qualitative feedback we received in our study, it seems useful to present guided EDA sessions with more context in the form of action sequences rather than a single view. We also conclude that the regions of interest and chart titles were generally helpful, although occasionally inaccurate. Their accuracy could be improved by fine-tuning the parameterization, expanding the range of title types identified, or exploring alternative approaches. In addition, participants' comments suggest that their usefulness varies depending on the user's experience level and personal preferences. Consequently, a potential solution could be to display them on demand, allowing users to decide when they need additional information about the insight being displayed. Overall, our approach serves as a baseline for future research, and more advanced algorithms could potentially generate better EDA sessions. There are limitations to the presented work due to the scope of this paper which focuses on the feasibility of such an approach. Our current guidance generation algorithm is susceptible to repetitive interactions that do not contribute to insight, and the need for experts to identify roughly the same insights in the data. We controlled for this in our study to some extent by requiring experts to first agree on the insights and then display them in the data at their discretion. This is also evidenced in our study by the fact that the identified regions of interest are sometimes generated in the white space of the chart and, thus, do not contain interesting data patterns. The corresponding comments of the participants suggest that this was sometimes confusing and contributed to them being rated as mostly 'sometimes' useful. Advanced filters and control mechanisms need to be developed to ensure data quality in the long run. We demonstrated the feasibility of our approach for two exemplary use cases. However, the difficulty of understanding the insights based on the example EDA notebooks varies from use case to use case. Therefore, there may be a limit where insights from very complex or complicated use cases are no longer comprehensible based on the generated notebooks in their current form and need to be further enriched or extended.

Overall, more research is needed on how to implement such a system and how to visualize a guided EDA session, as well as extensive user studies including more use cases to better understand the benefits and limitations of such an approach. An essential prerequisite for this is the collection of a larger corpus of use cases, insights and interaction data. Furthermore, approaches to *guidance input* could include new modalities such as eye tracking or gesture analysis. *Guidance generation* also offers great potential for further research. In related areas, such as recommender systems in e-commerce applications that predict customer purchase behaviour, there are approaches to represent interaction sequences in latent vector space [AMMM22]. Approaches to describe visualizations more formally are offered by grammars such as the vega-lite grammar [SMWH17]. A combination of these approaches could simplify the comparison of similar conditions and replace the

concrete formulation of a similarity measure. It could also facilitate the use of more advanced models, such as seq2seq models, which are currently mainly used in text generation. In particular, Dibia and Demiralp [DD19] have already used such a model to generate automated visualizations of data. In the case of *guidance output*, future research should first focus on the feedback of EDA sessions into interactive visualizations within the previously used EDA application. Subsequently, forms of presentation for guidance and the visualization of degrees of guidance should be investigated. In conclusion, our study provides a starting point for future research in this area, with the potential for significant benefits in the field of exploratory data analysis.

8. Conclusion

In this paper, we addressed the question of whether it is possible to derive some of the information domain novices need to generate insights about a domain directly from domain expert interaction data during EDA. To do this, we have set up a system design that generates EDA sessions from the states of an exploration tool and the interactions that domain experts perform while using that tool. The data served as input for an unsupervised algorithm, which splits the data into individual sequences, abstracts them using a custom distance function, clusters them, and applies them again to the dataset to obtain an EDA session. We then implemented this system design as a prototype and evaluated it in a user study using two real world exemplary industrial use cases. The data from our study suggest that it is possible to gain insights from notebooks generated from the interaction data of domain experts and to reproduce the experts' insights by following the notebooks.

Since our approach is based on collected interaction data from domain experts, the transferability of this data between use cases is relevant for our future research. This would reduce the effort involved in bringing our approach into practice and significantly increase its benefits. There are also several other ways to improve the automatically generated EDA notebooks. For example, experts could be given the opportunity to manually annotate the notebook and add descriptions. The effort would still be considerably reduced compared to the manual creation of an EDA notebook and even more information would be provided.

Acknowledgements

The authors have nothing to report.

Open access funding enabled and organized by Projekt DEAL.

References

[AABZ20] Adi E., Anwar A., Baig Z., Zeadally S.: Machine learning and data analytics for the IoT. *Neural Computing and Applications 32*, 20 (2020), 16205–16233.

[AMMM22] Alves Gomes M., Meyes R., Meisen P., Meisen T.: Will This online shopping session succeed? Predicting customer's purchase intention using embeddings. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management* (New York, NY, USA, 2022), Al Hasan M., Xiong L., (Eds.), ACM, pp. 2873–2882. doi: https://doi.org/10.1145/3511808.3557127.

[BBB20] Barczewski A., Bezerianos A., Boukhelifa N.: How domain experts structure their exploratory data analysis: Towards a machine-learned storyline. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2020), Bernhaupt R., Mueller F. F., Verweij D., Andres J., McGrenere J., Cockburn A., Avellino I., Goguey A., Bjørn P., Zhao S., Samson B. P., Kocielnik R., (Eds.), ACM, pp. 1–8. doi: https://doi.org/10.1145/3334480.3382845.

[BBT*19] Boukhelifa N., Bezerianos A., Trelea I. C., Perrot N. M., Lutton E.: An exploratory study on visual exploration of model simulations by multiple types of experts. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2019), Brewster S., Fitzpatrick G., Cox A., Kostakos V., (Eds.), ACM, pp. 1–14. doi: https://doi.org/10.1145/3290605.3300874.

[BC94] Berndt D. J., Clifford J.: Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining* (1994), AAAIWS'94, AAAI Press, pp. 359–370.

[BH19] Battle L., Heer J.: Characterizing exploratory visual analysis: A literature review and evaluation of analytic provenance in tableau. *Computer Graphics Forum 38*, 3 (2019), 145–159.

[BMS20] Bar El O., Milo T., Somech A.: automatically generating data exploration sessions using deep reinforcement learning. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 2020), Maier D., Pottinger R., Doan A., Tan W.-C., Alawini A., Ngo H. Q., (Eds.), ACM, pp. 1527–1537. doi: https://doi.org/10.1145/3318464.3389779.

[Boe16] Boehmke B. C.: *Data Wrangling with R*. Use R! Springer, Cham, 2016.

[CAA*20] Ceneda D., Andrienko N., Andrienko G., Gschwandtner T., Miksch S., Piccolotto N., Schreck T., Streit M., Suschnigg J., Tominski C.: Guide me in analysis: A framework for guidance designers. *Computer Graphics Forum 39*, 6 (2020), 269–288.

[CBYE19] Cui Z., Badam S. K., Yalçin M. A., Elmqvist N.: DataSite: Proactive visual data exploration with computation of insight-based recommendations. *Information Visualization 18*, 2 (2019), 251–267.

[CCSK18] Camisetty A., Chandurkar C., Sun M., Koop D.: Enhancing web-based analytics applications through provenance. *IEEE Transactions on Visualization and Computer Graphics* (2018). doi: https://doi.org/10.1109/TVCG.2018.2865039.

[CGL20] Cutler Z., Gadhave K., Lex A.: Trrack: A library for provenance-tracking in web-based visualizations. In *2020 IEEE*

Visualization Conference (VIS) (2020), IEEE, pp. 116–120. doi: https://doi.org/10.1109/VIS47514.2020.00030.

[CGM*17] Ceneda D., Gschwandtner T., May T., Miksch S., Schulz H.-J., Streit M., Tominski C.: Characterizing guidance in visual analytics. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (2017), 111–120.

[DD19] Dibia V., Demiralp C.: Data2Vis: Automatic generation of data visualizations using sequence-to-sequence recurrent neural networks. *IEEE Computer Graphics and Applications 39*, 5 (2019), 33–46.

[DIN03] DIN 8584-3: *Manufacturing Processes Forming under Combination of Tensile and Compressive Conditions - Part 3: Deep Drawing; Classification, Subdivision, Eerms and Definitions*. Beuth Verlag, Berlin, 2003.

[EKSX96] Ester M., Kriegel H.-P., Sander J., Xu X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (1996), KDD'96, AAAI Press, pp. 226–231.

[KHW*19] Kieslich P. J., Henninger F., Wulff D. U., Haslbeck J. M. B., Schulte-Mecklenbeck M.: Mouse-Tracking: A practical guide to implementation and analysis 1. In *A Handbook of Process Tracing Methods*. Routledge, 2019, pp. 111–130.

[Klo11] Klocke F.: *Manufacturing Processes 1*. Springer, Berlin, Heidelberg, 2011. doi: https://doi.org/10.1007/978-3-642-11979-8.

[KMSZ06] Keim D. A., Mansmann F., Schneidewind J., Ziegler H.: challenges in visual data analysis. In *Tenth International Conference on Information Visualisation (IV'06)* (2006), IEEE, pp. 9–16. doi: https://doi.org/10.1109/IV.2006.31.

[KSM13] Kah P., Suoranta R., Martikainen J.: Advanced gas metal arc welding processes. *The International Journal of Advanced Manufacturing Technology 67*, 1-4 (2013), 655–674.

[KWHH17] Kim Y., Wongsuphasawat K., Hullman J., Heer J.: GraphScape: A model for automated reasoning about visualization similarity and sequencing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2017), Mark G., Fussell S., Lampe C., schraefel m., Hourcade J. P., Appert C., Wigdor D., (Eds.), ACM, pp. 2628–2638. doi: https://doi.org/10.1145/3025453.3025866.

[LDC13] Li Vigni M., Durante C., Cocchi M.: Exploratory data analysis. In *Chemometrics in Food Chemistry*, vol. *28* of *Data Handling in Science and Technology*. Elsevier, 2013, pp. 55–126. doi: https://doi.org/10.1016/B978-0-444-59528-7.00003-X.

[LKL*18] Lin Q., Ke W., Lou J.-G., Zhang H., Sui K., Xu Y., Zhou Z., Qiao B., Zhang D.: BigIN4: Instant, interactive insight identification for multi-dimensional big data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (New York, NY, USA, 2018), Guo Y., Farooq F. (Eds.), ACM, pp. 547–555. doi: https://doi.org/10.1145/3219819.3219867.

[LM21] Langer T., Meisen T.: System design to utilize domain expertise for visual exploratory data analysis. *Information 12*, 4 (2021), 140.

[LMM22] Langer T., Meyes R., Meisen T.: Gideon Replay: A library to replay interactions in web-applications. *SoftwareX 17* (2022), 100964.

[LS10] Liu Z., Stasko J. T.: Mental models, visual reasoning and interaction in information visualization: A top-down perspective. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 999–1008.

[LWM22] Langer T., Welbers V., Meisen T.: Gideon-TS: Efficient exploration and labeling of multivariate industrial sensor data. In *Proceedings of the 24th International Conference on Enterprise Information Systems* (2022), SCITEPRESS - Science and Technology Publications, pp. 321–331. doi: https://doi.org/10.5220/0011037200003179.

[MDSM19] Meyes R., Donauer J., Schmeing A., Meisen T.: A recurrent neural network architecture for failure prediction in deep drawing sensory time series data. *Procedia Manufacturing 34* (2019), 789–797.

[MHHH15] Moritz D., Halperin D., Howe B., Heer J.: Perfopticon: Visual query analysis for distributed databases. *Computer Graphics Forum 34*, 3 (2015), 71–80.

[MR13] Mooney C. H., Roddick J. F.: Sequential pattern mining – Approaches and algorithms. *ACM Computing Surveys 45*, 2 (2013), 1–39.

[MS18] Milo T., Somech A.: Next-step suggestions for modern interactive data analysis platforms. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (New York, NY, USA, 2018), Guo Y., Farooq F., (Eds.), ACM, pp. 576–585. doi: https://doi.org/10.1145/3219819.3219848.

[PMH15] Paaßen Benjamin, Mokbel Bassam, Hammer Barbara: A toolbox for adaptive sequence dissimilarity measures for intelligent tutoring systems. In *Proceedings of the 8th International Conference on Educational Data Mining* (2015), O. Christina Santos, J. Gonzalez Boticario, C. Romero, M. Pechenizkiy, A. Merceron, P. Mitros, J. Maria Luna, C. Mihaescu, P. Moreno, A. Hershkovitz, S. Ventura, M. Desmarais (Eds.), Proceedings of the 8th International Conference on Educational Data Mining, International Educational Datamining Society, p. 632.

[RESC16] Ragan E. D., Endert A., Sanyal J., Chen J.: Characterizing provenance in visualization and data analysis: An organizational framework of provenance types and purposes. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 31–40.

[SGP*18] Stitz H., Gratzl S., Piringer H., Zichner T., Streit M.: KnowledgePearls: Provenance-based visualization retrieval. *IEEE Transactions on Visualization and Computer Graphics* (2018). doi: https://doi.org/10.1109/TVCG.2018.2865024.

[Shn96] Shneiderman B.: The eyes have it: a task by data type taxonomy for information visualizations. In *Proceedings 1996 IEEE Symposium on Visual Languages* (1996), IEEE Comput. Soc. Press, pp. 336–343. doi: https://doi.org/10.1109/VL.1996.545307.

[SMWH17] Satyanarayan A., Moritz D., Wongsuphasawat K., Heer J.: Vega-Lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (2017), 341–350.

[SPC*17] Shneiderman B., Plaisant C., Cohen M. S., Jacobs S. M., Elmqvist N.: *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, sixth edition ed. Pearson, Boston [i pozostałe], opyright 2017.

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

[THY*17] Tang B., Han S., Yiu M. L., Ding R., Zhang D.: Extracting top-K insights from multi-dimensional data. In *Proceedings of the 2017 ACM International Conference on Management of Data* (New York, NY, USA, 2017), Chirkova R., Yang J., Suciu D., (Eds.), ACM, pp. 1509–1524. doi: https://doi.org/10.1145/3035918.3035922.

[WJL*15] Walker J. S., Jones M. W., Laramee R. S., Bidder O. R., Williams H. J., Scott R., Shepard E. L. C., Wilson R. P.: TimeClassifier: A visual analytic system for the classification of multi-dimensional time series data. *The Visual Computer 31*, 6-8 (2015), 1067–1078.

[WMA*16] Wongsuphasawat K., Moritz D., Anand A., Mackinlay J., Howe B., Heer J.: Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 649–658.

[XOW*20] Xu K., Ottley A., Walchshofer C., Streit M., Chang R., Wenskovitch J.: Survey on the analysis of user interactions and visualization provenance. *Computer Graphics Forum 39*, 3 (2020), 757–783.

[ZXC*21] Zhao J., Xu S., Chandrasegaran S., Bryan C. J., Du F., Mishra A., Qian X., Li Y., Ma K.-L.: ChartStory: Automated partitioning, layout, and captioning of charts into comic-style narratives. *IEEE Transactions on Visualization and Computer Graphics PP* (2021). doi: https://doi.org/10.1109/TVCG.2021.3114211.

Supporting Information

Additional supporting information may be found online in the Supporting Information section at the end of the article.

Supporting Information