Interfaces to Scripting Languages in Visual Analytics Applications

Johanna Schmidt*

*VRVis Zentrum für Virtual Reality und Visualisierung Forschungs-GmbH, Vienna, Austria
E-mail: johanna.schmidt@vrvis.at

Abstract—The need to use data visualization and visual analysis in various fields has led to the development of feature-rich standalone applications such as Tableau and MS Power BI. These applications provide ready-to-use functionality for loading, analyzing and visualizing data, even for users who are not familiar with programming and scripting. Meanwhile, data scientists have to combine many different tools and techniques in their daily work, since no standalone application can yet cover the entire workflow. As a result, a rich landscape of open source libraries is available today, covering various tasks from data analysis to modeling and visualization. To combine the best of both worlds, interfaces for scripting languages have been integrated into standalone applications in recent years. We analyzed which interfaces to six common scripting languages are offered. The interfaces offer different levels of integration and therefore support different steps of the data science workflow. In this paper we investigated the integration levels of script languages in standalone applications and divided them into four groups. We used this classification to evaluate 13 standalone visual analysis applications currently available on the market. We then analyzed which groups of applications best support which steps in the data science workflow. We found that a tight integration of scripting languages can especially support the explorative analysis and modeling phase of the data science workflow. We also discuss our results in the light of visual analysis research and give suggestions for future research directions.

Index Terms—visual analytics; visual analytics applications; scripting language interfaces

I. INTRODUCTION

Data visualization and visual analysis have successfully entered and influenced many different application areas [1]. Visualization and visual analysis techniques are today an essential part of applications in geographic information processing [2], in health care [3], in biology [4], and in Industry 4.0 [5]. The success of visualization techniques, which are used in many areas, has led to the emergence of open source libraries, but also to feature-rich, standalone visual analysis applications. These applications, such as Tableau, Microsoft Power BI and Qlik View, provide easy access to data visualization and visual data exploration for users unfamiliar with programming, scripting, data wrangling and/or data visualization design. As many of these applications are available, data visualization and visual analysis are more widely known and used today in many different domains and are used and applied by many users and domain experts.

Parallel to research in the field of visualization and visual analysis, data science has emerged as an important emerging scientific field. Data science can roughly be defined as a "concept to unify statistics, data analysis, machine learning and their related methods" in order to "understand and analyze actual phenomena with data" [6]. Data science comprises pure statistical data analysis and the interdisciplinary integration of techniques from mathematics, statistics, computer science and information science [7]. The increased interest in data science has led to the development of new software applications for data analysis, many of which are open source [8]. The use of open source technologies is a great advantage, since data scientists can then rely on a large community that can provide them with advice and support, as well as access to a wide range of libraries and plugins. Especially for Python, there are libraries for high-performance computing, numerical calculations, regression modeling and visualization, which are regularly extended and maintained [9]. Not surprisingly, studies show that the script programming languages Python and R are very important for people working with data [10].

Data science usually consists of several steps, ranging from data preparation to analysis and visualization. For this reason of a very interactive and undirected workflow [11], there are no applications yet that can cover the entire data science workflow. Data scientists must therefore always use a list of combinations of different tools, scripts and applications to achieve their goals [12]. In a recent survey [13], over 70 data science tools and applications that are commonly used by data scientists were identified. These tools are often focused on specific tasks, such as efficient data storage and access (e.g., for big-data applications), data wrangling (i.e., mapping data to another format), or automated analysis (e.g., machine learning).

The distribution of the tasks to different tools offers a great flexibility, which is not given by standalone data analysis systems. While standalone applications offer ready-to-use functions for loading and visualizing data, the functions for advanced data analysis are not updated as the open source libraries grow. As a prime example we can consider how clustering is implemented in different applications. The standalone application Tableau provides integrated k-means clustering functionality, and a k-means plugin is available for Microsoft Power BI, but the popular Python library scikit-learn already supports ten different clustering approaches [14].

To take advantage of this enormous amount of data analysis capabilities and to better integrate into the data science community, interfaces to scripting languages have been
integrated in different ways into standalone visual analysis applications. As a result, standalone applications have achieved varying degrees of integration with scripting languages. Integration can be on a rather loose level, where both components are still considered individual applications, or it can be very tight, where customer-specific interpreters are integrated into the standalone application. In this paper we investigate the implementations of different script language interfaces in current standalone visual analysis applications. We present the results of our survey, which classifies standalone applications according to their degree of integration. Scripting and programming have different meanings for data scientists in the steps of their data science workflow. We therefore evaluate which parts of the data science workflow are supported by which degree of integration. Furthermore, we discuss the results of our evaluation and summarize directions for future research in visual analytics.

II. DEFINITIONS

In this section we will give a clear definition of the terms used in this paper.

a) Standalone visual analytics applications: A standalone program can be defined as a software solution that "does not load any external module, library function" and is "not part of some bundled software" [15]. As standalone visual analytics applications we therefore consider software applications that are targeted towards visual analytics (data visualization + data analytics), do not require any additional modules (e.g., programming languages) to run on a computer, and do not require the users to have programming skills to start and use them. For example, we consider Tableau as a standalone visual analytics applications, but D3.js is not.

b) Scripting languages: By definition, scripting languages are programming languages that are interpreted, which means that they are translated into machine code when the code is run, not beforehand [16]. Examples for scripting languages typically used in data science are Python and R.

c) Interfaces to scripting languages: In computing, an interface is defined as "a shared boundary where two or more components can exchange information" [17]. A key principle of interfaces is to allow access only via precisely defined entry points and to prohibit other access by default. Components can reveal information about their interfaces, but not about internal implementation details. We define interfaces to scripting languages in standalone visual analysis applications in the same way that applications and scripting languages can exchange data, but do not know any further details. As an interface to scripting languages we consider, for example, a network interface that can be used to load data into a standalone visual analysis application, but not a hard-coded routine implemented in Python that has been integrated into a standalone application.

https://d3js.org/
We started our evaluation with the following standalone visual analytics applications mentioned in the study by Behrisch et al. [20]:

- Qlik View [2]
- TIBCO Spotfire [3]
- Tableau [4]
- SAS Visual Analytics [5]
- JMP Pro [6]
- SAP Lumira [7]
- Microsoft Power BI [8]

We additionally added the following tools mentioned in the analysis by Gartner, Inc. [21]:

- Pyramid Analytics [9]
- Looker [10]
- Infor Birst [11]
- Sisense [12]
- Yellowfin [13]

Since the VRVis recently launched a spin-off, we also added its application as another commercial visual analytics application:

- Visplore [14]

In total, this sums up to 13 standalone visual analytics applications being evaluated in the study.

After finalizing the list of standalone visual analytics applications to be analyzed, we selected scripting languages which we will use in the evaluation. From the multitude of options available, we considered the following 6 scripting languages which are, according to recent surveys [10], relevant for data scientists:

- Python
- R
- Matlab
- Perl
- Julia
- Ruby

We did not consider web-based scripting languages like JavaScript or TypeScript, since these scripting languages are considered to work in a client-server environment, and the standalone visual analytics applications are supposed to be run on a single computer. The results of our study also confirmed (see below) that these are the scripting languages application developers mainly decided to built interfaces to.

### A. Available Interfaces

In a first step it was evaluated which interfaces are provided by the selected visual analysis applications. The results are to be seen in Table I. Above all we found out that interfaces to Python and R are supported by all applications. Also the interfaces to Matlab are provided by at least five applications. This shows that application developers consider Python, R, and Matlab as the most important scripting languages for data scientists. Tableau offers the greatest variety of interfaces to different scripting languages, which confirms this application as one of the driving forces of applications in data science. We could not identify any application that offers direct interfaces to Perl. Although Julia is considered an "up-and-coming language" in data science [30], it is not yet very well supported by interfaces and APIs in the current standalone visual analysis applications we evaluated.

<table>
<thead>
<tr>
<th></th>
<th>Python</th>
<th>R</th>
<th>Matlab</th>
<th>Perl</th>
<th>Julia</th>
<th>Ruby</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infor Birst</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JMP Pro</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Looker</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Microsoft Power BI</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pyramid Analytics</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Qlik View</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAP Lumira</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAS Visual Analytics</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sisense</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Tableau</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>TIBCO Spotfire</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visplore</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yellowfin</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### B. Interfaces Classification

Based on the initial results, we started to investigate more closely the way interfaces to scripting languages were designed and implemented. Interfaces can allow a very tight or rather loose integration of scripting languages into applications. We primarily identified three types of interfaces that were implemented to script languages and therefore decided to classify interface implementations in the following way:
The standalone visual analytics application provides generic interfaces (e.g., network connectors) to communicate with other tools. These interfaces can be used by scripting languages to call functions inside the standalone application (e.g., send/get data), or by the applications to call functions in the scripting environment.

The standalone visual analytics application enables users to run scripts directly from within the application. The application uses the scripting environment installed on the system.

The standalone visual analytics application enables users to run scripts directly from within the application. The application is delivered with a built-in interpreter for the scripting language.

In the following we classified the previously discovered interfaces (outlined in Table I) into the three classes COMMUNICATIVE, SHARED, and INTEGRATIVE. The results are shown in Table II. It can be seen that the COMMUNICATIVE and SHARED approaches were mainly used when implementing interfaces to scripting languages.

For the COMMUNICATIVE approach in many cases network connector interfaces using Representational State Transfer (REST) protocols were used (Looker, Microsoft Power BI, SAP Lumira, SAS Visual Analytics). In other cases Remote Procedure Calls (Qlik View) or similar network communication sockets (Tableau) have been implemented. Infor Birst and Yellowfin can connect to Python and R over additional cloud interfaces. The advantage of these network interfaces is that they can also be accessed by other tools that are able to access the provided protocol. COMMUNICATIVE interfaces are thus not limited to scripting languages.

An example for a COMMUNICATIVE interface is given in Figure 1. The standalone application and the scripting environment are shown as isolated containers which can communicate via REST messages. With COMMUNICATIVE interfaces it is clearly defined which information can be exchanged between the standalone visual analytics application and the scripting interfaces. COMMUNICATIVE interfaces can be accessed either from inside the application, or within scripts in the

### TABLE II

CLASSIFICATION OF INTERFACES TO SCRIPTING LANGUAGES. THIS TABLE SHOWS THE DISTRIBUTION OF INTERFACES AMONG THE THREE CLASSES COMMUNICATIVE, SHARED, AND INTEGRATIVE.

<table>
<thead>
<tr>
<th>Interface</th>
<th>Language(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMUNICATIVE</td>
<td></td>
</tr>
<tr>
<td>SHARED</td>
<td></td>
</tr>
<tr>
<td>INTEGRATIVE</td>
<td></td>
</tr>
</tbody>
</table>

- **COMMUNICATIVE**
  - The standalone visual analytics application provides generic interfaces (e.g., network connectors) to communicate with other tools. These interfaces can be used by scripting languages to call functions inside the standalone application (e.g., send/get data), or by the applications to call functions in the scripting environment.

- **SHARED**
  - The standalone visual analytics application enables users to run scripts directly from within the application. The application uses the scripting environment installed on the system.

- **INTEGRATIVE**
  - The standalone visual analytics application enables users to run scripts directly from within the application. The application is delivered with a built-in interpreter for the scripting language.
scripting environment. From the scripting interface point of view, COMMUNICATE interfaces can be used to, for example, send data to the application, call visualization methods, and retrieve information like certain analysis results (e.g., outlier detection), or user interactions (e.g., selections). From the application point of view, COMMUNICATIVE interfaces can be used to extend the range of analytic functions (e.g., regression, modeling) by functions that are available in the scripting language environment.

The SHARED approach is applied by six applications for implementing interfaces. Customized connectors are provided by the applications (TabPy by Tableau), or applications directly access the scripting interfaces installed on the system (JMP Pro, Microsoft Power BI, Sisense, Yellowfin). The SHARED approach offers a lot of freedom in accessing the analytical functions of scripting languages. With a SHARED approach the standalone application makes use of the functions provided by the scripting environment (e.g., libraries), but no bidirectional communication channel is installed in this case. This means that SHARED interfaces are meant to be used from within the standalone application. Users can write and run scripts directly in the standalone visual analytics application and in this way can directly access the data currently loaded in the application.

Four applications opted for the INTEGRATIVE approach. In this case interpreters, often as additional modules, are provided by the application developers. Again, users are able to write and execute scripts directly in the standalone application. To make INTEGRATIVE interfaces work, interpreters or wrappers need to be kept up-to-date to provide the same functionalities as when directly using scripting languages in their native environment. This generates an additional overhead, since libraries in the open source scripting libraries are growing and changing quickly. It seems that most vendors of standalone visual analysis applications decided to avoid this additional effort. An example for an INTEGRATIVE interface is given in Figure 2. It can be seen that the script was created and is run in the standalone application user interface environment.

V. RELATION TO DATA SCIENCE

It is obvious that different interface implementations offer different functionality to users. Depending on the tasks to be solved, the three interface classes have different advantages and disadvantages. In this section we analyze which interface classes can best support which tasks. For this purpose, we focus on the workflow usually followed by data scientists, since these users are the main target group for combining standalone visual analysis applications with scripting capabilities.

The workflow of data scientists can be summarized into five high-level categories [22]. First, data workers usually search for suitable datasets (Discover). This research is mostly done online, and sometimes existing databases in a company are queried.

Once available, the datasets need to be brought into a certain format so that they can be used for the analysis (Wrangle). The data wrangling pass usually involves file parsing and manipulating the layout of the data. In some cases it is also necessary to combine several data sources, which might even have a heterogeneous structure. Data wrangling is considered to be an important, but also very time consuming part of the workflow.

After being available in the desired format, the data needs to be analyzed in greater depth (Profile). This involves judging the quality of the data, and estimating the suitability of the data for the analysis. Since datasets very often contain severe flaws (e.g., missing data, outliers, erroneous values), understanding the structure of the data is considered an important task. Visualization and visual analytics techniques are applied in many cases in this step. After the data could be identified as fulfilling all requirements, the datasets can be used as training data to train prediction models (Model). Modeling also involves evaluating the outcomes of the models.

All analysis results need to be reported to external people, which might be colleagues, customers, or other stakeholders (Report). Here usually dashboards or reports are used, where visualization techniques are applied to visually communicate the insights.
The steps of the data science workflow are outlined in Figure 3 and an analysis of the types of interfaces that support each step is shown in Table III. The evaluation was done based on our experience in working in data science projects. For our evaluation we will omit the Discover step, since standalone visual analytics applications and scripting languages are not targeted towards finding suitable datasets. These applications assume that suitable datasets already exist that need to be analyzed further.

In the Wrangle phase, the data must be converted into a desired format so that it can be used later for analysis and modeling. Data wrangling is about parsing and processing data files, which is often problematic and tedious when using third-party data. In many cases, data files from different sources must be combined and problems such as inconsistencies and missing elements must be solved. Although some of the evaluated applications offer ways to combine data sets (e.g., Tableau), data wrangling is not yet well supported. At this stage, standalone visual analysis applications can provide an overview of the data sources, helping data analysts understand how to manipulate the data. In the Wrangle phase, Communicative interfaces are therefore a suitable tool to quickly send data to an application where it can be visualized.

The Profile phase of the workflow summarizes the tasks involved in exploratory data analysis to understand data structure and data quality and to detect previously unknown patterns and anomalies. This phase benefits greatly from visual analysis solutions, which is why visualization techniques are increasingly used in this phase. The Profile phase also includes many circular processes in which data scientists must rethink the actions they have taken and restart the analysis process from different directions. In this phase all kinds of interfaces (COMMUNICATIVE, COMMON and INTEGRATED) can strongly support the analysis process. The most important point for analysts here is that they are able to integrate analysis results (e.g., statistical attributes), which can be calculated using scripting languages, into the visual analysis. Communicative interfaces have a certain advantage over the other two in this respect, as they offer the possibility of using all functions directly from the scripting environment and thus do not restrict users in accessing the latest scripting libraries.

The Model phase can be started after the data sets have been proven to be suitable for modeling. In this phase, a direct connection to the scripting environment is important, since analysts usually use the functionalities of open source libraries for modeling. Modeling libraries also change very quickly. In this case, the Communicative and Shared interfaces can greatly enhance the analytical capabilities of standalone applications and are therefore very well suited to support this phase in the workflow. Data scientists have to evaluate the models they create and therefore benefit from standalone applications that can interpret the models created with scripting languages. Integrated interfaces might be too limited in their supported functionalities to successfully support users in this step.

In the Report phase, scientists must summarize the results and findings in reports that are read and interpreted by colleagues, customers and other stakeholders. Reports can be static (e.g., PDF documents), but also dynamic with interactive views (e.g., dashboards). Standalone visual analysis applications, especially if they are focused on Business Intelligence (BI), are very well suited for creating interactive reports. In this step, data scientists usually rely less on dynamic interfaces to scripting languages. Nevertheless, Communicative and Integrated interfaces can also be useful in this case, e.g., for calculating key performance indicators (KPIs), which are then displayed in the report.

<table>
<thead>
<tr>
<th>TABLE III</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTERFACE IN THE DATA SCIENCE WORKFLOW. INTERFACES TO SCRIPTING LANGUAGES CAN SUPPORT THE DIFFERENT STEPS OF THE DATA SCIENCE WORKFLOW IN DIFFERENT WAYS. X INDICATES THAT THIS PARTICULAR INTERFACE TYPE IS VERY WELL SUITED TO SUPPORT THE TASKS IN THIS WORKFLOW STEP.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>Communicative</td>
</tr>
<tr>
<td>Shared</td>
</tr>
<tr>
<td>Integrated</td>
</tr>
</tbody>
</table>
VI. CONCLUSION AND FUTURE WORK

In this paper we presented a study in which we classified the types of interfaces used in standalone visual analysis applications in scripting languages.

A. Conclusion

We evaluated 13 applications currently on the market and classified the interfaces to six scripting languages commonly used by data scientists. We could identify three different types of interfaces currently used by standalone application developers. The interfaces provide different levels of integration into the standalone applications. Communication over network and communication protocols like REST or RPC is preferred over very close integration with interpreters or wrappers. These protocols could also be useful for the integration in other external environments. For example, data scientists may prefer high- and low-level programming languages like Java, C++, or C# in their workflow, mainly due to performance reasons. Network protocols in standalone applications are in this case versatile and can also be addressed by these programming languages. Open source environments and libraries are changing very quickly, and it is therefore tedious and slow to keep closely integrated wrappers and interpreters up-to-date.

The main idea behind the integration of scripting functionalities in standalone visual analytics applications is to better support the way data scientists work. Data scientists are used to stitch together different scripts and tools for different tasks, which is not supported by standalone applications alone. Through interfaces to scripting languages, these applications allow users to import and export data and analysis results in a very handy way. We analyzed how the three types of interfaces we identified integrate with the different steps in the data science workflow. Here we could see that interfaces that allow users to access a large variety of functionalities provided by the scripting interfaces better support data scientists in their highly interactive and indireted workflow. In our study we concentrated on standalone visual analytics applications being designed for data analysis. We would also like to mention that there are applications which are specifically designed to support data wrangling (e.g., Trifacta), which we did not include in the study.

B. Future Work

We believe that future research in visual analytics should continue to support data scientists in their workflows. A better integration of visual analytics into current workflows is important to foster collaboration with the data science community. Interview studies with data scientists have shown that they are very interested in exploring and integrating more advanced visualization techniques into their workflows. By providing interfaces to scripting languages or a more flexible access to visualization (e.g., by starting it from the command line) it would be possible to address a larger group of potential end users.

The study also pointed out interesting directions for future work to investigate the already developed interfaces between visualization and analytics. Visual analytics is a constant exchange between visualization and analysis, which is controlled by the users. By examining the interfaces used today, we will get a better idea of how the information flow should be implemented and which tasks are mainly needed by the users.

ACKNOWLEDGMENTS

VRVis is funded by BMK, BMDW, Styria, SFG and Vienna Business Agency in the scope of COMET - Competence Centers for Excellent Technologies (854174) which is managed by FFG.

REFERENCES


