Computers & Graphics (2022)

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

A Data-Driven Platform for the Coordination of Independent Visual Analytics Tools

ARTICLE INFO

Article history:
Received June 28, 2022

Keywords: Human-centered computing, Visualization systems and tools, Information systems, Data exchange

ABSTRACT

Visual analysis of unknown data requires the combined use of various functions that are often part of standalone visual analytics (VA) tools. Performing cross-tool visual analysis with standalone VA tools, however, is a challenging and cumbersome endeavor. Some dedicated frameworks address this issue, yet in order to utilize any of them, a visual analytics tool needs to support their required API or architecture. Contrary to most existing frameworks, we present an approach that does not rely on a single predefined interchange mechanism for the entire ensemble of VA tools. Instead, we propose using any available channel for data exchange between two consecutive VA tools. This allows mixing and matching of different data exchange strategies over the course of a cross-tool analysis. In this paper, we identify the challenges associated with establishing such tool chaining platform for data-driven coordination. We further describe the structure and capabilities of data exchange and explain various functionalities of our platform in detail. Based on a demonstrating example, we discuss the limitations of our approach and elaborate new insight for the coordination of the visual output of multiple VA tools.

1. Introduction

Software integration is a longstanding challenge in computer science [1, 2]. For Visual Analytics (VA), this challenge has been addressed with a variety of approaches ranging from analysis libraries [3] to extensible frameworks [4]. Yet, all these approaches have not eliminated the need to switch between different, highly specialized VA tools in an interactive visual data analysis. The reason is that no framework can be top of the class at all possible VA tasks. Hence, even in the age of powerful VA software like KNIME [5] and Tableau [6], we still rely on separately running OpenRefine [7] for any serious data cleaning task or Gephi [8] for advanced network analysis and visualization.

Switching between VA tools breaks the analytic flow by having to worry about technical constraints of data export/import between tools. Something simple like going back to a previous tool to readjust a parameter and observe its effects in the subsequent tool becomes a rather painful experience – in particular if one needs to do this back-and-forth between tools multiple times to get the parameter right. In some cases, VA tools do not support the export/import of data or visualizations at all, let alone in a standardized format. So, while switching between independent VA tools is necessary for complex application scenarios, it comes at a considerable cost.

Our approach for VA tool coordination acknowledges the need for multiple specialized VA tools to perform complex analytical tasks. Yet, we aim to reduce the cost of using these independent VA tools in conjunction by facilitating the switches between them. To that end, we build on our conceptual models for VA tool coordination [9] and data exchange among VA tools [10] to realize such a coordination and to demonstrate that it is feasible and useful. For the realization of this concept, we follow the idea of data-flow oriented visualization software. Except that instead of composing individual modules of the same system into one data flow, we do so for individual VA tools and exchange data along that flow by utilizing the available means of pairwise communication. To this end, we present a software platform called *AnyProc* that enables technical experts to customize and connect independent VA tools and data sources so that the domain experts can then execute their workflow with little to no interference. Our solution realizes a reasonable subset of data-driven exchange mechanisms that are shown in a demonstrating scenario. This can be considered as a first step towards our vision of fully enabling lightweight coordination of independent VA tools.

This paper is an extension of our work presented at the EuroVA Workshop 2021 [11], which was held in conjunction with the EuroVis Conference in Zürich. We extended the original workshop paper with the following contributions:

- We added *new sections* on our preliminary work, design challenges, and decisions to put our approach into context and show the fundamental considerations for our platform.
- We provided *more detail* on which features our platform offers and how they have been technically implemented.
- We *expanded the discussion* of the scope of our solution and included advances in user interface (UI) coordination.

This paper is structured as follows: The research background containing related work and our prior work are described in Sec. 2. Our design goals, challenges, and choices are outlined in Sec. 3. The platform and its features are detailed in Sec. 4. The implementation and data exchange specifications are described in Sec. 5. In Sec. 6, the application scenario and user feedback are presented. Finally, limitations, recent advanced, and future work are outlined in Sec. 7 and Sec. 8.

2. Background

Our work is equally motivated and driven by related work on integrating different analysis tools and our prior work on coordinating independent VA tools.

2.1. Related Work

Various approaches exist to support working with multiple individual tools. These approaches can be broken down along the three concerns they mainly address: *data exchange between tools*, *UI integration between tools*, and *analytic process support across tools*.

Data Exchange between Tools. Getting data from one tool to another in a seamless way can be done either via a central hub such as a relational database [12, 13], or via decentralized web services [14, 15]. Dedicated software design patterns like the *proxy tuple* can aid different tools to access a unified data model [16]. Yet, for integrating a tool in a software ecosystem based on any of these approaches, it is commonly required to use the same approach.

UI Integration between Tools. UI integration plays a major role when using multiple individual tools concurrently. Such integration can be rather lightweight by adding visual links or information scents to highlight the same data across tools [17, 18], or it can be more involved and actually blend different tools into a combined UI [19, 20]. Though, in particular the blending approaches have the downside that domain experts can no longer rely on their established mental map of the individual tool UIs.

Analytic Process Support across Tools. Analysis processes are supported in two ways: by guiding the domain expert along a given analytic workflow from tool to tool [21], or by establishing analytic provenance across tools [22, 23]. The challenge shared among these mechanisms is to capture internal states and transitions from within the individual tools to either register the domain expert's progress along a given workflow or to establish that workflow from interaction logs in the first place.

2.2. Own Prior Work

In most user workflows, VA tools are used in a chained manner by applying the specialized tools for each analysis step from cleaning, preprocessing, and visual-interactive analysis of the data to fine-tuning the visual results. In previous work [9], we presented the idea of plugging these VA tools pairwise together using custom scripts and copy & paste keystrokes. Thereby, we separated different aspects of cross-tool analyses in three layers for coordination.

1. The *Usage Flow* captures the intended order in which tools are used, so as to know between which tools coordination is necessary and only to realize it between them.
2. The *Data Flow* captures different characteristics of data exchange [10] between each pair of VA tools, so as to specify individually how the data exchange is performed.
3. The *Control Flow* captures the actions of the domain expert and the system, so as to react and signal necessary adjustments in the visualization.

To develop mechanisms for the coordination of multiple VA tools based on these three layers, we define a *tool chain* as a representation of the user's workflow, where each individual processing step is bound to the utilization of one or more corresponding VA tools. This allows us to model application scenarios from a domain expert's perspective as a chain of independent VA tools that can be executed to drive the analysis. To use the individual layers of pairwise exchange (namely usage flow, data flow, and control flow) together with our definition of a tool chain, we have developed a so-called coordination graph. Therein, the nodes represent the VA tools and directed edges capture the coordination between pairs of VA tools (Fig. 1).

In our prior work, we have so far discussed a largely theoretical model. This model can be used to represent arbitrary application scenarios with unknown relationships between data and tools. However, translating this model into a concrete platform for coordinating analytical tool chains requires further analysis of the specific requirements and design decisions to be made.

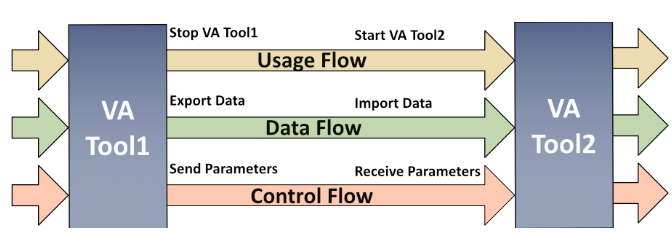

Fig. 1. Representation of the pairwise exchange between two independent VA tools within the coordination graph. Information for the consecutive use is transferred from VA Tool1 to VA Tool2 along the available layers for Usage Flow, Data Flow, and Control Flow.

3. Objectives, Design Principles and Challenges

Before we introduce the developed platform and show its application, we share some design details about that influenced our decisions for the implementation.

3.1. Objectives

The development of cross-tool VA use cases usually involves two parties with different roles. The first role is that of the *domain expert*, who has knowledge of the domain workflow and the associated sequence in which the tools must be applied. Domain experts are the ones to apply the workflow by performing actual data analysis using the given tools to draw conclusions based on their professional expertise. The second role is that of the *technical expert*, who possesses knowledge about data visualization, the necessary data exchange, and available communication channels between tools. The technical expert is responsible for coordinating the tools and configuring the data exchange based on a given domain workflow. Both roles work together to establish how the available tools can be combined and what options are feasible.

Our goal is to assist exactly at this point, supporting the technical expert in configuring a suitable tool chain as well as the domain expert in navigating it and executing the workflow accordingly. With our platform, we particularly want to ease the configuration process so that switching between tools at execution time can be automated as much as possible.

3.2. Design Principles

Since the coordination of independent VA tools for visual data analysis is fundamentally driven by the data and the insights found in them, we focus our design primarily on data exchange. This entails that we have to deal with the various challenges of passing data along tool chains. To do so, we rely on the following three design principles:

1. *Data exchange is* opportunistic and uses any available data channel between two VA tools. This allows us to mix tools from different software ecosystems – e.g. native applications using the file system for data exchange and web apps using a server-based data exchange.
2. *Data exchange is* minimalistic and exchanges data only between VA tools used subsequently or concurrently during the analysis. So instead of broadcasting data updates to all tools, our approach simply propagates data changes along the analytic workflow as it is carried out.
3. *VA tools are seen as atomic* and the data exchange focuses on the switch points when the domain expert changes from one VA tool to another. We do not aim to inspect the internal processes of the VA tools [24] by observing their initial states, but merely to capture their output and feed it into the next tool, as the domain expert progresses along the analytic workflow. Nevertheless, this may still require minimal code changes by the technical expert, e.g., in tools that have no standardized data input/output functionality.

3.3. Implementation challenges

Considering our objectives and design principles, we see the following challenges to transform our theoretical concept into a viable platform that can coordinate individual data-flow oriented operations.

Transport of Data between VA Tools. For the purpose of coupling different VA tools in an independent way, we have to remain cautious about case-specific limitations. Hence, it is reasonable to assume that there are missing details about the technical structure of each VA tool and their communication capabilities. To enable data exchange in such an environment, one needs to provide *generic options for the transport of data*. Although, this is unlikely to be a feasible goal for every system, it is nevertheless desirable to provide multiple exchange mechanisms that are easy to deploy.

Variety of Data Characteristics. Another challenge related to the variety of possible application workflows is the heterogeneity of the data. It can be assumed that the analysis to be carried out will span a variety of data sources with different formats and unknown sizes. Hence, adjustments regarding performance and comparability might be necessary for the efficient use of the tool chain. Such potential data transformations or conversions require a flexible interface with different options for the modification to be carried out. Depending on the data exchange capabilities of the VA tools involved, this might be done automatically by the system or manually by the technical expert.

Assurance of Operations. So far, we have primarily addressed concerns of the technical expert. However, we must also note a challenge in terms of collaboration with the domain expert. For the domain expert, it is important that the data is tracked and can therefore be traced back along the tool chain so that the information presented is reliable throughout the analysis. Hence, our platform must not only support data exchange, but also ensure *reliability of persisting information*. This requires mean of provenance and status feedback through representation of data states and operations.

4. A Platform for Coordinating Analytical Tool Chains

To address the described challenges and to support switching between VA tools, we present the *Analytical Process Constructor (AnyProc)*. AnyProc provides a graphical interface with two different modules to structure stand-alone tools in an analytic workflow and to carry out this workflow across tools. While the *editor module*, aims to reduce the cost for the technical expert to link various VA tools and data sources, the *executor module* aims to guide the domain expert through the analytic workflow.

Designed for the technical expert, the AnyProc editor module consists of three different containers (see Fig. 2). The top and bottom containers are used to import VA tools and data sources through a drag-and-drop mechanism. The imported sources can then be arranged in the central container to create custom tool chains based on the domain expert's workflow. By providing containers to hold imported data and VA tools at the top and bottom, respectively, the technical expert can re-use them in

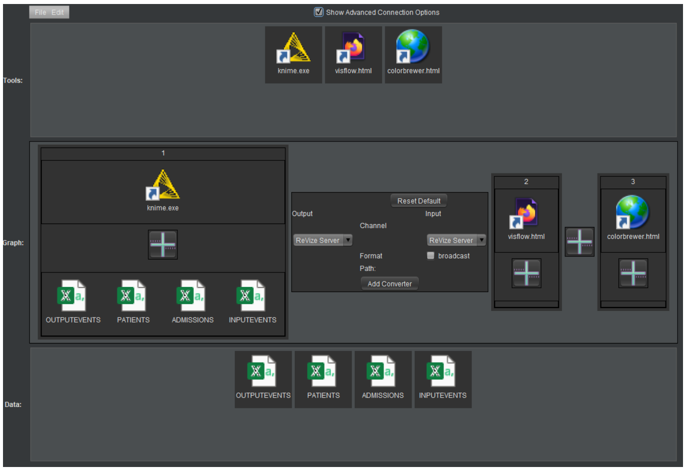

Fig. 2. Screenshot of the editor module with the VA tools KNIME, VisFlow and ColorBrewer (top) and the data sources patients, admissions, d_items and outputevents (bottom) imported.

different process steps of the tool chain. This saves time and provides an overview in situations where the domain expert's workflow requires multiple instances of a VA tool or the repeated use of specific data sources.

In the center of the editor module is the main container used for assembling the tool chain. Every time a source is dropped in from the outlying containers, the editor creates a new process step. This way, the technical expert can configure an order of task execution that matches the domain expert's workflow. As part of this, each process step can contain multiple VA tools and data sources, which may be linked by drawing connections between the placed VA tools or data source icons. This allows different cardinality characteristics [10] to be modeled, enabling both sequential and parallel executions with single or multiple data sources. During execution, all VA tools are opened according to their links with the associated data sources using the available communication channel.

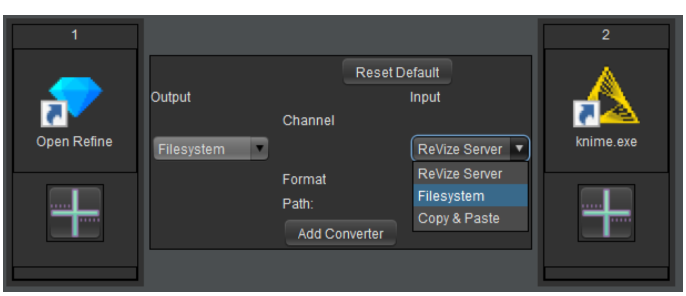

Fig. 3. Screenshot of the `ConnectionNode` within the editor module showing the specification for the connection between OpenRefine and KNIME. Data channels can be adjusted through drop-down menus and data converters can be added using the [Add Converter]-Button to resolve any mismatches.

This basic structure allows for modeling of diverse tool chains, that can be executed with the default settings of our implementation. However, as mentioned in Sec. 3.3, this is often not enough. We therefore added optional settings that allow the technical expert to *directly control the data transport between independent VA tools*. Between each process step there are so-called `ConnectionNodes`, which allow the technical expert to make independent adjustments, for example, by changing the communication channels from websockets to the file system or the clipboard for data exchange (see Dropdown Menu in Fig. 3). Moreover, the technical expert can add own converter configurations to be used as utility tools between the process steps to perform data transformations before the information is transmitted along the chosen communication channel (see [Add Converter]-Button in Fig. 3).

Since some VA tools may not be automatically compatible with the required data sources, it might be necessary for the technical expert to make further adjustments. In order to *support a variety of data characteristics* not only during data transport but also during import, we have included the option for connector programs on each created link between VA tools and data sources (see [+]-Button in Fig. 2). Technical experts can use this option to include custom utility tools that support the import of data into compatible formats for a specific VA tool.

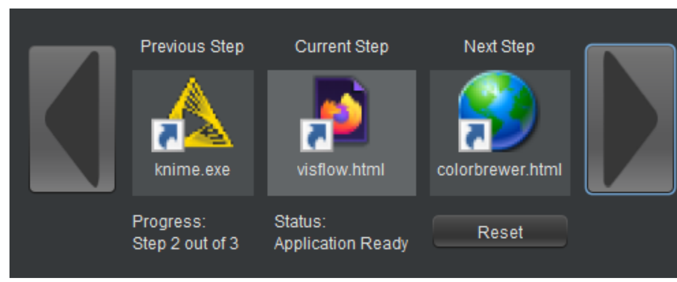

Fig. 4. Screenshot of the executor module showing the VA tools KNIME, VisFlow, and ColorBrewer (center). The currently opened tool (VisFlow) is slightly highlighted by a lighter color. The subsequent or previous steps of the tool chain can be opened via the control elements (arrows left and right).

In addition to AnyProc's support for the technical expert, it also offers dedicated features for the domain expert. The executor module consists mainly of a visual interface that provides guidance for navigating the tool chain. Using the left and right arrow buttons shown in Fig. 4, the domain expert can start the next or restart the previous tools with the linked data sources according to the settings in the editor module. The defined data flow will then be carried out accordingly – whether this means to simply open the same data file in another tool or to contact a server for a more elaborate data exchange. In this way, AnyProc is able to *generate assurance of operations*, as the domain expert can track the current progress along the tool chain and plan further ahead or revisit previous steps.

5. Options for Data Exchange

AnyProc offers several channels to facilitate data exchange between VA tools. In particular, we consider data exchange via websockets, the file system, and the clipboard.

The primary channel for data exchange in AnyProc are *websockets*, which allow tools to automatically send and receive data to and from each other via a TCP connection. To achieve this, all visualization tools communicate with a central server that distributes data to other tools. The main benefit of this

Feature	Supported by AnyProc?
Datasets	Yes (via all channels)
Filters	Yes (via Vega-Lite)
Data transformations	Yes (via Vega-Lite)
Models	No
Visual encodings	Yes (via Vega-Lite)
Selections/brushes	Yes (via Vega-Lite)

Table 1. Parts of the visualization that can be shared between tools via data exchange channels in AnyProc.

channel is the full automation it provides: The domain expert simply launches tools using the executor panel, and AnyProc automatically exchanges data between them, allowing the domain experts to continue their work without thinking about importing or exporting data. Moreover, websockets provide flexibility in the way in which tools can be used. For example, we can have multiple tools exchanging data at the same time.

More specifically, AnyProc uses the websocket integration provided by ReVize, an open source JavaScript library for extending web-based visualization tools [25]. With ReVize, visualization tools use websockets to import and export JSON objects that contain descriptions of the visualization pipeline, using the declarative visualization grammar Vega-Lite [26]. We chose a visualization grammar in general and Vega-Lite in particular as a common interface for the data exchange for multiple reasons. First, all static parameters and operators of the visualization pipeline are accessible to any tool, which means that the same channel can be used at any step in the tool chain. Second, the grammar serves as a standard data exchange format, meaning that any tool "speaking" that language can be used (in any order) in the tool chain. A third benefit is that Vega-Lite, while using a relatively simple grammar to construct visualizations, has been adapted to many use cases and is used in many tools.[1] A detailed overview of the capability of our websocket interface can be found in Table 1.

To make a VA tool "ready" for exchanging data over websockets, it needs to fulfill two conditions: First, it needs to be able to read and write TCP websockets, for instance using the `socket.io`[2] library, which is available for many programming languages. Second, the tool needs to be able to import and export Vega-Lite specifications. For browser-based VA tools, both can be facilitated by the ReVize library [25].

Whenever a tool does not support the websocket channel, AnyProc provides fallback options to nevertheless automate parts of the data exchange. The first fallback option is the *file system*, i.e., two tools exchange data by exporting and importing it to a file accessible to both. The general procedure for this channel is that the domain experts store the data in a file using their current tool, specify this file as input to the next tool using the AnyProc editor, and then open the next tool using the executor panel. This means that, in any case, domain experts have to export their data manually via the interface of their respective tool. For some tools, however, AnyProc can automate the data import, for example, by passing the file URI as a parameter to the next tool. For tools that do not support passing a file URI as a parameter, technical experts can provide connectors that extend the functionality of the tool to automate the import. If no connector can be provided, for example, whenever the source code is not available or implementing one, data import can still be performed manually.

Since not all tools may have access to local storage, using the file system may not always be a feasible option, either. Thus, another fallback option to websockets is to use the *clipboard* to exchange data between tools via copy-and-paste. To support this channel, AnyProc allows technical experts to copy and paste data between tools. For example, to include a visualization tool that can import the Vega-Lite grammar, but does not implement the ReVize websocket channel we outlined above, we can instead copy and paste the JSON description from one tool into the next. Some of the steps required for this can again be automated, for example, by automatically copying data from the previous tool to the clipboard, or by automatically opening the input field in the target tool using scripting. Nevertheless, the clipboard remains a largely manual data exchange channel.

An implementation of our AnyProc platform with the aforementioned user interface and data exchange options is made freely available under a permissive open source licence.[3]

6. Application to a Cross-Tool VA Scenario from Health IT

In our application scenario, we use AnyProc for the visual analysis of several tables from the critical care dataset MIMIC-III [27] with three independent VA tools. From this dataset, we use the ca. 46,000 patient records, their ca. 58,000 hospital admissions, as well as their registered fluid intake events (ca. 17.5 million entries) and fluid output events (ca. 3.6 million entries). Based on the introduced role system, the domain expert could be a physician or a hospital administrator, while a data scientist or health IT professional could be the technical expert configuring the tool chain. The scenario itself is not particularly complex, but requires already quite a bit of back-and-forth between different tools. In that sense, it is nevertheless able to show the fundamental challenges of a cross-tool analysis and how AnyProc can support the domain expert in dealing with them. In the following, we describe the specifications of usage flow, data flow, and control flow as it is done in AnyProc.

6.1. Specifying the Usage Flow

We plan to use the following three independent VA tools – in that order, but allowing for a back and forth between them if needed:

1. KNIME [5] for preprocessing the data using unsupervised machine learning methods like clustering and PCA;
2. VisFlow [28] for the exploratory, interactive visual analysis of the results obtained using KNIME;
3. ColorBrewer [29] for fine-tuning the color scale in the plots generated by VisFlow.

[1] `https://vega.github.io/vega-lite/ecosystem.html`
[2] `https://socket.io/`
[3] `https://github.com/nonnemann/anyproc_public`

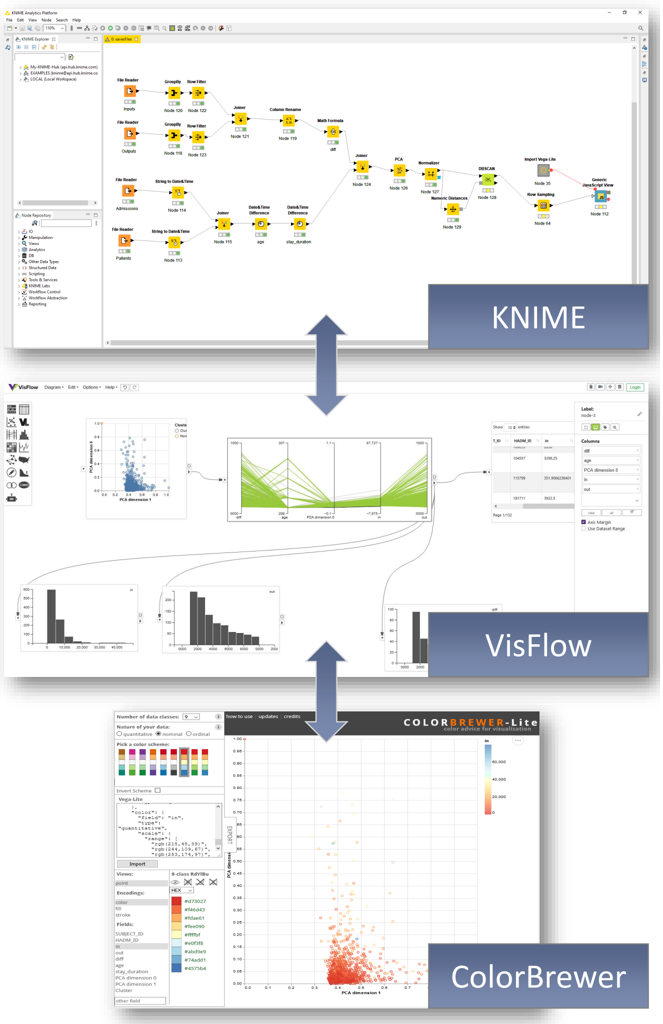

Fig. 5. Initial tool chain with KNIME, VisFlow, and ColorBrewer.

Each of these three VA tools excels at a different stage of the analysis process. KNIME is powerful in its computational analysis capabilities, but it does not lend itself to the visual-interactive exploration of the computed results. VisFlow has its strength in the flexible configuration and exploration of data visualizations, but lacks the advanced computational capabilities of KNIME. Finally, ColorBrewer is used for its wide range of color schemes to choose from. It also allows to specifically constrain the color choice to only color-blind friendly palettes – a feature not supported by VisFlow. Fig. 5 illustrates this usage flow among the three VA tools.

6.2. *Specifying the Data Flow*

In the second step, we define how these three tools pass data among each other. At least in part, this requires some code changes as, for example, ColorBrewer in its original form does not allow any data to be passed in or out at all. Hence, we need a solution that allows us to add this functionality to ColorBrewer, while at the same time demanding less coding effort than integrating ColorBrewer's functionality into VisFlow. Since both tools are open source and web-based, we chose ReVize websockets as a communication interface (cf. Sec. 5) between tools, such that they exchange a Vega-Lite specification of the latest visualization with the next tool in the tool chain through websockets. To achieve this, we specifically use variants of VisFlow and ColorBrewer as well as KNIME that are "ReVize-enabled", meaning they have been extended to support this channel.

By doing so, We can configure the connections as needed (as shown in Fig. 2), which results in the following data exchange characteristics [10]: Each data transfer includes the *full dataset* together with additional *metadata about its visual representation* using the *structured data format* Vega-Lite. Using the ReVize server as a *central infrastructure*, the data exchange is used for *inputs and outputs* from tools, as well as to distribute interactive *modifications* from *one to many* other VA tools connected to that server. This is done *bidirectionally* (backward and forward along the tool chain) in a *synchronous* fashion where changes are *pushed* to the server when switching from one VA tool to another. At all times, the connected VA tools have *full access* to the most recent Vega-Lite specification, which is kept *persistent* on the server.

6.3. *Specifying the Control Flow*

Switching to the AnyProc executor, KNIME is started as the first VA tool in the tool chain. Since we configured it to connect via ReVize to the next tool in the tool chain, KNIME is opened already with a minimal analytic pipeline in place that shows the four data sources, as well as with a final Vega-Lite node that will output the necessary specification in the end. We first add data transformations (e.g., date conversions), computations (e.g., determining the patients' ages), cleaning (e.g., removing missing entries), and aggregations (e.g., summing up fluid intakes and outputs per admission), before joining all data of interest into a single table. We then perform a PCA on this multi-dimensional data to map them into two dimensions. On top of this mapping, we perform a DBSCAN clustering to find groups of patients and then we finally plot the resulting data as shown in Fig. 6. This is as far as KNIME takes us and we switch to VisFlow for interactively exploring the plot.

To do so, we click the [Next] button in the AnyProc executor, which opens up VisFlow with the plot generated in KNIME. In the background, KNIME has continuously pushed any updates

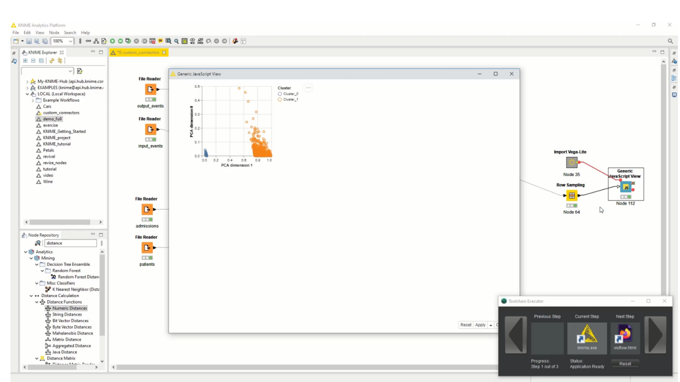

Fig. 6. Screenshot of the visual result from the first step in our application scenario. The AnyProc executor module (in the foreground on bottom right corner) is on top of the local KNIME installation (in the background) to help the domain expert with the navigation through the previously defined tool chain.

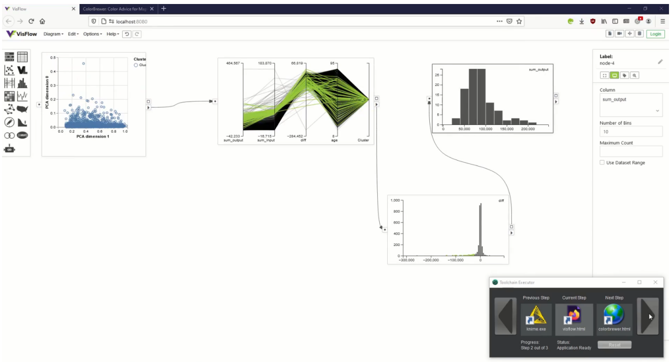

Fig. 7. Screenshot of the visual result from the VisFlow web tool (in the background) after filtering out incorrect information with KNIME. Switching between these two VA tools is possible through the AnyProc executor module (in the foreground on bottom right corner).

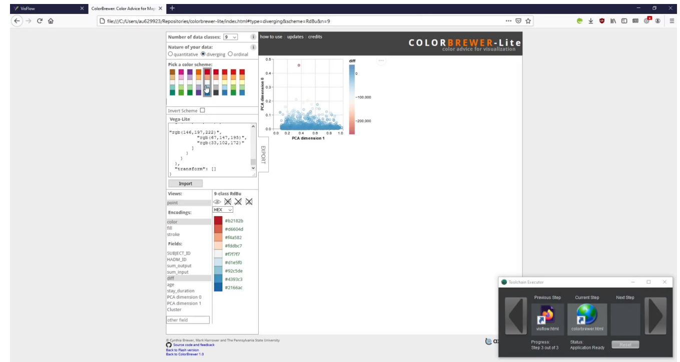

Fig. 8. Screenshot of the visual result from the ColorBrewer web tool (in the background) after setting a new color scale for the VisFlow visualization. The AnyProc executor module (bottom right corner) can be used to apply the previewed changes and return to the VisFlow tool.

of the plot to the ReVize server – but only now with the click of the button, the ReVize server is instructed to pass these updates on to other tools. Furthermore, since the Vega-Lite grammar does not capture all transformation operations we performed in KNIME, the results of the transformation are instead added to the data as additional columns. All of this is hidden from the domain expert who only sees VisFlow starting up showing the plot from KNIME. We now use VisFlow to add Parallel Coordinates that provide more detail on the cluster characteristics. We then use brushing & linking and interactive tooltips to explore the clustered data. We soon realize that one of the clusters contains patients that are over 300 years old. According to the documentation of the dataset, this is an artifact of the data anonymization process. To prevent these unusually old patients from skewing subsequent analyses, we go back to KNIME to filter out these data.

A click on the [Previous] button in the AnyProc executor brings up KNIME again. We revise our analysis pipeline to filter out patients older than 130 years. After re-running the pipeline, the cluster of patients over 300 year old is gone. Via the [Next] button, we switch again to VisFlow, relying on AnyProc to update the contents in VisFlow according to the changes we made in KNIME.

In VisFlow, we explore the data further using the special workflow node that is available in the Vega-Lite enabled version, which can render, send, and receive Vega-Lite specifications from the ReVize server. We find some curious cases of patients who were given medication, but whose fluid intake is nevertheless registered as 0 ml or even −1 ml. On top of that, we notice that the absolute fluid levels are not very useful for our analysis as they vary drastically with the duration of the hospital stay. Instead, we would like to see, for example, a patient's average fluid levels per day or the difference between a patient's overall fluid intake and output. So, we switch back to KNIME using the [Previous] button, add the necessary filters and computations to the pipeline, and adjust the PCA to use the newly computed values.

Switching to VisFlow via the [Next] button updates the charts in VisFlow (shown in Fig. 7) and we can proceed with our analysis. When investigating the newly computed difference between fluid intake and output, we would like to use a diverging color scale to better discern net loss from net gain. Unfortunately, VisFlow only offers a limited set of color schemes to choose from, and ensuring color-blind friendliness is not possible, either. To nevertheless make the desired changes to the charts' colors, we switch to the next tool in the tool chain using the [Next] button.

This opens ColorBrewer showing the visualization from the Vega-Lite node in VisFlow. As it was the case for KNIME, any update in VisFlow was already constantly pushed to the ReVize server in the background – the button click then triggered the server to distribute the latest update to the connected tools. Now we can select an appropriate color scale while previewing its effects on the visualization (Fig. 8). When done, we go back to VisFlow using the [Previous] button and there the changes made in ColorBrewer show up. Using VisFlow, we find more implausible data, such as patients who output an average of 17 liters of fluid per day or patients who output over 160 liters of fluid more than they took in. We could remove them by adding more filters to the preprocessing in additional roundtrips between KNIME and VisFlow. Yet, this no longer seems like a viable solution and we instead opt for adding the data cleaning tool OpenRefine to the beginning of our tool chain, which is further explained in Sec. 6.4.

Up to this point, we have made seven switches between tools. Without AnyProc, most of these switches would have involved, for example, navigating the file system to export and import the data, as well as navigating the start menu to find the next VA tool to be used. This is of course assuming that the tools rely on the same file format. With AnyProc, these costs of switching between tools are not gone, but they are paid once, up-front when setting up the usage flow and data flow, so that the technicalities involved no longer interrupt the analysis. The more tool switches are necessary in an analysis, the larger the benefit of using AnyProc. And as illustrated by this scenario, already a rather simple setup of two or three tools can make a considerable number of tool switches necessary.

6.4. Customizing the Usage Flow and the Data Flow

Bringing a new tool like OpenRefine into the analysis during the execution of a workflow, is a crucial feature to be able to accommodate the exploratory nature of many analyses and to dive into the unexpected findings they yield. To that end, we first extend the usage flow by opening the AnyProc editor again and dragging a new VA tool – in our case OpenRefine – into the tool chain. Once placed at the beginning of the tool chain, we also need to switch the data sources over from KNIME to OpenRefine. Finally, we need to specify how it connects to KNIME, i.e., which data channel to use to pass on its output. Since OpenRefine is not compatible with our ReVize server, we add a connection node (shown in Fig.3) to change the data channel to the local file system. With OpenRefine in place, we now have access to a powerful suite of cleaning functionalities, which makes it much easier to find and mitigate inconsistencies in the input data. For example, we can identify inconsistencies along entire columns and rows and remove those entries from the dataset with a few clicks. This way, we are able to filter the MIMIC data tables prior to their preprocessing in KNIME.

More detailed information about our demonstrating example including a video walk-through can be found on the project website: `https://vis-au.github.io/anyproc/`

7. Discussion

In our application scenario, we demonstrated how AnyProc supports cross-tool analysis of critical care data. Managing the usage flow, data flow, and control flow eased the required back-and-forth between different tools for the domain expert. We also showed the advantages of an upfront, well-defined tool chain for targeted analysis processes. However, since processes in visual analytics are often exploratory in nature, it is important to provide flexibility in the configuration of tool chains. Therefore, our editor makes it possible to adapt the tool chain on the fly, to integrate new tools, and to customize the data exchange accordingly. In this way, new parts can be added, existing parts can be adjusted, and unsuitable parts can be removed. In the following, we will now summarize our findings from our application scenario and discuss some topics raised by VA experts, who provided informal input on our platform at different stages of its realization.

7.1. Relation to our Previous Work

Our platform acknowledges all three of the aforementioned coordination layers (Sec. 2.2) in the following ways:

The Usage Flow is highly domain-dependent and influenced by preferences of the domain expert and the analysis task at hand. This makes it difficult to pre-define a universal order for switches between tools, except in instances where an agreed upon best practice is to be carried out [21]. Hence, AnyProc exposes the usage flow directly to the technical expert through the graphical editor in which to assemble the tool chain. The editor uses the metaphor of data-flow oriented systems by allowing the technical expert to include VA tools from a repository of available tools and to link them in the order of their intended use.

The Data Flow determines the technical realization of which data is passed in which way between subsequently and concurrently used VA tools. To that end, AnyProc allows further configuration of any pairwise data exchange in terms of which communication channel to use (e.g. a web server or the file system), which format to use by providing options for including necessary converters and connectors, and which data to use by providing options to connect data with the regarded tool.

The Control Flow is the actual manifestation of the usage and data flows when performing an analysis. Through the invocation of VA tools, it describes how often and how long they are used, and it thus initiates the underlying syncing of data between VA tools. To some degree, the control flow also captures the way in which the VA tools are used through the parameters set by the user in the tool. In our usage scenario, this was for example the case for the color scheme being chosen and passed on to other tools as part of the Vega-Lite specification. AnyProc provides a small persistent executor widget that enables ready access to the tool chain by featuring buttons to can complete the current processing step and move forward or backward along the analytic workflow.

7.2. Limitations of our Data Exchange Options

We want to note that any interface language used between tools inherently limits the data exchange to the quirks and features of that language. Using Vega-Lite – like we did in AnyProc – to exchange data between tools also comes as a trade-off. An obvious drawback is that any tool that does *not* support Vega-Lite cannot use this channel for data exchange, or – provided the source code is available – additional implementation overhead is required. Websocket-based data exchange in AnyProc is also limited by the features in the Vega-Lite language, which for instance prevents the exchange of network visualizations. For example, in our scenario in Sec. 6, we saw how Vega-Lite did not support all data transformations we performed in KNIME, so we instead added the results of these transformations as columns in the data shared with other tools. This technical limitation could be alleviated by supporting multiple or more expressive visualization grammars, and having the central server transpile between them, whenever two tools do not "speak the same language". Nevertheless, we chose Vega-Lite for its simplicity and wide-spread use. Another consideration is that in the websocket implementation in ReVize, the entire visualization description is shared (including, potentially, the dataset as uncompressed text), so additional burden is placed on tools that import these descriptions, which can lead to longer loading times. We still chose ReVize and Vega-Lite in AnyProc as a proof-of-concept for using a visualization grammar for websocket-based data exchange. To guide future implementations, we provide the code for adapting some exemplar visualization tools that implement the ReVize websocket channel on the project website.

7.3. Coordination of the Visual Output

At this point, AnyProc does not fully support advanced coordination of the visual output, i.e., the individual views of the tools. Regarding the tool chain, the executor currently provides basic functionality to switch between tools. Each time the tools are switched, the respective views of the active tool are displayed separately but without explicitly specifying how their contents are related to the previously active tool views. In addition, the executor allows to go forward or backward in the tool chain only one step at a time. In our application scenario, this was sufficient since our tool chain consisted of only three to four tools. In a more complex scenario, however, it quickly becomes difficult to keep track of the content being displayed as the number of tool views and switching back-and-forth between them increases. In such situations, it may also be necessary to activate more than one tool at a time or to run through several steps in the tool chain at once. Therefore, we argue that appropriate view coordination needs to be facilitated in cross-tool analyses, and attention needs to be paid to how the changing content is presented on the screen.

In our prior work [9], we have proposed various interface topologies to address view coordination issues in cross-tool analysis scenarios. The topologies include:

Tabbed UIs for exclusive use of individual tools. Here the topology is a path, meaning that the tool views are shown sequentially. The tool views are provided, for example, in a tabbed interface where each tool is assigned its own tab.

Tiled UIs for combined use of multiple tools. Here the topology is still a path, but with the possibility to display two or more tool views at the same time. The views can be shown side by side in a tiled display.

Nested UIs for flexible use of tools that goes beyond simple back-and-forth. An example is a star topology where single or multiple tool views are displayed starting from a central VA application. The central application is often shown in the background as an overview so that other tool views can be activated and superimposed on top.

Combining these interface topologies with AnyProc's existing coordination capabilities would add meaningful view layouts to its support for tool configuration and execution. The main challenge is to come up with a layout incorporating the different topologies so that there is no hard break between the different tools in the tool chain. Rather, the layout should support a seamless view transition throughout the visual data analysis. How to design such a unified interface and evaluate its effectiveness is a topic for future research.

8. Conclusion

With AnyProc, we provide a platform for the configuration and execution of VA tool chains that supports the automatic data exchange in the background. At this point, our platform can be considered a first step in the direction of full-fledged VA tool coordination.

In terms of future research, there are still many unanswered questions that need to be addressed for such a platform. This includes capturing and harnessing different aspects of the data, its analysis and visualization, potentially even in ambiguous ways. In particular the latter is a problem for data exchange between VA tools, as we have shown previously that even standardized data formats like NetCDF do not prevent different tools from interpreting the same data very differently [30]. From a domain expert's perspective, it is imperative to provide visual feedback on how far along the tool chain the analysis is and what exactly the previous and the next VA tools are. In light of our discussion (Sec. 7), we also see the development of a unified interface that facilitates the coordination of tool views as a next step for future work. Along the same lines, there is a need to link the coordinated tool chain and views with the actual analytic tasks of the given domain workflow. In previous work [31], we outlined how integrating an interactive visualization of the workflow and the different tool views can improve applicability, reliability, and reproducibility in the visual analysis of clinical data. Combining these ideas with our approach presented here, would allow the domain expert to switch between the workflow and the corresponding tool chain, see how they are connected, and what data are exchanged at each step. We believe this can greatly improve the user experience and reduce the effort required to coordinate and track progress in cross-tool data analysis scenarios.

Regardless of these open questions, the work presented here provides a first impression of what VA tool coordination can do for us: we neither have to accept the limitations of individual tools, nor do we have to carry all of the burden of integrating them with each other. A pragmatic middle ground is possible that eases the combined use of individual tools without fully integrating them.

Acknowledgments

We gratefully acknowledge funding of this research by the German Research Foundation (DFG) through project *UnIVA*, as well as by the Innovation Fund Denmark (IFD) through the Grand Solution project *Hospital@Night*. We are infinitely thankful for having had the chance to collaborate with Prof. Bodo Urban on this project in general and on this article in particular until he passed away on March 11, 2022.

References

[1] Gorton, I, Thurman, D, Thomson, J. Next generation application integration: Challenges and new approaches. In: Proc. of the 27th Annual International Computer Software and Applications Conference (COMPAC). IEEE; 2003, p. 576–581. doi:10.1109/CMPSAC.2003.1245398.

[2] Land, R, Crnković, I. Existing approaches to software integration – and a challenge for the future. In: Proc. of the 4th Conference on Software Engineering Research and Practice in Sweden. 2004, p. 1–10.

[3] Ventocilla, E, Riveiro, M. Visual analytics solutions as 'off-the-shelf' libraries. In: Proceeding of the 21st International Conference on Information Visualisation (IV). IEEE; 2017, p. 281–287. doi:10.1109/iV.2017.77.

[4] Behrisch, M, Streeb, D, Stoffel, F, Seebacher, D, Matejek, B, Weber, SH, et al. Commercial Visual Analytics Systems–Advances in the Big Data Analytics Field. IEEE Transactions on Visualization and Computer Graphics 2019;25(10):3011–3031. doi:10.1109/TVCG.2018.2859973.

[5] Berthold, MR, Cebron, N, Dill, F, Gabriel, TR, Kötter, T, Meinl, T, et al. KNIME – the Konstanz information miner: Version 2.0 and beyond. SIGKDD Explorations Newsletter 2009;11(1):26–31. doi:10.1145/1656274.1656280.

[6] Stolte, C, Tang, D, Hanrahan, P. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. IEEE Transactions on Visualization and Computer Graphics 2002;8(1):52–65. doi:10.1109/2945.981851.

[7] Ham, K. OpenRefine (version 2.5). http://openrefine.org. free, open-source tool for cleaning and transforming data. Journal of the Medical Library Association 2013;101(3):233–234. doi:10.3163/1536-5050.101.3.020.

[8] Bastian, M, Heymann, S, Jacomy, M. Gephi: An open source software for exploring and manipulating networks. In: Proc. of the International AAAI Conference on Web and Social Media (ICWSM). AAAI; 2009, p. 361–362.

[9] Schulz, HJ, Röhlig, M, Nonnemann, L, Hogräfer, M, Aehnelt, M, Urban, B, et al. A layered approach to lightweight toolchaining in visual analytics. In: Cláudio, AP, Bouatouch, K, Chessa, M, editors. Computer Vision, Imaging and Computer Graphics Theory and Applications. Communications in Computer and Information Science; Springer; 2020, p. 313–337. doi:10.1007/978-3-030-41590-7_13.

[10] Nonnemann, L, Schulz, HJ, Schumann, H, Urban, B, Aehnelt, M. A characterization of data exchange between visual analytics tools. In: Proc. of the International Conference on Information Visualisation (IV). IEEE; 2020, p. 340–349. doi:10.1109/IV51561.2020.00066.

[11] Nonnemann, L, Hogräfer, M, Schumann, H, Urban, B, Schulz, HJ. Customizable Coordination of Independent Visual Analytics Tools. In: Proc. of the EuroVis Workshop on Visual Analytics (EuroVA). Eurographics Association; 2021, p. 25–29. doi:10.2312/eurova.20211094.

[12] North, C, Shneiderman, B. Snap-together Visualization: A User Interface for Coordinating Visualizations via Relational Schemata. In: Proc. of the working conference on Advanced Visual Interfaces (AVI). ACM; 2000, p. 128–135. doi:10.1145/345513.345282.

[13] Pattison, T, Phillips, M. View coordination architecture for information visualisation. In: Proc. of the Asia-Pacific Symposium on Information Visualisation (APVIS). 2001, p. 165–169.

[14] Rogowitz, BE, Matasci, N. Metadata Mapper: a web service for mapping data between independent visual analysis components, guided by perceptual rules. In: Rogowitz, BE, Pappas, TN, editors. Human Vision and Electronic Imaging XVI; vol. 7865. SPIE; 2011, p. 165–177. doi:10.1117/12.881734.

[15] Liu, C, Wang, J, Han, Y. Mashroom+: An interactive data mashup approach with uncertainty handling. Journal of Grid Computing 2014;12(2):221–244. doi:10.1007/s10723-013-9280-5.

[16] Fekete, JD, Hémery, PL, Baudel, T, Wood, J. Obvious: A meta-toolkit to encapsulate information visualization toolkits – one toolkit to bind them all. In: Proc. of the IEEE Conference on Visual Analytics Science and Technology (VAST). IEEE; 2011, p. 91–100. doi:10.1109/VAST.2011.6102446.

[17] Waldner, M, Puff, W, Lex, A, Streit, M, Schmalstieg, D. Visual links across applications. In: Proc. of Graphics Interface (GI). 2010, p. 129–136.

[18] Fourney, A, Lafreniere, B, Chilana, P, Terry, M. InterTwine: Creating interapplication information scent to support coordinated use of software. In: Proc. of the 27th annual ACM Symposium on User Interface Software and Technology (UIST). ACM; 2014, p. 429–438. doi:10.1145/2642918.2647420.

[19] Stuerzlinger, W, Chapuis, O, Phillips, D, Roussel, N. User interface façades: Towards fully adaptable user interfaces. In: Proc. of the 19th Annual ACM Symposium on User Interface Software and Technology (UIST). ACM; 2006, p. 309–318. doi:10.1145/1166253.1166301.

[20] Tan, DS, Meyers, B, Czerwinski, M. WinCuts: Manipulating arbitrary window regions for more effective use of screen space. In: CHI'04 Extended Abstracts on Human Factors in Computing Systems. ACM; 2004, p. 1525–1528. doi:10.1145/985921.986106.

[21] Streit, M, Schulz, HJ, Lex, A, Schmalstieg, D, Schumann, H. Model-driven design for the visual analysis of heterogeneous data. IEEE Transactions on Visualization and Computer Graphics 2012;18(6):998–1010. doi:10.1109/TVCG.2011.108.

[22] Waldner, M, Bruckner, S, Viola, I. Graphical histories of information foraging. In: Proc. of the 8th Nordic Conference on Human-Computer Interaction (NordiCHI). ACM; 2014, p. 295–304. doi:10.1145/2639189.2641202.

[23] Archambault, D, Fekete, JD, Herschel, M, Jankun-Kelly, TJ, Kerren, A, Laramee, RS, et al. VAPS: Visual analytics provenance standard for cross-tool integration of provenance handling. In: Dagstuhl Report on Provenance and Logging for Sense Making. 2019, p. 42–46. doi:10.4230/DagRep.8.11.35.

[24] Mühlbacher, T, Piringer, H, Gratzl, S, Sedlmair, M, Streit, M. Opening the black box: Strategies for increased user involvement in existing algorithm implementations. IEEE Transactions on Visualization and Computer Graphics 2014;20(12):1643–1652. doi:10.1109/TVCG.2014.2346578.

[25] Hogräfer, M, Schulz, HJ. ReVize: A Library for Visualization Toolchaining with Vega-Lite. In: Proc. of Smart Tools and Apps for Graphics (STAG). Eurographics Association; 2019, p. 129–139. doi:10.2312/stag.20191375.

[26] Satyanarayan, A, Moritz, D, Wongsuphasawat, K, Heer, J. Vega-Lite: A grammar of interactive graphics. IEEE Transactions on Visualization and Computer Graphics 2017;23(1):341–350. doi:10.1109/TVCG.2016.2599030.

[27] Johnson, AE, Pollard, TJ, Shen, L, wei H. Lehman, L, Feng, M, Ghassemi, M, et al. MIMIC-III, a freely accessible critical care database. Scientific Data 2016;3(1):1–9. doi:10.1038/sdata.2016.35.

[28] Yu, B, Silva, CT. VisFlow – web-based visualization framework for tabular data with a subset flow model. IEEE Transactions on Visualization and Computer Graphics 2017;23(1):251–260. doi:10.1109/TVCG.2016.2598497.

[29] Harrower, M, Brewer, CA. ColorBrewer.org: An online tool for selecting colour schemes for maps. Cartographic Journal 2003;40(1):27–37. doi:10.1179/000870403235002042.

[30] Schulz, HJ, Nocke, T, Heitzler, M, Schumann, H. A systematic view on data descriptors for the visual analysis of tabular data. Information Visualization 2017;16(3):232–256. doi:10.1177/1473871616667767.

[31] Röhlig, M, Stachs, O, Schumann, H. Detection of diabetic neuropathy – can visual analytics methods really help in practice? In: Proc. of the EuroVis Workshop on Reproducibility, Verification, and Validation in Visualization (EuroRV3). Eurographics Association; 2016, p. 19–21. doi:10.2312/eurorv3.20161111.