# 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 two 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

*LEVIA'20: Leipzig Symposium on Visualization in Applications 2020*

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[1] 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.

[1]https://d3js.org/

## III. Related Work

Several applications and tools have already been developed to support a visual analytics process [18]. To get a better overview, research has been conducted on the features provided by standalone visual analytics applications currently on the market. Zhang et al. [19] and Behrisch et al. [20] conducted extensive studies on provided features and used this information to classify commercial visual analytics applications. Gartner, Inc., a global research and advisory firm, publishes an analysis of analysis and business intelligence tools every year [21]. In 2020, similar to the years before, Tableau and Microsoft Power BI have been identified as the market leaders.

Visual analytics applications are of special interest for data scientists. Kandel et al. [22] and Alspaugh et al. [12] analyzed the way data scientists work and which tools they use in their workflows. The studies show that data scientists have to rely on many different tools in all steps of their workflow. Schmidt [8] evaluated visualization tools according to featured visualization techniques. The *Chartmaker Directory* [23] creates and regularly updates a catalog for the usage of charts in different visualization tools. Holtz and Healy [24] present recent examples for the usage of visualization in data science projects online. The studies on the usage of visualization show that advanced visualization techniques are not yet applied in data science, which also confirms the so-called Interactive Visualization Gap [25] in interactive data exploration.

Research on the integration of scripting languages in visual analytics applications has been conducted from different directions. Kehrer et al. [26] presented a generic model for the integration of interactive visualization and statistical modeling. They used R as a prime example to show how scripting can be integrated into the standalone system Visplore, written in C++. Fekete [27] conducted a study on hardware and software infrastructure for visualization. He identified three important layers for interactive applications, namely visualization, analytics, and data management, which are not well integrated yet to support visual analytics applications. From a different point of view, Anderson [28] looked at the integration of scripting in entertainment applications and games engines. Mühlbacher et al. [29] looked at strategies for involving users in long-running algorithm processes. Research on the integration of scripting languages in visual analytics applications has not yet been done, and we see this paper as a starting point for future work in this direction.

## IV. Scripting Interfaces in Visual Analytics

Script languages are often used in data science projects. Several standalone applications for visual analysis started to develop interfaces to scripting languages or even to integrate them tightly into their systems. In this way, these applications can access the rich functionality offered by open source scripting languages, and visual analysis applications can be better embedded in data science workflows. In this study, we provide an overview of the standalone visual analysis applications we investigate and the approaches they use to build bridges to scripting environments.

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.

[2]https://www.qlik.com/
[3]https://www.tibco.com/
[4]https://www.tableau.com/
[5]https://www.sas.com
[6]https://www.jmp.com
[7]https://www.sap.com/products/lumira.html
[8]https://powerbi.microsoft.com/
[9]https://www.pyramidanalytics.com/
[10]https://looker.com
[11]https://www.infor.com
[12]https://www.sisense.com
[13]https://www.yellowfinbi.com/
[14]https://visplore.com/

### 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 I
AVAILABILITY OF INTERFACES TO SCRIPTING LANGUAGES. THIS TABLE SHOWS WHICH STANDALONE VISUAL ANALYTICS APPLICATIONS PROVIDE INTERFACE TO THE SCRIPTING LANGUAGES INCLUDED IN THE STUDY.

| | Python | R | Matlab | Perl | Julia | Ruby |
|---|---|---|---|---|---|---|
| Infor Birst | ✓ | ✓ | | | | |
| JMP Pro | ✓ | ✓ | ✓ | | | |
| Looker | ✓ | ✓ | | | | ✓ |
| Microsoft Power BI | ✓ | ✓ | ✓ | | | |
| Pyramid Analytics | ✓ | ✓ | | | | |
| Qlik View | ✓ | ✓ | | | | |
| SAP Lumira | ✓ | ✓ | | | | |
| SAS Visual Analytics | ✓ | ✓ | | | | |
| Sisense | ✓ | ✓ | | | | ✓ |
| Tableau | ✓ | ✓ | ✓ | | ✓ | ✓ |
| TIBCO Spotfire | ✓ | ✓ | ✓ | | | |
| Visplore | ✓ | ✓ | ✓ | | | |
| Yellowfin | ✓ | ✓ | | | | |

### 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:

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

| | COMMUNICATIVE  | SHARED  | INTEGRATIVE  |
|---|---|---|---|
| Infor Birst | *Python · R* | | |
| JMP Pro | | *Python · R · Matlab* | |
| Looker | *Python · R · Ruby* | | |
| Microsoft Power BI | *Matlab* | *Python · R* | |
| Pyramid Analytics | | | *Python · R* |
| Qlik View | *Python · R* | | |
| SAP Lumira | *Python · R* | | |
| SAS Visual Analytics | *Python · R* | | |
| Sisense | | *Ruby* | *Python · R* |
| Tableau | *Matlab · Ruby* | *Python · R · Julia* | |
| TIBCO Spotfire | *Matlab* | | *Python · R* |
| Visplore | *Matlab · Python · R* | | *Python* |
| Yellowfin | | *Python · R* | |

- **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.


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 PI, 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

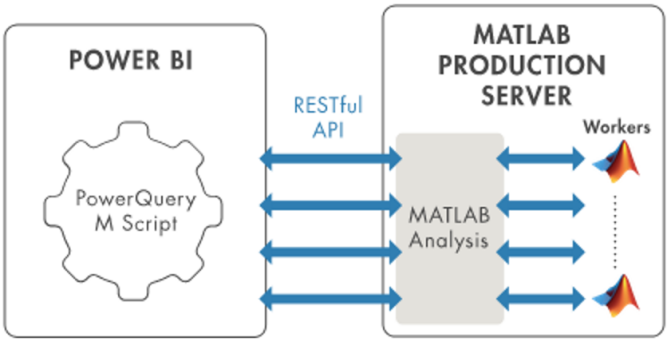


Fig. 1. Microsoft and Power BI and Matlab. The figure illustrates a COMMUNICATIVE interface between the standalone application and the scripting environment, which in this case was realized through REST messages. Illustration taken from [31].

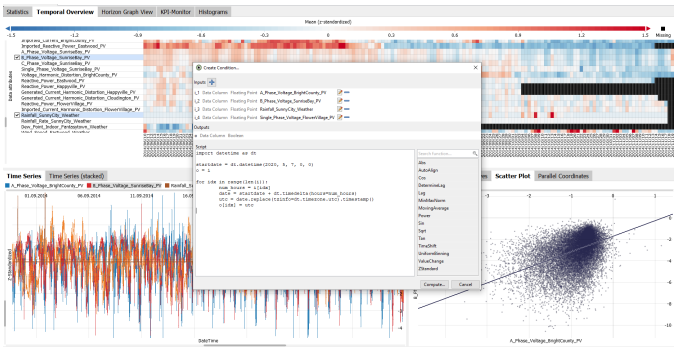

Fig. 2. Visplore and Python. With an INTEGRATIVE interface it is possible to run scripts (in this case Python) directly in the standalone application (in this case Visplore). This allows the results of the script to be integrated and used directly in the visual analysis.

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.

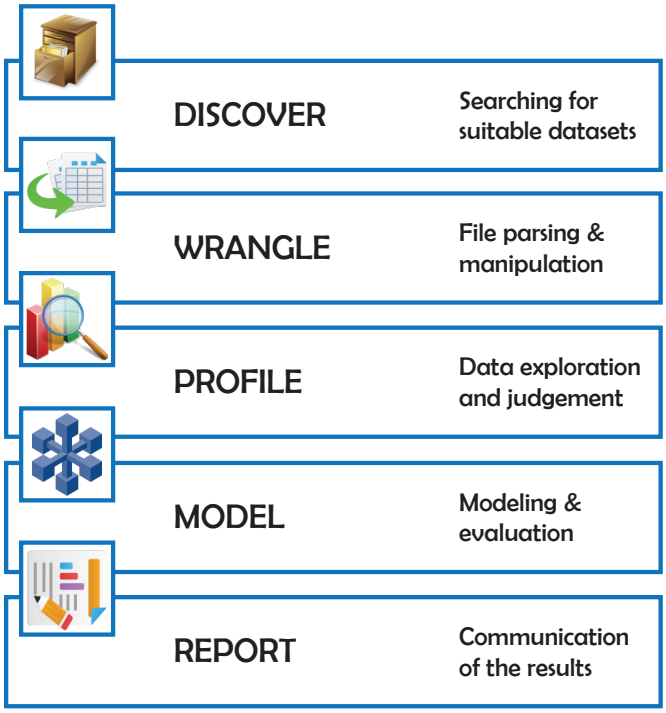


Fig. 3. Tableau and Python. With a SHARED interface it is possible to run scripts (in this case Python) directly in the standalone application (in this case Tableau). This allows to directly include the results of the script in the visual analysis. Illustration taken from [32].

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. SHARED 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, SHARED 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 III
INTERFACE IN THE DATA SCIENCE WORKFLOW. INTERFACES TO SCRIPTING LANGUAGES CAN SUPPORT THE DIFFERENT STEPS OF THE DATA SCIENCE WORKFLOW [22] IN DIFFERENT WAYS. X INDICATES THAT THIS PARTICULAR INTERFACE TYPE IS VERY WELL SUITED TO SUPPORT THE TASKS IN THIS WORKFLOW STEP.

| | *Wrangle* | *Profile* | *Model* | *Report* |
|---|---|---|---|---|
| COMMUNICATIVE | X | X | X | |
| SHARED | | X | X | X |
| INTEGRATED | | X | | X |

## 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 [33] 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 undirected 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 [15]), 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.

[15]https://www.trifacta.com/de/

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

[1] W. Cui, "Visual Analytics: A Comprehensive Overview," *IEEE Access*, vol. 7, pp. 81 555–81 573, 2019.

[2] R. de Amicis, R. Stojanovic, and G. Conti, *GeoSpatial Visual Analytics: Geographical Information Processing and Visual Analytics for Environmental Security*. Springer Netherlands, 2009.

[3] N. Kamal, S. Wiebe, J. D. Engbers, and M. D. Hill, "Big Data and Visual Analytics in Health and Medicine: From Pipe Dream to Reality," *Journal of Health & Medical Informatics*, vol. 5, no. 5, 2014.

[4] A. Slingsby and E. E. van Loon, "Exploratory Visual Analysis for Animal Movement Ecology," *Computer Graphics Forum*, vol. 35, no. 3, pp. 471–480, 2016.

[5] F. Zhou, X. Lin, C. Liu, Z. Ying, P. Xu, L. Ren, T. Xue, and L. Ren, "A survey of visualization for smart manufacturing," *Journal of Visualization*, vol. 22, p. 419–435, 2019.

[6] C. Hayashi, "What is Data Science? Fundamental Concepts and a Heuristic Example," in *Data Science, Classification, and Related Methods*. Springer Japan, 1998, pp. 40–51.

[7] M. A. Parsons, Øystein Godøy, E. LeDrew, T. F. de Bruin, B. Danis, S. Tomlinson, and D. Carlson, "A conceptual framework for managing very diverse data for complex, interdisciplinary science," *Journal of Information Science*, vol. 37, no. 6, pp. 555–569, 2011.

[8] J. Schmidt, "Usage of Visualization Techniques in Data Science Workflows," in *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, ser. VISIGRAPP '20. Valletta, Malta: SciTePress, Feb. 27–29 2020, pp. 309–316.

[9] R. Desai, "Top 10 Python Libraries for Data Science," https://towardsdatascience.com/top-10-python-libraries-for-data-science-cd82294ec266, Dec. 2019, [accessed 2020-08-19].

[10] B. Hayes, "Business Broadway: Programming Languages Most Used and Recommended by Data Scientists," https://businessoverbroadway.com/2019/01/13/programming-languages-most-used-and-recommended-by-data-scientists/, 01 2019, [accessed 2020-08-21].

[11] J. Liu, N. Boukhelifa, and J. R. Eagan, "Understanding the Role of Alternatives in Data Analysis Practices," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 66–76, 2019.

[12] S. Alspaugh, N. Zokaei, A. Liu, C. Jin, and M. A. Hearst, "Futzing and Moseying: Interviews with Professional Data Analysts on Exploration Practices," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 22–31, 2019.

[13] P. Barlas, I. Lanning, and C. Heavey, "A survey of open source data science tools," *International Journal of Intelligent Computing and Cybernetics*, vol. 8, pp. 232–261, 2015.

[14] scikit-learn developers, "Clustering," https://scikit-learn.org/stable/modules/clustering.html, 2020, [accessed 2020-07-13].

[15] Wikipedia, "Standalone software," https://en.wikipedia.org/wiki/Standalone_software, May 2020, [accessed 2020-08-01].

[16] D. Barron, *The World of Scripting Languages*. John Wiley & Sons, 2000.

[17] Wikipedia, "Software interfaces," https://en.wikipedia.org/wiki/Interface_(computing), May 2020, [accessed 2020-08-01].

[18] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon, *Visual Analytics: Definition, Process, and Challenges*. Springer-Verlag, 2008, pp. 154—-175.

[19] L. Zhang, A. Stoffel, M. Behrisch, S. Mittelstadt, T. Schreck, R. Pompl, S. H. Weber, H. Last, and D. Keim, "Visual analytics for the big data era – A comparative review of state-of-the-art commercial systems," in *In Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, ser. VAST '12, Seattle, WA, USA, Oct 14–19 2012, pp. 173–182.

[20] M. Behrisch, D. Streeb, F. Stoffel, D. Seebacher, B. Matejek, S. H. Weber, S. Mittelstaedt, H. Pfister, and D. Keim, "Commercial Visual Analytics Systems-Advances in the Big Data Analytics Field," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, 2018.

[21] L. D. Sciences, "Gartner Magic Quadrant for Analytics and Business Intelligence Platforms 2020," https://looker.com/learn/gartner-magic-quadrant, Feb. 2020, [Accessed 2020-07-09].

[22] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer, "Enterprise Data Analysis and Visualization: An Interview Study," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2917–2926, 2012.

[23] A. Kirk, "The Chartmaker Directory," http://chartmaker.visualisingdata.com/, 2020, [accessed 2020-08-13].

[24] Y. Holtz and C. Healy, "The Chartmaker Directory - Data Story," https://www.data-to-viz.com/#story, 2017, [accessed 2020-07-25].

[25] A. Batch and N. Elmqvist, "The Interactive Visualization Gap in Initial Exploratory Data Analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 278–287, Jan 2018.

[26] J. Kehrer, R. N. Boubela, P. Filzmoser, and H. Piringer, "A generic model for the integration of interactive visualization and statistical computing using R," in *Proceedings of the 2012 IEEE Conference on Visual Analytics Science and Technology (Posters)*, ser. VAST '12, Seattle, WA, USA, Oct. 14–19 2012, pp. 233–234.

[27] J.-D. Fekete, "Software and Hardware Infrastructures for Visual Analytics," *Computer*, vol. 43, no. 8, pp. 22–29, 2013.

[28] E. F. Anderson, "A Classification of Scripting Systems for Entertainment and Serious Computer Games," in *Proceedings of the Third International Conference on Games and Virtual Worlds for Serious Applications*, ser. VS-GAMES '11, Athens, Greece, May 4–6 2011, pp. 47–54.

[29] T. Mühlbacher, H. Piringer, S. Gratzl, M. Sedlmair, and M. Streit, "Opening the Black Box: Strategies for Increased User Involvement in Existing Algorithm Implementations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1643–1652, 2014.

[30] nature, "Julia: come for the syntax, stay for the speed," https://www.nature.com/articles/d41586-019-02310-3, July 2019, [Accessed 2020-08-23].

[31] M. P. BI, "Scalable Analytics with Microsoft Power BI and MATLAB Production Server," https://www.mathworks.com/products/reference-architectures/power-bi.html, 2020, [Accessed 2020-08-01].

[32] B. Beran, "Leverage the power of Python in Tableau with TabPy," https://www.tableau.com/de-de/about/blog/2016/11/leverage-power-python-tableau-tabpy-62077-0, Nov. 2020, [Accessed 2020-08-20].

[33] B. Hayes, "Business Over Broadway: Programming Languages Most Used and Recommended by Data Scientists," https://businessoverbroadway.com/2019/01/13/programming-languages-most-used-and-recommended-by-data-scientists/, Jan. 2019, [accessed 2020-10-28].